МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ

им. А. И. ГЕРЦЕНА»



Основная профессиональная образовательная программа

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) «Технологии разработки программного обеспечения»

форма обучения – очная

**Выпускная квалификационная работа**

«Разработка кроссплатформенного приложения с использованием технологии Unity для диагностики уровня речевого развития детей дошкольного возраста»

Обучающегося 4 курса

Бахурева Николая Юрьевича

Научный руководитель:

Кандидат педагогических наук, доцент

Государев Илья Борисович

Санкт-Петербург

2019

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc11270364)

[ГЛАВА 1. ОБЗОР ОСНОВНЫХ ПОНЯТИЙ 6](#_Toc11270365)

[1.1. Программное обеспечение и кроссплатформенность 6](#_Toc11270366)

[1.2. Жизненный цикл программного обеспечения 11](#_Toc11270367)

[1.3. Обзор движков и обоснование выбора Unity 15](#_Toc11270368)

[ГЛАВА 2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 20](#_Toc11270369)

[2.1. Обзор существующих аналогов 20](#_Toc11270370)

[2.2. Подбор коррекционных упражнений 23](#_Toc11270371)

[ГЛАВА 3. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА 30](#_Toc11270372)

[3.1 Постановка задачи 30](#_Toc11270373)

[3.2 Разработка программного продукта 30](#_Toc11270374)

[3.2.1 Проектирование приложения 30](#_Toc11270375)

[3.2.2 Создание приложения 42](#_Toc11270376)

[3.3 Тестирование приложения 46](#_Toc11270377)

[Выводы 50](#_Toc11270378)

[ЗАКЛЮЧЕНИЕ 51](#_Toc11270379)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 53](#_Toc11270380)

# ВВЕДЕНИЕ

В последнее время информационные технологии - перспективное средство коррекционно-развивающей работы с детьми, которые имеют речевые нарушения.

Одна из основоопределяющих стадий развития человека на раннем этапе взросления – речь. Открывая широкие возможности для коммуникации со сверстниками и взрослыми, язык позволяет войти в наш мир. Человек познает мир с помощью речи, делится с окружающими своими мыслями и взглядами. Сбалансированное речевое развитие является необходимым компонентом для полноценной коммуникации в социуме [1].

Развитие речи происходит с возрастающими темпами. К пяти годам сбалансированно развитый ребенок способен правильно произносить все звуки родного языка, обладает достаточным словарным запасом, усвоил основы грамматики, владеет начальными формами связной речи (диалог и монолог), которые способствуют установлению контакта с людьми в окружении ребенка.

Простое осознание явлений родного языка начинается в дошкольном возрасте. Ребенок знакомится с фонетическим строением слова, с синонимами и антонимами, со словесным составом предложения и многими другими понятиями. Он выявляет закономерности построения развернутого высказывания (монолога), осваивает правила ведения диалога. Наблюдение и воспроизведение языковых и речевых явлений формирует у детей произвольность речи, позволяет овладеть более продвинутыми способами коммуникации – чтением и письмом.

Задержка или другое нарушение в ходе речевого развития отрицательно сказывается на деятельности, поведении и формировании личности ребенка. Упущения и недостатки в развитии речи, как правило, четко проявляются в дошкольном возрасте.

Стоит упомянуть, что дети дошкольного возраста с речевыми нарушениями демонстрируют снижение интереса к обучению, отсутствие мотивации для посещения дополнительных занятий, повышенную утомляемость. Нужны новые подходы и технологии, чтобы заинтересовать таких детей и предоставить им мотивацию, сделать их обучение осознанным.

Как упоминалось выше, одна из проблем – повышенная утомляемость. Коррекционная работа может быть кропотливой, непростой и долгой – в традиционном виде. Использование же разнообразных новаторских методов предотвращает утрату концентрации и поддерживает у детей интерес к самосовершенствованию – таким образом, логопедическая работа продвигается в разы эффективнее.

Использование компьютерных программ и мобильных приложений может служить одним из средств оптимизации процесса коррекции речи – как для повышения мотивации, так и для, непосредственно, реализации упражнений и коррекционных задач, которые невозможно задать ребенку без использования современных технологий.

Всё вышеперечисленное обуславливает актуальность исследования.

Объект исследования – инструменты и средства информационных технологий, информатики и вычислительной техники в области коррекционной педагогики.

Предмет исследования – кроссплатформенное приложение с использованием технологий Unity для диагностики уровня речевого развития детей дошкольного возраста.

Целью исследования является разработка программного обеспечения, представляющего собой кроссплатформенное приложение с использованием технологий Unity для диагностики уровня речевого развития детей дошкольного возраста.

Для достижения цели исследования необходимо решить следующие задачи:

* описать кроссплатформенное программное обеспечение;
* выбрать модель жизненного цикла программного обеспечения;
* сделать обзор вариантов базового программного обеспечения и обосновать выбор Unity;
* исследовать существующие аналоги;
* подобрать коррекционные упражнения;
* сформулировать задачу разработки;
* разработать приложение:
  + спроектировать приложение;
  + реализовать продукт в соответствии с проектом;
* протестировать полученный программный продукт.

Практическая значимость работы заключается в том, что наличие кроссплатформенного приложения, реализующего нетрадиционный подход к коррекционной педагогике, повысит мотивацию детей к дальнейшему развитию речи и позволит заниматься совершенствованием речевого аппарата в удобном формате.

Структура работы обусловлена целями и задачами исследования. Работа содержит введение, три главы, первая и третья поделены на три параграфа, вторая – на два, заключение и список использованной литературы.

Гипотеза – технологии Unity подходят для разработки кроссплатформенных приложений любой сложности.

Методы исследования - анализ и синтез информации – использованы для сравнения современных дидактических комплексов, нацеленных на диагностику речевого развития, а также для выбора необходимого инструментария для разработки программного продукта.

# ГЛАВА 1. ОБЗОР ОСНОВНЫХ ПОНЯТИЙ

## Программное обеспечение и кроссплатформенность

Рассмотрим понятие программного обеспечения и кроссплатформенного программного обеспечения, как его частный случай.

Программное обеспечение (ПО) – это собирательное наименование программ, которые обеспечивают функционирование электронно-вычислительной техники и решение с их помощью задач предметных областей. Программное обеспечение является неотъемлемой частью компьютерной системы, логическим продолжением технических средств и определяет сферу применения компьютера [4].

Кроссплатформенное ПО — программное обеспечение, которое работает более чем на одной аппаратной платформе, а также операционной системе.

Обеспечивается благодаря использованию высокоуровневых языков программирования, сред разработки и выполнения, поддерживающих условную компиляцию, компоновку и выполнение кода для различных платформ.

Среди операционных систем (ОС) для персональных компьютеров лидирующее место уверенно занимает Windows. Если рассматривать все версии, начиная с Windows XP и заканчивая Windows 10, в общей сложности, 91,5% компьютеров в мире работают именно под управлением данной операционной системы. И, всё же, остается 8,5% техники, которая работает под управлением других операционных систем. Эту долю делят между собой два наиболее значимых игрока на рынке программного обеспечения – Mac OS (6,2%) и Linux (2,3%).

Что касается рынка мобильных устройств, там наблюдается следующая ситуация – лидирующее положение занимает Android (71,5%), на втором месте – iOS(19,7%). Далее – Windows Phone и прочие (Tizen, BlackBerry OS).

Даже не углубляясь в статистику распределения, уже можно сделать вывод, что кроссплатформенное программное обеспечение крайне востребовано в условиях современного рынка операционных систем.

На заре компьютерной эры, когда еще не было таких гигантов IT-индустрии, как Microsoft, Google и Apple, возникла необходимость в совместимости программных продуктов. Каждая операционная система имеет свой интерфейс для взаимодействия с программным обеспечением. Каждая программа имеет свой формат данных. Всё это подводит к необходимости существования такой категории ПО, как кроссплатформенное программное обеспечение.

Запуск одного и того же прикладного ПО на различных архитектурах без необходимости обеспечивать совместимость на уровне ОС реализуется путём стандартизации языков, библиотек, компиляторов и среды выполнения, а также путём перехода на исполнение ПО в виртуальной машине и стандартном окружении, которые реализуются для каждой платформы и гарантируют единообразное исполнение ПО независимо от платформы.

Современные подходы к разработке кроссплатформенного программного обеспечения:

* Единое стилистическое решение;
* Адаптивный интерфейс;
* Гибридная схема.

При едином стилистическом решении программа должна выглядеть одинаково под всеми операционными системами. Положительная сторона такого подхода – закрепление элементов управления. Отрицательная – отличие программы от общего стиля операционной системы.

Программа, сконструированная по принципу адаптивности, должна легко интегрироваться в интерфейс операционной системы за счет изменения тем оформления. Предполагается полное или частично автоматическое определение языковых параметров и оптимальных размеров экрана, под которые должно подстроиться программное обеспечение. Положительные стороны – мимикрия под стиль операционной системы. Отрицательные – сложность и высокая стоимость разработки;

Гибридная схема сочетает в себе положительные и отрицательные стороны предыдущих подходов. Легкая интеграция и частичная автоматизация настройки и, вместе с тем, различие в стилях оформления и сложности, связанные с компоновкой элементов управления.

Рассмотрим классификацию программного обеспечения.

Программное обеспечение современных компьютеров включает множество разнообразных программ, которые можно классификационно разделить на три группы:

* Системное программное обеспечение;
* Прикладное программное обеспечение;
* Инструментальное обеспечение.

Системное программное обеспечение – это набор программ, которые управляют функционированием компьютера и выполняют различные вспомогательный набор работ, такие как:

* управление ресурсами;
* создание копий информации;
* проверка работоспособности периферийных устройств;
* выдача справочной информации о компьютере.

Они предназначены для всех категорий пользователей, используются для эффективной работы компьютера и пользователя, а также эффективного выполнения прикладных программ.

Центральное место среди системных программ занимают операционные системы.

Операционная система – это совокупность программ, которые предназначены для управления загрузкой, запуском и выполнением пользовательских программ, а также для планирования и управления вычислительными ресурсами ЭВМ, т.е. управления работой ПЭВМ с момента включения до момента выключения электропитания [5]. ОС загружается автоматически при включении компьютера, ведет диалог с пользователем, осуществляет управление компьютером, его ресурсами (оперативной памятью, дисковым пространством и т.д.), запускает другие программы на выполнение и обеспечивает пользователю и программам удобный способ общения – интерфейс – с устройствами компьютера. Иначе говоря, операционная система обеспечивает функционирование и взаимосвязь всех составляющих компьютера, а также предоставляет пользователю доступ к его аппаратным возможностям. ОС определяет производительность использования ЭВМ, степень защиты данных, выбор программ, с которыми можно работать на компьютере, требования к аппаратным средствам. Примерами ОС являются MS DOS, OS/2, Unix, Windows 9х, Windows XP.

Прикладное программное обеспечение предназначено для решения задач пользователя. В его состав входят прикладные программы пользователей и пакеты прикладных программ (ППП) различного назначения.

Прикладная программа пользователя – это любая программа, способствующая решению какой-либо задачи в пределах данной предметной области. Прикладные программы могут использоваться либо автономно, либо в составе программных комплексов или пакетов.

Пакеты прикладных программ (ППП) – это программные комплексы, организованные специальным образом, которые рассчитаны на общее применение в определенной предметной области и дополненные соответствующей технической документацией.

Различают следующие типы ППП:

* ППП общего назначения;
* методо-ориентированные ППП;
* проблемно-ориентированные ППП;
* интегрированные ППП.

ППП общего назначения представляют собой универсальные программные продукты, предназначенные для автоматизации самого широкого класса задач пользователя. К ним относятся текстовые, табличные процессоры, системы управления базами данных, презентаций, графические редакторы, издательские системы, электронные словари и системы перевода. Часто вышеупомянутые средства объединяются в так называемый офисный пакет.

Методо-ориентированные ППП включают в себя реализацию математических методов решения задач, реализации алгоритмов, визуализации процессов. К ним относятся системы компьютерной алгебры и статистического анализа данных.

Проблемно-ориентированные ППП предназначены для решения определенной задачи в конкретной предметной области. К ним относятся пакеты бухгалтерского учета, банковские и информационно-правовые системы.

Интегрированные ППП представляют собой набор нескольких программных продуктов, объединенных в один инструмент.

Осталось рассмотреть инструментальное программное обеспечение – оно служит для разработки нового ПО и включает в себя:

* системы программирования;
* инструментальные среды;
* системы моделирования.

Системы программирования включают в себя:

* редактор исходного кода;
* транслятор;
* библиотеки подпрограмм.

Примерами являются Pascal и Basic.

Инструментальные среды представляют собой обертку над системой программирования и служат для быстрого и удобного манипулирования возможностями того или иного языка, зачастую включая в себя средства визуального программирования.

К системам моделирования относят программное обеспечение, которое заменяет исследуемую систему моделью, с достаточной точностью описывающей реальную систему, и проводит с моделью эксперименты с целью получения данных об изначальной системе.

## Жизненный цикл программного обеспечения

В основе процесса разработки любого программного продукта лежит понятие жизненного цикла. Жизненный цикл представляет собой некоторую абстрактную модель создания и последующего применения программного продукта, отражающую все его возможные состояния [2].

Принято выделять ряд обязательных этапов жизненного цикла:

* формулирование требований – это первый этап жизни программного обеспечения. Здесь важно не только составить список требований к будущему продукту, но также провести уточнение с обеих сторон – заказчика и исполнителя, после чего задокументировать результат;
* проектирование – на данном этапе происходит формирование логической модели продукта с учетом списка требований, полученного на предыдущем этапе. Кроме того, этап проектирования включает в себя разработку спецификаций будущего программного продукта;
* кодирование – процесс непосредственного создания программного кода на основе спецификаций, полученных ранее;
* тестирование и отладка – на данном этапе проводится проверка корректности функционала полученного программного продукта. Важно зафиксировать, что поведение соответствует установленным требованиям заказчика. В случае выявления каких-либо расхождений с требованиями, они устраняются на этом же этапе;
* документирование – этап разработки документации к полученному программному продукту;
* внедрение – этап подготовительных работ, которые требуются для начала эксплуатации программного продукта, например, обучение персонала;
* эксплуатация – использование программного продукта;
* сопровождение – решение вопросов и проблем, возникающих на стороне заказчика в процессе эксплуатации [3].

Существует несколько типовых моделей жизненного цикла программного обеспечения:

* Каскадная;
* Спиральная;
* Инкрементная.

Каскадная – классическая модель разработки, отличительной чертой которой является однозначная хронологическая последовательность ее этапов. Достоинство данной модели заключается в ее простоте. Однако, область использования каскадной модели очень узка - она может применяться только в идеальных процессах, когда все исходные требования сформулированы заранее и не подлежат редактированию в процессе разработки. Графическое представление этапов каскадной модели приведено на рисунке 1.

Разработка требований

Проектирование

Реализация

Тестирование

Эксплуатация и поддержка

Рисунок 1 – Каскадная модель

Спиральная – модель, являющаяся усовершенствованием каскадной. В основе спиральной модели лежит идея разработки итерациями, в состав каждой из которых входят все обязательные этапы жизненного цикла. Таким образом, результат каждой итерации представляет собой работоспособный программный продукт. Достоинство данной модели заключается в том, что она позволяет совмещать этапы разработки и тестирования, сокращая тем самым возможное число ошибок, выявляемых на финальных этапах разработки. Также важно отметить, что непосредственно в процессе разработки заказчик может получать рабочие версии программного продукта. Недостатком данной модели является рост риска потерь концептуальной целостности в том случае, когда архитектурные решения существенно расходятся между отдельными итерациями. Графическое представление этапов спиральной модели приведено на рисунке 2.



Рисунок 2 – Спиральная модель

Инкрементная – данная модель является частным случаем спиральной модели. Процесс разработки здесь также распадается на итерации, однако, программный продукт разрабатывается по принципу приращений - на каждом этапе жизненного цикла разработчик обладает возможностью использования данных, полученных на предыдущих итерациях. Новые данные формируются не только в процессе разработки, но и в качестве результатов использований уже существующих данных. Отличительной чертой инкрементной модели является простота реализации подмножества требований к программному продукту и совершенствование модели в последовательности релизов до тех пор, пока заявленные требования не будут реализованы в полном объеме. Результат каждой итерации может менять программный продукт, добавляя новые функции. Программный продукт, получаемый на каждой итерации называется инкрементом. В состав первого инкремента обязательно должны входить базовые функции программного продукта. Дальнейшие инкременты предназначены для расширения этих функций до тех пор, пока не будет выполняться исходный набор требований. Каждый инкремент может поставляться заказчику в виде самостоятельной версии программного продукта. Одной из особенностей инкрементной модели является большое количество циклов разработки при их малой продолжительности, а также небольшие отличия между инкрементами соседних циклов. К достоинствам данной модели относятся:

* постоянство требований в рамках инкремента;
* наличие возможности редактирования требований;
* наличие возможности получения работоспособного программного продукта по итогам каждого инкремента.

Недостатком инкрементной модели является сложность планирования работ. Графическое представление этапов инкрементной модели жизненного цикла приведено на рисунке 3.

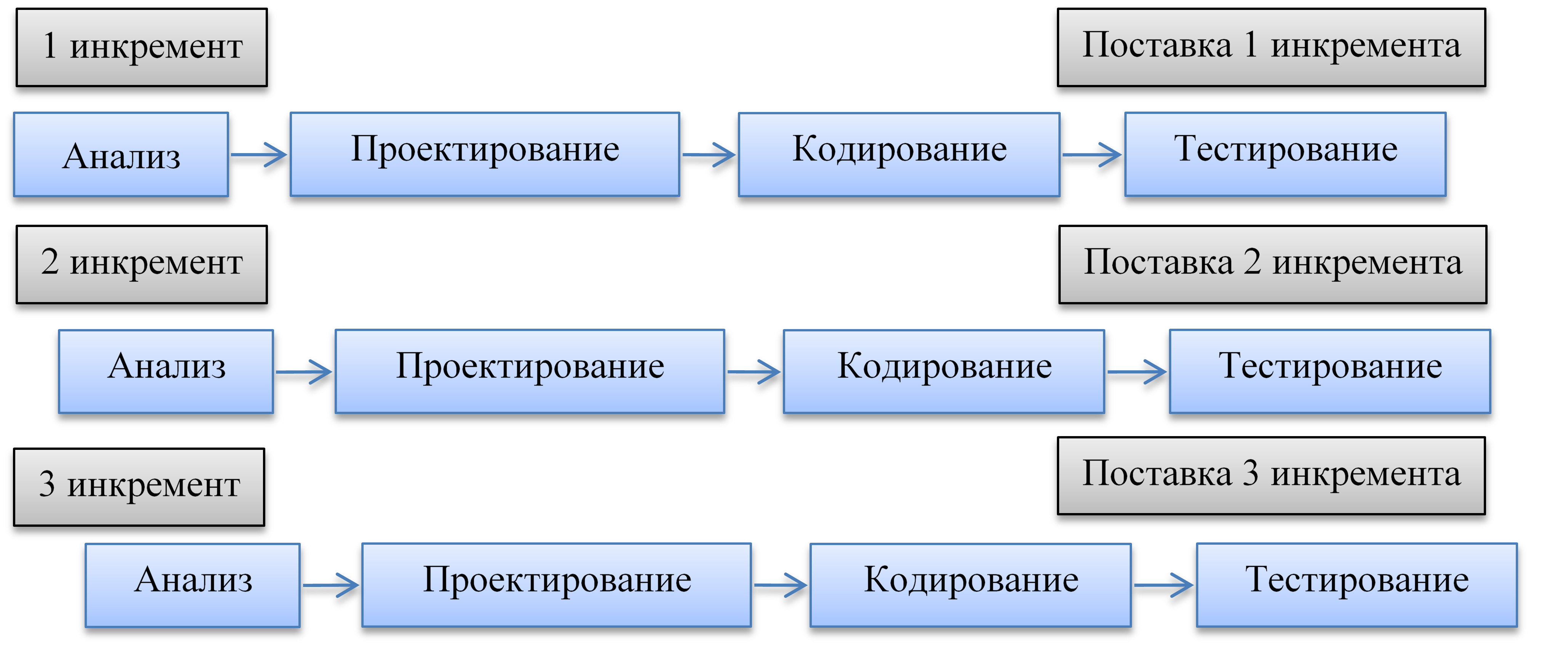


Рисунок 3 – Инкрементная модель

## Обзор движков и обоснование выбора Unity

Рассмотрим и опишем основные отличительные особенности некоторых наиболее популярных движков для создания кроссплатформенного приложения:

* Unity 5;
* Unreal Engine 4;
* Cry Engine 3.

Unity 5 - базовое программное обеспечение для разработки двухмерных и трехмерных приложений и игр под различные аппаратные платформы и операционные системы. У Unity есть две версии: бесплатная и платная. Они отличаются функционалом. Стоит уточнить, что для разработки небольшого проекта более чем достаточно и тех возможностей, что предоставляет бесплатная версия. Более глубокий и широкий функционал необходим большим студиям при разработке профессиональных глобальных проектов. Бесплатная версия Unity3d поддерживает только Android, Web Player, PC-платформы. Полная версия охватывает все самые известные платформы, такие как: PC, Linux, Mac, Windows Store, IOS, Android, Windows Phone 10 Store, Blackberry 10, Wii U, PS3, Xbox 360, PS4, Xbox One. Есть возможность делать софт, предназначенный под очки и шлемы виртуальной реальности: Hololens, Oculus Rift, StarVR и прочие, а также писать программы для Kinect 2.0, LeapMotion. Полную версию Unity 5 можно адаптировать под свои нужды с помощью ряда расширений [6].

Unity3d имеет простой, интуитивно понятный интерфейс, основанный на механике «Drag and Drop». Движок не имеет русификации. Unity разбит на несколько окон:

* Hierarchy, где находятся названия всех объектов на сцене, которые можно группировать и легко переходить по ним;
* Scene, где можно рассмотреть определенную сцену под нужным вам ракурсом;
* Inspector, который поможет с настройкой выделенного объекта;
* Project, где видны все материалы проекта;
* Toolbar, где предоставлено меню с инструментами.

Unity 5 поддерживает два языка: C# и Javascipt [7]. Разработчику необходимо знать один из языков в совершенстве, а другой на среднем уровне, так как некоторые моменты Unity 5 делает только на одном из двух языков, или это делается намного труднее, чем на другом языке программирования. Предпоследняя версия Unity3d, а именно Unity 4, поддерживала язык программирования Boo (диалект Phyton), но его убрали из 5-ой версии, так как им практически никто не пользовался. Расчеты физики в Unity 5 производит NVIDIA PhysX.

Объекты в Unity3d могут быть пустыми, (чтобы объединить несколько объектов в одну группу, т.е. сделать их дочерними GameObject), содержать компоненты, с которыми взаимодействуют скрипты, могут быть названы одним и тем же именем, им могут быть присвоены теги, которые служат для навигации по объектам из скриптовой составляющей. К объектам в Unity3d можно присоединить обработчики столкновений:

* box collider - куб, в который попадает модель объекта;
* sphere collider – сфера;
* character collider – обработчик столкновений, который был специально введен в Unity 5 для использования под персонажей;
* mesh collider - обработчик столкновений, повторяющий геометрию объекта;
* wheel collider - обработчик столкновений для колес;
* terrain collider - обработчик столкновений для площадки, которую используют для отображения земли.

Для создания анимации моделей можно использовать как стороннее ПО (3Ds Max, Blender), так и встроенный редактор Unity.

Материалы в Unity 5 играют важную роль. Импортированные текстуры в Unity3d прикрепить к объекту нельзя, необходимо создать материал, который можно присваивать игровому объекту. К назначенным материалу шейдерам будут присвоены текстуры. Шейдеры можно редактировать прямо в Unity3d. Unity позволяет генерировать карты нормалей, карты освещения и каналы прозрачности. Также, имеется поддержка MIP-текстурирования.

В полной версии Unity 5 возможно полное настраивание шейдеров, а в бесплатной - нет.

Unity 5 имеет две очень важные особенности: «Occlusion Culling» и «Level Of Detail». Обе модели позволяют сильно снизить нагрузку на центральный процессор, благодаря грамотной детализации. Например, в играх жанра 2D и 3D Runner при преодолении определенной дистанции все, что было позади вас, удаляется, а то, что впереди вас, генерируется. Таким образом, при длительной игре память распределяется рациональным образом. «Occlusion Culling» не обрабатывает геометрию и обработчики столкновений объектов, находящихся не в поле зрении камеры, а «Level Of Detail» снижает степень детализации объекта по заданным уровням, вплоть до изменений текстур, моделей и применяемых эффектов, причем разработчик сам настраивает эту систему. То есть небольшой проект может использовать невероятную детализацию на больших расстояниях, тогда как крупные проекты выставляют этот параметр на минимум.

На официальном сайте Unity есть специальный раздел, в котором можно найти статистику по игровым движкам. По этим данным Unity 5 используют более 50% разработчиков видеоигр. 20% принадлежат Unreal Engine, а остальные игровые движки - 30%.

Лицензирование – бесплатная версия Unity не может быть использована в коммерческих целях при ежегодном доходе в 100000$, исходный код движка при покупке платной версии не предоставляется.

Unity 5 обладает весомым количеством преимуществ перед другими игровыми движками – и вот некоторые из них:

* сообщество пользователей Unity 5 на сегодняшний момент является самым большим в мире, что предоставляет доступ к готовым решениям, материалам, методам;
* совместимость проектов, созданных на ранних версиях движка;
* наличие структурированной, полной и подробной документации с примерами;
* продуманный и дружественный к пользователю интерфейс;
* быстрая сборка проектов;
* возможность создать один проект для всех целевых платформ;
* наличие бесплатной полнофункциональной версии;
* компонентно-ориентированный подход к моделированию проекта;
* свободная работа при разрядности ОС x32.

Unreal Engine 4 — это набор инструментов для разработки игр, имеющий широкие возможности: от создания двухмерных игр на мобильные до AAA-проектов для консолей [13].

Полностью бесплатен со 2 марта 2015 года. Предоставляется доступ к исходному коду. Если прибыль разработчиков проекта составляет более 3000$ за календарный квартал, должны производиться отчисления разработчикам движка.

Список поддерживаемых платформ: Windows, PS3, XBox360, WiiU, PS4, XBoxOne, Android, iOS, WindowsPhone, SteamOS, Linux, OculusRift, HTML5.

Значительно сокращенное время разработки за счет использования модели визуального программирования – движок можно полноценно использовать, имея минимальные знания в программировании.

Основной язык программирования – С++.

Так же, как и Unity, имеет развитое сообщество и магазин готовых модулей, которые можно применить в разработке.

Наличие подробной документации и большое количество видео-уроков, которые в деталях разъясняют методы работы с движком.

Cry Engine 3 - движок, разработанный компанией Crytek [14].

Список поддерживаемых платформ: Windows, PS3, Xbox 360, WiiU.

Отсутствие развертывания на большинстве современных платформ, неудовлетворительная работа службы поддержки, устаревшая документация, маленькое сообщество и сложный процесс сборки проекта заставляют усомниться в целесообразности сравнения Cry Engine с Unity и Unreal Engine. Cry Engine 3 обладает качественной реализацией технологии отрисовки в реальном времени и позволяет моделировать сцены с высоким уровнем детализации, но на этом его преимущества заканчиваются.

Резюмируя всё вышесказанное, становится очевидным, что лучшим выбором для разработки небольшого кроссплатформенного приложения будет движок Unity 5.

Таким образом, была изучена область программного обеспечения, **требования**, предъявляемые к программному продукту, в частности, **кроссплатформенность**, была выбрана **спиральная модель** разработки, что позволило совместить этапы разработки и тестирования. После изучения самых **популярных движков** для реализации поставленных задач был выбран движок **Unity 5** – бесплатный, целостный, простой в освоении и полнофункциональный в использовании.

# ГЛАВА 2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 2.1. Обзор существующих аналогов

Рассмотрим представителей интересующей нас отрасли:

* программно-дидактический комплекс «Мерсибо Интерактив»;
* программно-методический комплекс «Игры для Тигры».

Мерсибо Интерактив - программно-дидактический комплекс с кооперативными и соревновательными развивающими играми.

Подходит для занятий с детьми от 4 до 10 лет.

Типы задач:

* совместные - дети кооперироваться и действовать коллективно, чтобы успешно решить задачу. Такие игры учат детей сотрудничать, помогать друг другу и распределять роли в команде. В комплексе 15 совместных игр;
* соревновательные - дети конкурируют между собой, кто качественнее всех выполнит задание или уложится в отведенное время. Такие игры учат детей максимальному сосредоточению, последовательному достижению финала, корректному поведению в ситуациях победы и поражения;
* соревновательно-последовательные – игры, в которых каждая задача разбита на несколько этапов. Такую задачу дети проходят пошагово, при этом они конкурируют и набирают очки прогресса. В комплексе 8 таких игр;
* индивидуальные - в этих играх дети принимают решения и действуют самостоятельно от начала до конца. В комплексе 37 индивидуальных игр, которые развивают мышление, речь, навыки чтения и счета, моторику, знакомят с русским языком и окружающим миром.

Охват тем:

* игры для детей до 5 лет:
  + словарный запас;
  + кругозор;
  + зрительное внимание и память;
  + мышление;
* работа с экраном:
  + навык работы с интерактивным оборудованием;
  + мелкая моторика;
  + глазомер;
* звуки и буквы:
  + гласные и согласные буквы;
  + запоминание образа букв;
  + выделение ударных гласных;
* окружающий мир:
  + развитие кругозора;
  + определение времени по часам;
  + изучение частей тела;
  + расширение словарного запаса;
* математика:
  + работа с двузначными числами;
  + сравнение чисел;
  + уравновешивание весов;
* чтение и русский язык:
  + звуковой состав слова;
  + чтение слогов, слов и предложений;
  + расстановка предлогов;
  + работа с непроизносимыми согласными;
  + составление предложений из слов;
  + поиск слова в другом слове;
* внимание, память, логика:
  + поиск парных и неправильных элементов;
  + логические ряды;
  + повторение схемы по образцу;
  + составление узора;
  + сравнение изображений.

Программно-дидактический комплекс доступен для покупки в двух вариантах:

* на съемном носителе информации – 11490 руб.;
* в цифровом дистрибутиве – 9990 руб.

Для применения программно-дидактического комплекса необходимым является наличие интерактивной доски или стола с функцией «Мультитач».

Игры для Тигры - программно-методический комплекс, содержащий программу «Игры для Тигры» для персонального компьютера и учебно-методическое пособие с подробными методическими рекомендациями ее пошагового применения в коррекционно-образовательном процессе.

Программно-методический комплекс спроектирован на основе методик обучения детей с отклонениями развития Г. А. Каше, Л. В. Лопатиной, Н. В. Серебряковой, Р. И. Лалаевой, Н. С. Жуковой, Е. М. Мастюковой, Т. Б. Филичевой, а также Программы воспитания и обучения в детском саду под редакцией М. А. Васильевой.

Программно-методический комплекс предлагает более 50 упражнений с несколькими уровнями сложности, объединенных в четыре тематических блока:

* просодика - комплекс элементов, включающий ритм, темп, тембр и логическое ударение, служащий на уровне предложения для выражения различных синтаксических значений и категорий, а также экспрессии и эмоций;
* звукопроизношение;
* фонематика;
* лексика.

Программно-методический комплекс доступен для покупки на DVD.

Также, можно получить цифровой дистрибутив демо-версии комплекса.

Активация лицензии рассчитана на один персональный компьютер. Цена одной лицензии составляет 2700 руб.

## 2.2. Подбор коррекционных упражнений

В рамках исследования методик диагностики речевого развития детей дошкольного возраста, было рассмотрено несколько направлений анализа:

* обследование фонематики:
  + восприятие;
  + синтез;
  + анализ;
  + представление;
* обследование словаря;
* обследование грамматического строя речи.

Фонематика (фонематический слух) - идентификация (анализ и синтез) звуков частей речи, которое является необходимой основой для адекватного понимания смысла сообщения. При недостаточном уровне речевого различения звуков человек может воспринимать информацию в критически искаженном виде.

Развитие фонематического слуха у детей необходимо для успешного обучения их чтению и письму. Дети часто путают близкие по звучанию фонемы, что в некоторых случаях тормозит развитие связной речи, не говоря уже про общее искажение информации как от ребенка к окружающей среде, так и от окружающей среды к ребенку. В дальнейшем развитие фонематического слуха необходимо для успешного обучения иностранным языкам.

Фонематическое восприятие — это способность воспринимать звуковой состав слова и выявлять его фонетические характеристики, а именно:

* Количество слогов в слове;
* Количество звуков в слове;
* Какой согласный звук расположен в конце слова;
* Какой гласный звук расположен в середине слова.

Сформированное фонематическое восприятие является залогом четкого произнесения звуков, правильной слоговой структуры слов и основой легкости овладения грамматическим строем языка, а значит успешного освоения письма и чтения.

Набор упражнений для диагностики и развития фонематического восприятия:

* Определить наличие или отсутствие заданного звука в слове;
* Ребенок, имея на руках несколько изображений, должен выбрать только те, в названии которых присутствует заданный звук;
* Прослушать предложение и выбрать слово, в котором есть заданный звук;
* Подобрать и соотнести картинки к словам-паронимам.

Фонематический синтез – это умение соединять отдельные звуки в целое слово с последующим узнаванием этого слова, мысленно составленного из звуков. Под фонематическим синтезом понимаются умственные действия по синтезу звуковой структуры слова – слияние отдельных звуков в слоги, а слогов в слова.

Набор упражнений для диагностики и развития фонематического синтеза:

* Составить слова из звуков, данных в нарушенной последовательности;
* Составить слова из звуков, которые произносятся последовательно и изолированно.

Фонематический анализ – это умение расчленять слышимое слово на составляющие его звуки, четко представлять себе его звуковую структуру. Под фонематическим анализом понимаются умственные действия по анализу звуковой структуры слова – разложение его на последовательный ряд звуков, подсчет их количества, классификация.

Недостаточное развитие анализа и синтеза проявляется:

* при письме в искажениях структуры слова и предложения. Особенно распространенными при этом будут искажения звуко-буквенной структуры слова. Наиболее характерны следующие ошибки: пропуски согласных при их стечении, пропуски гласных, перестановки букв, добавление букв, пропуски, добавления, перестановка слогов;
* при чтении - пропуски согласных звуков при стечении, вставки гласных между согласными при их стечении, перестановки звуков, пропуски и вставки звуков при отсутствии стечения согласных в слове.

В основу обследования фонематического анализа положена методика Л.Ф. Спировой, которая рекомендует начинать обследование с выделения гласных звуков, стоящих в начале слова под ударением, и согласных звуков, стоящих в конце слова, переходя постепенно к более сложным для детей позициям звуков [8].

Набор упражнений для диагностики и развития фонематического анализа [9]:

* определить место звука в слове – сначала гласного, потом согласного;
* выделить согласный звук из стечения двух звуков;
* определить количество гласных и согласных звуков в слове;
* определить из скольких звуков состоит слово;
* выделить все одинаковые звуки в двух названных словах;
* определить место звука в слове;
* определить последовательность звуков в слове.

Фонематические представления - это сохранившиеся в сознании образы звуковых оболочек слов, которые образовались на основе предшествовавших восприятий этих слов.

Для полноценного овладения чтением и письмом недостаточно умения дифференцировать звуки речи только в момент их непосредственного восприятия на слух. Необходимо, чтобы ребенок сохранял в своей памяти устойчивое представление о каждом звуке, то есть мог мысленно представить себе звучание любого звука. Во время самостоятельного чтения и письма он может соотнести каждый звук с соответствующей ему буквой только на основе таких мысленных представлений о звуках — ведь в это время он не имеет возможности воспринимать эти звуки на слух.

Недоразвитие фонематических представлений приводит к трудностям в овладении процессом чтения и письма, специфическим стойким ошибкам при чтении и письме, и, следовательно, к неуспеваемости в школе, ряду поведенческих и психологических проблем.

Набор упражнений для диагностики и развития фонематического анализа:

* подобрать слова определенной тематики на заданный звук;
* подобрать слово на заданный звук;
* подобрать слово на заданный звук с указанием его места в слове;
* подобрать слово, которое начинается на тот же звук, что и заданное.

Для обследования словаря подходит прием называния предметов, действий и качеств, умение подбирать однородные слова, обобщать понятия. Упражнения разработаны в соответствии с методикой, предлагаемой С.Г. Шевченко.

Набор упражнений для обследования словаря:

* имена существительные:
  + назвать части лица и части туловища;
  + назвать поочередно все однородные предметы каждой группы и подобрать к ним обобщающее слово;
  + назвать малознакомые предметы;
  + назвать предмет по его описанию;
  + назвать по изображениям слова, относящиеся к одной предметной области;
* глаголы:
  + назвать характерное действие по предъявленному предмету;
  + назвать близкие по значению слова-действия;
  + назвать характерное для предмета действие по предъявленному предмету;
* имена прилагательные:
  + подобрать к картинке слово с противоположным значением;
* качественные прилагательные:
  + назвать, какой формы предмет;
  + назвать, какого цвета предметы;
  + назвать оттенки;
  + назвать, какие на вкус нарисованные фрукты, овощи;
* относительные прилагательные:
  + определить из какого материала сделан предмет на изображении;
  + назвать прилагательное по его назначению;
  + назвать притяжательные прилагательные;
* наречия:
  + назвать качественные наречия;
  + назвать наречия времени;
* имена числительные:
  + назвать количественные числительные;
  + назвать порядковые числительные по изображению;
* местоимения:
  + назвать местоимения единственного и множественного числа.

Грамматический строй языка представляет собой комплексную систему, объединяющую морфемику, морфологию и синтаксис.

Морфемика - раздел науки о языке, в котором изучаются значимые части слова и способы словообразования.

Морфология - раздел грамматики, в котором изучаются слова как части речи. Она изучает общие значения и изменения слов, их формы.

Синтаксис - изучает законы построения предложений и словосочетаний как грамматических конструкций.

Набор упражнений для обследования грамматического строя речи:

* Понимание обращенной речи:
  + Понимание предлогов;
  + Понимание предложений с причинно-следственной связью;
  + Понимание изменения числа существительного в предложении;
* Словоизменение:
  + Изменить существительные по падежам;
  + Изменить прилагательное по родам;
  + Подобрать глагол к существительному;
  + Изменить существительные женского, среднего, мужского родов по числам;
  + Изменение существительных с непродуктивными основами по числам;
  + Назвать множественное число существительных в родительном падеже;
  + Согласование существительных с числительными;
  + Изменение глаголов по временам;
* Словообразование:
  + Образовать глаголы с противоположным значением с помощью приставки;
  + Образовать существительные с помощью уменьшительно-ласкательного суффикса;
  + Образовать слова, обозначающие профессию, от существительного с помощью суффиксов;
  + Образовать прилагательные от существительных;
  + Образовать прилагательные, обозначающие неполноту действия;
  + Образовать глаголы от прилагательных.

Оба освещенных программно-методических комплекса нацелены на одну аппаратную платформу, что не отвечает современным требованиям к программному обеспечению. Более подробно этот вопрос был рассмотрен в первой главе данной выпускной квалификационной работы.

Были изучены упражнения для обследования различных областей речевого развития и выбраны те из них, что лучше всего поддаются реализации в наборе стандартных механик, которые предоставляет модель однопользовательского приложения.

# ГЛАВА 3. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

## 3.1 Постановка задачи

В рамках выполнения выпускной квалификационной работы необходимо реализовать кроссплатформенное приложение с использованием технологий Unity.

Приложение должно соответствовать следующим требованиям:

* Разрабатываемый комплекс упражнений должен представлять собой локальную программу, работа с которой возможна без использования подключения к глобальной сети Интернет;
* Разрабатываемый комплекс упражнений должен работать под управлением операционных систем Windows, Unix, Android, Mac OS, iOS;
* в качестве базового программного обеспечения должен быть использован движок Unity;
* среда разработки приложения – MonoDevelop, Microsoft Visual Studio [10];
* язык программирования – C# [15];
* полученное приложение должно быть протестировано и отлажено.

## 3.2 Разработка программного продукта

### 3.2.1 Проектирование приложения

В основе концепции Unity лежит объектно-компонентный подход. Иначе говоря, каждая сущность на сцене может быть представлена как объект, имеющий определенные управляющие компоненты – они контролируют логику его работы и определяют прочие свойства, такие как, например, внешний вид и местоположение в пространстве относительно системы координат [11].

Каждое упражнение представляет собой отдельную Unity-сцену, которые собраны в единую систему через отдельный объект меню, в которое можно выйти в любой момент.

Была разработана следующая структура приложения:

* фонематика:
  + фонематическое восприятие;
  + фонематический анализ;
  + фонематический синтез;
  + фонематическое представление;
* грамматика:
  + понимание предлогов;
* обследование словаря:
  + существительные;
  + прилагательные;
  + числительные.

Рассмотрим подробнее каждый из разделов.

В разделе «Фонематика» реализованы следующие упражнения:

* определить наличие заданного звука в слове;
* определить позицию звука в слове;
* определить количество гласных и согласных звуков;
* определить количество звуков в слове;
* составить слово из звуков, данных в нарушенном порядке;
* подобрать слово, начинающееся на заданный звук.

Упражнение «Определить наличие звука в слове».

Пользователь прослушивает запись слова и видит на экране звук в символьной записи, после чего должен вынести решение – присутствует ли этот звук в озвученном слове.

Управляющие элементы задания:

* Text – для отображения звука в символьном виде;
* Button – для выбора одного из двух вариантов ответа и прослушивания записи;
* AudioSource – для воспроизведения записанного слова.

Макет интерфейса упражнения представлен на рисунке 5.

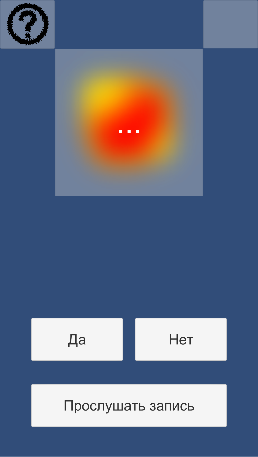


Рисунок 5 – Макет для упражнения «Определить наличие звука в слове»

Упражнение «Определить позицию звука в слове».

Пользователь прослушивает запись слова и видит на экране звук в символьной записи, после чего должен вынести решение – в какой части прослушанного слова этот звук расположен.

Управляющие элементы задания:

* Text – для отображения звука в символьном виде;
* Button – для выбора одного из трех вариантов ответа и прослушивания записи;
* AudioSource – для воспроизведения записанного слова.

Макет интерфейса упражнения представлен на рисунке 6.

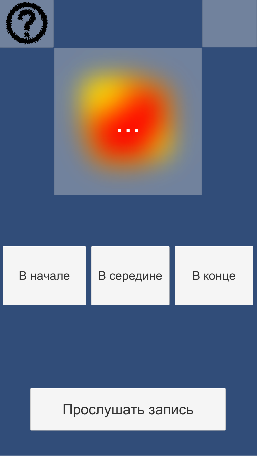


Рисунок 6 – Макет для упражнения «Определить позицию звука в слове»

Упражнение «Определить количество гласных и согласных звуков».

Пользователь прослушивает запись слова, после чего должен вынести решение – сколько в прослушанном слове гласных и сколько согласных звуков.

Управляющие элементы задания:

* Button – для прослушивания записи и подтверждения ответа;
* InputField – два поля для ввода ответа;
* AudioSource – для воспроизведения записанного слова.

Макет интерфейса упражнения представлен на рисунке 7.



Рисунок 7 – Макет для упражнения «Определить количество гласных и согласных звуков в слове»

Упражнение «Определить количество звуков в слове».

Пользователь прослушивает запись слова, после чего должен вынести решение – сколько всего в прослушанном слове звуков. Суть упражнения заключается в том, что пользователь должен не только подсчитывать звуки, но и не запутаться в понятиях «буква» и «звук».

Управляющие элементы задания:

* Button – для прослушивания записи и подтверждения ответа;
* InputField – для ввода ответа;
* AudioSource – для воспроизведения записанного слова.

Макет интерфейса упражнения представлен на рисунке 8.

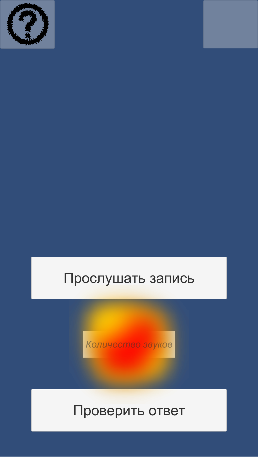


Рисунок 8 – Макет для упражнения «Определить количество звуков в слове»

Упражнение «Составить слово из звуков, данных в нарушенном порядке».

Пользователь видит перед собой четыре карточки с символьным представлением звуков. Карточки перемешаны в случайном порядке. Задача пользователя – расположить карточки в правильном порядке, то есть, таким образом, чтобы символы на карточках образовали существующее слово.

Управляющие элементы задания:

* Button – для подтверждения ответа;
* Script – для подключения разработанной системы компонентов, реализующих механику перетаскивания объектов и их объединения;
* Text – для визуализации карточек.

Макет интерфейса упражнения представлен на рисунке 9.

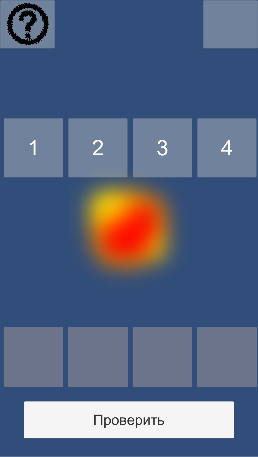


Рисунок 9 – Макет для упражнения «Составить слово из звуков, данных в нарушенном порядке»

Упражнение «Подобрать слово, начинающееся на заданный звук».

Пользователь прослушивает запись слова и видит перед собой шесть карточек с написанными на них словами. Пользователь должен вынести решение – какие из карточек содержат слова, начинающиеся на тот же звук, что и прослушанное слово.

Управляющие элементы задания:

* Button – для прослушивания записи и подтверждения ответа;
* Toggle – для выбора карточек;
* Text – для визуализации карточек.

Макет интерфейса упражнения представлен на рисунке 10.

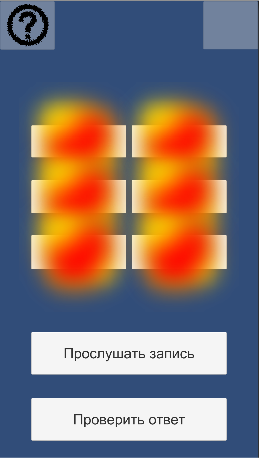


Рисунок 10 – Макет для упражнения «Подобрать слово, начинающееся на заданный звук»

В разделе «Обследование словаря» реализованы следующие упражнения:

* определить части лица и тела;
* назвать однородные предметы и подобрать обобщающее слово;
* определить объект по его описанию;
* определить цвет по его названию;
* составить слово в нужном числе и падеже;
* порядковые числительные.

Упражнение «Определить части лица и тела».

Пользователь видит изображение человеческого тела, некоторые части которого выделены. Также, есть набор карточек, на которых в случайном порядке написаны наименования выделенных частей тела. Задача пользователя – соотнести выделенные части тела и карточки с соответствующим наименованием.

Управляющие элементы задания:

* Button – для подтверждения ответа;
* Script – для подключения разработанной системы компонентов, реализующих механику соотнесения объектов;
* Text – для визуализации карточек;
* Image – для визуализации тела.

Макет интерфейса упражнения представлен на рисунке 11.

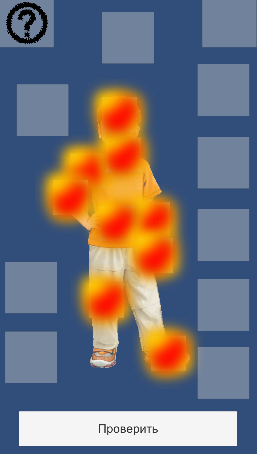


Рисунок 11 – Макет для упражнения «Определить часть тела»

Упражнение «Назвать однородные предметы и подобрать обобщающее слово».

Пользователь видит перед собой четыре изображения и четыре карточки, на котором в случайном порядке написаны соответствующие им наименования. Кроме того, ниже расположены карточки, на которых написаны четыре варианта обобщающего слова для сущностей выше. Верным является только одно. Задача пользователя – соотнести каждое изображение с карточкой наименования и выбрать для всех изображений одно обобщающее слово.

Управляющие элементы задания:

* Button – для подтверждения ответа;
* Script – для подключения разработанной системы компонентов, реализующих механику соотнесения объектов;
* Text – для визуализации карточек;
* Image – для визуализации изображений;
* Toggle – для выбора обобщающего слова.

Макет интерфейса упражнения представлен на рисунке 12.

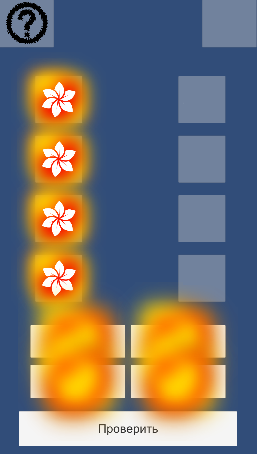


Рисунок 12 – Макет для упражнения «Назвать однородные предметы и подобрать обобщающее слово»

Упражнение «Определить объект по его описанию».

Пользователь видит перед собой текст вопроса, описывающий некий объект и шесть карточек с вариантами ответа. Задача пользователя – выбрать одну из них.

Управляющие элементы задания:

* Button – для выбора ответа;
* Text – для отображения вопроса.

Макет интерфейса упражнения представлен на рисунке 13.

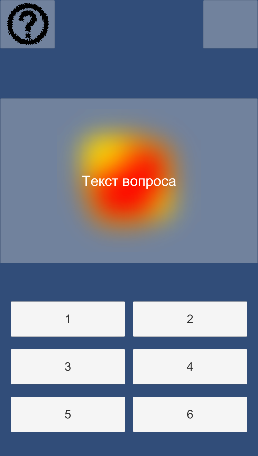


Рисунок 13 – Макет для упражнения «Определить объект по его описанию»

Упражнение «Определить цвет по его названию».

Пользователь видит название цвета и четыре карточки, окрашенные в разные цвета. Задача пользователя – определить какая карточка окрашена в цвет, название которого фигурирует в вопросе.

Управляющие элементы задания:

* Button – для выбора ответа;
* Image – для визуализации цвета;
* Text – для отображения вопроса.

Макет интерфейса упражнения представлен на рисунке 14.

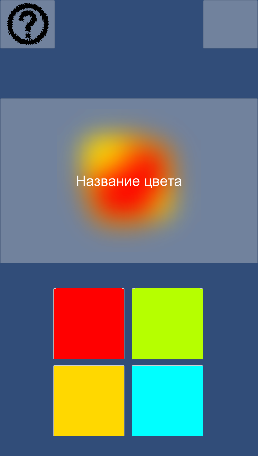


Рисунок 14 – Макет для упражнения «Определить цвет по названию»

Упражнение «Составить слово в нужном числе и падеже».

Пользователь видит текст задания и расположенную ниже панель, состоящую из нескольких объектов. Центральная часть является неизменяемой, левая и правая представляют собой набор карточек, которые можно проматывать. Задача пользователя – оперируя левой и правой частями, составить слово, являющееся решением задания.

Управляющие элементы задания:

* Button – для подтверждения ответа;
* Text – для отображения вопроса и визуализации карточек;
* Scroll View – для организации прокрутки карточек.

Макет интерфейса упражнения представлен на рисунке 15.

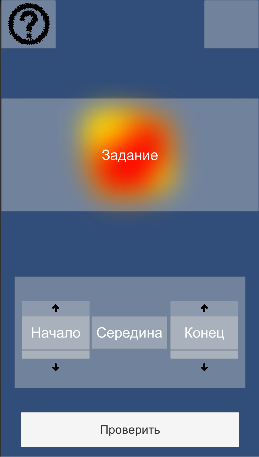


Рисунок 15 – Макет для упражнения «Составить слово в нужном числе и падеже»

Упражнение «Порядковые числительные».

Пользователь видит перед собой изображение, действующие лица которого выделены, и набор карточек, на которых написаны порядковые числительные. Задача пользователя – соотнести карточки с действующими лицами.

Управляющие элементы задания:

* Button – для подтверждения ответа;
* Script – для подключения разработанной системы компонентов, реализующих механику перетаскивания объектов и их объединения;
* Text – для визуализации карточек.

Макет интерфейса упражнения представлен на рисунке 16.

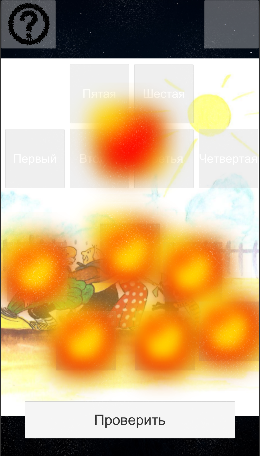


Рисунок 16 – Макет для упражнения «Порядковые числительные»

В разделе «Грамматика» реализованы следующие упражнения:

* понимание предлогов.

Упражнение «Понимание предлогов».

Пользователь видит несколько сцен и набор карточек, которые описывают эти сцены. Задача пользователя – соотнести сцену и соответствующее ей описание.

Управляющие элементы задания:

* Button – для подтверждения ответа и перехода к следующему заданию;
* Script – для подключения разработанной системы компонентов, реализующих механику соотнесения объектов;
* Text – для визуализации карточек.

Макет интерфейса упражнения представлен на рисунке 17.



Рисунок 17 – Макет для упражнения «Понимание предлогов»

Рассмотрим устройство каждой такой сцены.

Помимо основной части, в каждом задании должен быть экран «Помощь», в котором подробно изложено задание и, в случае необходимости, приведены примеры.

Что касается основной части, то она представляет собой набор управляющих элементов – объектов с набором определенных компонентов. Как правило, общую логику упражнения содержит один объект, имеющий скриптовый компонент. На момент старта сцены у него уже есть вся необходимая информация о других объектах и том, как они должны взаимодействовать.

К таким управляющим элементам относятся:

* Button – компонент «Кнопка». Из событий, которые обрабатывает компонент нам чаще всего будет необходим обработчик нажатия;
* Image – компонент «Изображение». Используется для отображения двухмерной графики;
* Script – компонент «Скрипт». Программный код на языке C#. Как правило, класс, реализуемый в этом компоненте, наследуется от класса MonoBehaviour;
* Toggle – компонент «Переключатель». Может быть в двух состояниях – активном и неактивном;
* LineRenderer – компонент «Визуализатор линий». Используется при соотнесении объектов для отслеживания связи;
* AudioSource – компонент «Источник звука». Используется для воспроизведения звуковых файлов в тех заданиях, где это необходимо;
* Animator – компонент «Контроллер анимации». Используется для централизованного управления анимацией объектов. Предоставляет слои, машину состояний и тонкую настройку для пользователя;
* Text – компонент «Текст». Предназначен для отображения символьной информации.

Элементы управления расположены по одному принципу на всех сценах. Кроме того, во всех упражнениях есть постоянные элементы, такие как кнопка «Помощь» и панель для отображения счёта.

### 3.2.2 Создание приложения

В каждой сцене, то есть каждом упражнении, есть один объект с компонентом «Script» - как правило в его названии фигурирует фрагмент «Init».

При помощи инспектора Unity инициализируем ссылки в скрипте на игровые объекты.

Чтобы сделать это, достаточно назначить полю класса, унаследованному от MonoBehaviour, модификатор видимости public. После этого в инспекторе можно с помощью технологии «Drag and Drop» перетащить на соответствующее поле нужный объект. Именно таким образом инициализировано большинство объектов в сценах. Процесс представлен на рисунке 18.

Код файла PPS\_Controller.cs

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class PPS\_Controller : MonoBehaviour

{

public GameObject[] Blocks;

public GameObject[] Places;

public Button CheckBtn;

public Animator CorrectAnimator;

public Animator WrongAnimator;

public Text Stat;

public GameObject MainScreen;

public GameObject HelpScreen;

public Button HS\_BackBtn;

public Button BtnHelp;

public Button HS\_ExitBtn;

…

}

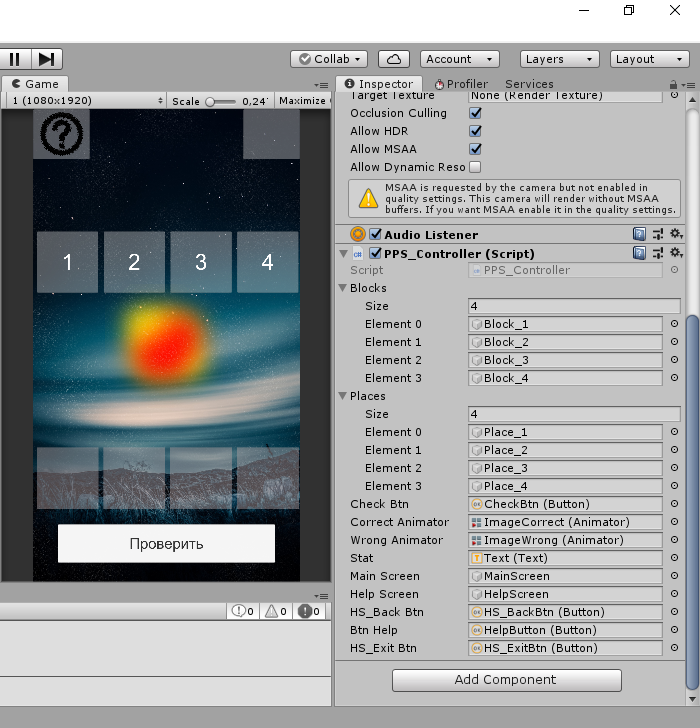


Рисунок 18 – Инициализация полей класса из инспектора объектов

Некоторые упражнения требуют внешних данных. В этом случае, загрузим их во время выполнения метода Awake нашего экземпляра класса MonoBehaviour.

Несмотря на то, что было бы удобно использовать один из готовых форматов хранения данных, XML или JSON, их применение могло бы несколько усложнить процесс кроссплатформенной разработки. Например, в случае с Android потребовалось бы скомпилировать Java-библиотеку, реализующую интересующий нас функционал, и определенным способом встроить ее в проект Unity. В целях упрощения разработки было решено использовать свой формат данных, базирующийся на текстовых файлах и разделителях.

Для использования внешних файлов в проекте Unity есть папка с названием StreamingAssets. В процессе компилирования эти файлы не будут затронуты, и вы можете получить к ним доступ во время работы приложения в их неизменном виде.

Работа с файлами на целевых устройствах может осуществляться посредством подключения библиотеки Java (в случае с Android), написания обертки над ней и использования этой обертки в Unity.

Однако есть способ проще – вместо этого можно использовать функционал Web-запросов, только работать мы будем локально. Это позволит избежать подключения лишних библиотек.

Следующий код позволяет загрузить внешний файл во время работы приложения:

public static string loadFileContent(string path)

{

string res = System.String.Empty;

WWW reader = new WWW(path);

while (!reader.isDone) {}

if (reader.error != null){

if (reader.error == "Unknown Error"){

Debug.LogError("Не удается прочитать файл [" + path + "]");

}

}

res = reader.text;

reader.Dispose();

return res;

}

Теперь, когда внешние данные загружены, организуем цикл упражнения. Когда задания закончатся, приложение перемешает их и начнет выдавать заново.

Что касается элементов управления, то в каждом упражнении они свои. Однако, так или иначе, есть оценивающий метод, в котором происходит сравнение ответа пользователя и известного правильного ответа. После чего приложение продолжает свой цикл.

Оценивающий метод в задании «Составить слово в нужном падеже и числе»:

void FinalButtonClicked(){

int start\_ind = GetStartFocusedElementIndex();

int end\_ind = GetEndFocusedElementIndex();

string start = StartScroll.content.transform.GetChild(start\_ind).GetChild(0).GetComponent<Text>().text;

string end = EndScroll.content.transform.GetChild(end\_ind).GetChild(0).GetComponent<Text>().text;

string result = start + Questions[Index].Middle + end;

if (result.Length > Questions[Index].CorrectAnswer.Length){

result = result.Substring(0,Questions[Index].CorrectAnswer.Length);

}

if (String.CompareOrdinal(result.ToUpper(),Questions[Index].CorrectAnswer.ToUpper()) == 0){

CorrectAnimator.Play("Unhide");

CorrectAttempts += 1;

Next();

} else {

WrongAnimator.Play("Unhide");

}

Attempts += 1;

Stat.text = CorrectAttempts.ToString() + "\\" + Attempts.ToString();

}

Главное меню реализовано с помощью механики, позволяющей сохранять объект сцены, несмотря на то, что сцена уничтожена и паттерна Singleton [12].

public class MM\_Controller : MonoBehaviour

{

static MM\_Controller Instance = null;

…

void Awake(){

if (Instance == null){

Instance = this;

DontDestroyOnLoad(gameObject);

} else {

Destroy(gameObject);

}

}

…

}

## 3.3 Тестирование приложения

В данном разделе описывается тестирование полученного приложения.

После запуска приложения на экране появляется главное меню, изображенное на рисунке 19.

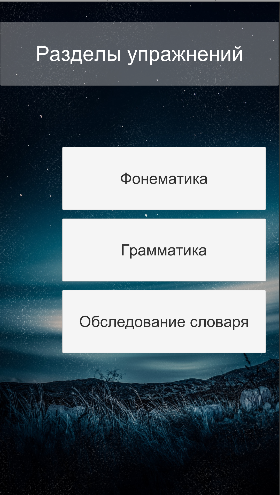


Рисунок 19 – Главное меню приложения

Для навигации по разделам использованы компоненты Button. Для возвращения на предыдущую страницу может быть использована кнопка «Назад» Android-устройства или появляющийся, когда пользователь покидает основную страницу, GameObject с компонентом Button.

Выбрав при помощи меню одно из упражнений, пользователь увидит экран загрузки – сцены загружаются не мгновенно, поэтому имело смысл сделать загрузку асинхронной и во время ее выполнения демонстрировать пользователю ее прогресс. Визуализация реализована при помощи постепенного, согласованного с прогрессом загрузки, изменения свойства Fill Amount компонента Image. Этот процесс изображен на рисунке 20.



Рисунок 20 – Экран загрузки упражнения

В качестве примера подробно разберем упражнение «Назвать однородные предметы и подобрать обобщающее слово». После загрузки мы видим сцену, находящуюся в стартовом состоянии. Это изображено на рисунке 21.

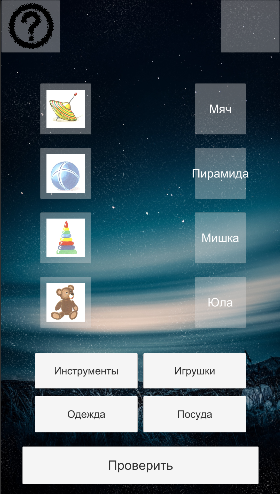


Рисунок 21 – Стартовое состояние сцены в упражнении «Назвать однородные предметы и подобрать обобщающее слово»

Если у пользователя возникают вопросы по заданию, то он может активировать экран справки с помощью кнопки, находящейся в левом верхнем углу экрана – для текущего задания он выглядит так как указано на рисунке 22.

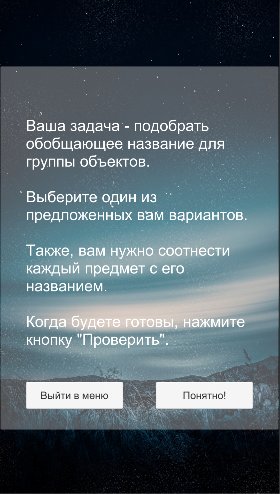


Рисунок 22 – Экран помощи в упражнении «Назвать однородные предметы и подобрать обобщающее слово»

Если пользователь не выполнил какое-либо требование, указанное в задании, то после нажатия кнопки «Проверить», приложение выделит оранжевым цветом элементы, с которыми пользователь еще не взаимодействовал. Это можно увидеть на рисунке 23.

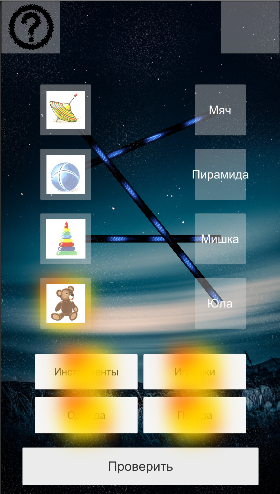


Рисунок 23 – Подсветка элементов, с которыми пользователь не взаимодействовал

Когда пользователь приведет в активное состояние все элементы, которые описываются заданием, приложение проверить правильность его ответа. Если пользователь допустил ошибку, приложение укажет конкретное место ошибки – вышеописанная ситуация иллюстрируется рисунком 24.

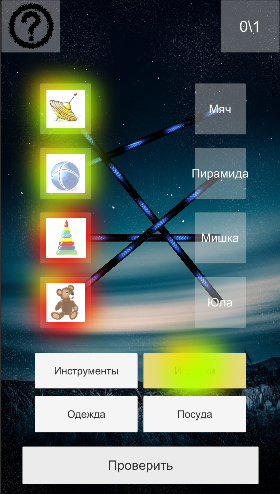


Рисунок 24 – Подсветка верных и неверных частей ответа пользователя

### Выводы

В рамках данной главы приведена постановка задачи - необходимо реализовать кроссплатформенное приложение с использованием технологий Unity на языке высокого уровня C#.

Разрабатываемый комплекс упражнений должен представлять собой локальную программу, работа с которой возможна без использования подключения к глобальной сети Интернет. Программа должна работать под управлением операционных систем Windows, Unix, Android, Mac OS, iOS.

Были приведены структура приложения и этап проектирования упражнений.

Отдельное внимание уделено процессам разработки и тестирования, которые описывают и демонстрируют функции полученного приложения.

# ЗАКЛЮЧЕНИЕ

В рамках выполнения данной выпускной квалификационной работы была рассмотрена тема «Разработка кроссплатформенного приложения с использованием технологии Unity для диагностики уровня речевого развития детей дошкольного возраста».

Актуальность рассматриваемой темы объясняется тем что использование компьютерных программ и мобильных приложений может служить одним из средств оптимизации процесса коррекции речи – как для повышения мотивации, так и для, непосредственно, реализации упражнений и коррекционных задач, которые невозможно задать ребенку без использования современных технологий.

Первая глава работы носит теоретический характер. В ней приводится понятие программного обеспечения и требования, предъявляемые к программному продукту, в частности, кроссплатформенность.

Отдельное внимание уделяется жизненному циклу программного обеспечения. Жизненный цикл представляет собой некоторую абстрактную модель создания и последующего применения программного продукта, отражающую все его возможные состояния. В рамках первой главы описаны наиболее популярные модели жизненного цикла:

* каскадная;
* спиральная;
* инкрементная.

Кроме того, приводятся краткая сравнительная характеристика самых популярных движков для решения поставленных задач:

* Unity 5;
* Unreal Engine 4;
* Cry Engine 3.

Вторая глава работы – аналитическая. Он включает в себя обзор существующих аналогов программного решения, которое было разработано в ходе выполнения данной работы.

Среди представителей интересующей нас отрасли были выделены:

* программно-дидактический комплекс «Мерсибо Интерактив»;
* программно-методический комплекс «Игры для Тигры».

Были освещены их характерные особенности и аппаратно-программные требования для работы с ними.

Также, были рассмотрены коррекционные упражнения для обследования различных областей речевого развития и выбраны те из них, что лучше всего поддаются реализации в наборе стандартных механик, которые предоставляет модель однопользовательского приложения.

Третья глава – практическая. В рамках данной главы описывается процесс проектирования и разработки приложения. Движок Unity идеально подошел для реализации каждого упражнения.

Отдельное внимание уделено тестированию приложения, в процессе которого демонстрируются его функциональные особенности.

Дальнейшая деятельность в направлении развития функционала приложения может быть связана с расширением числа и вида упражнений, что позволит повысить охват тем и, соответственно, практическую пользу приложения.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Грибова, О.Е. Становление текстовой компетенции у подростков с общим недоразвитием речи / О.Е. Грибова. - Москва: Изд-во Ленанд, 2014. - 320 c.
2. Зараменских Е.П. Управление жизненным циклом информационных систем. – Новосибирск: Изд-во ЦРНС, 2014. – 270 с.
3. Кумагина Е.А. Модели жизненного цикла и технологии проектирования программного обеспечения / Е.А. Кумагина, Е.А. Неймарк. – Нижний Новгород: Изд-во ННГУ, 2016. – 41 с.
4. Голицына, О.Л. Программное обеспечение /О.Л. Голицына, И.И. Попов, Т.Л. Партыка. — 2-е изд., перераб. и доп. — Москва: Изд-во Форум, 2008. — 448с.
5. Партыка, Т.Л. Среды и оболочки /Т.Л. Партыка, И.И. Попов. — Москва: Изд-во Форум, 2003. — 400с.
6. Крейтон, Р.Х. Unity Game Development Essentials / Р.Х. Крейтон. — Packt Publishing, 2010, 83 c.
7. Павловская, Т.А. C#. Программирование на языке высокого уровня / Т.А. Павловская - Спб; ПИТЕР, 2009 432 с.
8. Спирова, Л.Ф. Особенности речевого развития детей с тяжелыми нарушениями речи /Л.Ф. Спирова. — М: Изд-во Педагогика, 1980. — 192с.
9. Иншакова, О.Б. Альбом для логопеда /О.Б. Иншакова. — 2-е изд., испр. и доп. — М: Гуманит. изд. центр. ВЛАДОС, 2008. — 279с.
10. Культин, Н.Б. Microsoft Visual C# в задачах и примерах / Культин Н.Б. CПб: Изд-во БХВ-Петербург, 2009, 320 с.
11. Сью Блэкман, Beginning 3D Game Development with Unity/ Cью Блэкман; Apress, 2011, 992 c.
12. Тепляков С., Паттерны проектирования на платформе .NET /Тепляков С. — СПб: Питер, 2015. — 320с.
13. Уильям Шериф, Learning C++ by Creating Games with UE4 / Уильям Шериф — Packt Publishing, 2015. — 342с.
14. Дэн Трэйси, CryENGINE 3 Cookbook /Дэн Трэйси, Шон Трэйси. — Packt Publishing, 2011. — 324с.
15. Шилдт, Г. C# Полное руководство / Г. Шилдт И.В. Берштейн Москва., Вильямс, 2011. - 1056 c.