Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Высшей математики, информатики, экономики и естествознания
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Л.Н. Храмова
подпись инициалы, фамилия
«13» 06 2023 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии
код-наименование направления

РАЗРАБОТКА РАЗВИВАЮЩЕЙ КОМПЬЮТЕРНОЙ ИГРЫ НА ПЛАТФОРМЕ UNITY

Руководитель	13.06.23 подпись, дата	доцент, канд. пед. наук должность, ученая степень	А. В. Фирер инициалы, фамилия
Выпускник	13.06.23 подпись, дата		Н. В. Шорохов инициалы, фамилия
Нормоконтролер	13.06.2023 подпись, дата		Е. В. Киргизова инициалы, фамилия

Лесосибирск 2023

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка развивающей компьютерной игры с использованием платформы Unity» содержит 65 страниц текстового документа, 41 использованный источник, 34 рисунка, 4 таблицы, 3 приложения.

РАЗВИВАЮЩИЕ ИГРЫ, КОМПЬЮТЕРНАЯ РАЗВИВАЮЩАЯ ИГРА, UNITY, РАЗРАБОТКА ИГРЫ.

Актуальность данной темы обусловлена тем, что в научной и технической литературе недостаточно раскрыт вопрос о проектировании и разработке развивающих компьютерных игр на платформе Unity.

Цель исследования – теоретически обосновать и разработать развивающую компьютерную игру для детей 6–7 лет на платформе Unity.

Объект исследования – процесс разработки компьютерных игр.

Предмет исследования – процесс разработки развивающих компьютерных игр для детей 6–7 лет на платформе Unity.

Основные задачи исследования:

– на основе анализа учебной, научно-технической литературы и предметной области сформулировать требования к разрабатываемой развивающей игре;

– на основе сравнительного анализа обосновать выбор инструментальных средств разработки игры;

– спроектировать структуру развивающей компьютерной игры и сценарий игрового процесса;

– разработать и протестировать компьютерную развивающую игру.

В ходе исследования теоретически обоснована, спроектирована, разработана развивающая компьютерная игра на платформе Unity «Познай себя». Функциональное тестирование игры подтвердило качество разработанной игры.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Мышление ребёнка – одна из главных и важных характеристик его психики. Это процессы, за которые отвечает самый сложный орган нашего тела – мозг. Игра – это деятельность познавательная, она представляет собой своеобразную практическую форму размышления ребенка об окружающей его природе и социальной действительности. Благодаря особенностям игровых средств отображения действительности, ребенок в игре впервые приобщается к абстрактному мышлению. Умственное развитие – это образование, являющееся центральной частью общего психического развития ребенка во всей будущей жизни. В свою очередь, умственное развитие – это сложный процесс, направленный на формирование познавательных интересов, знаний и умений.

Компьютерные игры – новый вид развивающего обучения. В чем же польза от этих игр? Компьютерные технологии избавляют как педагога, так и ребенка от тяжелой рутинной работы. Ребенок нуждается в достаточном количестве знаний для дальнейшего формирования ума. В наше время современные дети больше склонны получать информацию из компьютера, нежели из книг, поэтому надо идти в ногу со временем.

Одним из способов решения данной задачи является разработка различного рода приложений игр. В игре вырабатываются такие жизненно важные качества, как внимательность, усидчивость, память, упорство, настойчивость в достижении цели. Таким образом, использование компьютерной программы повышает мотивацию не только за счет игровой стратегии, на которой программа базируется, но и потому, что ребенок получает одобрение, похвалу не только со стороны взрослых, но и со стороны компьютера.

Компьютерные игры, как нетрадиционный метод обучения может существовать наравне с традиционными методами преподавания. Задача преподавателя разъяснить, что эффективность от компьютерной игры будет

только в том случае, если ребёнок проводит ограниченное время за компьютером.

Индустрия разработки компьютерных игр – это новое направление и малоизученное. Применение цифровых технологий и распространение компьютерных игр среди учеников и студентов предполагает поиск новых методов с их применением и оптимизацию процесса обучения. Компьютерные игры способствуют самостоятельному изучению, мотивируют их на совершенствование навыков и коммуникативного общения.

Вопросу о развивающих играх, их структуре и функциях уделяется внимание в работах таких исследователей как Т. В. Рябова [20], А. А. Попов [19], Е. В. Горшков [19], А. З. Асроров [19], Е. С. Сергеев [21], А. Е. Сухова [21], И. С. Максимов [21], Н. А. Сенаторов [21] и др.

Использование платформы Unity в процессе разработки игр раскрыто в работах Р. Г. Бонда [4] и А.Торна [22].

Однако, вопрос о проектировании, структуре и функциях разработки развивающих игр для детей 6–7 лет на Unity раскрыт недостаточно полно, что свидетельствует об актуальности проведенного исследования.

В ходе выпускной работы будет создана развивающая компьютерная инди-игра. Инди-игра – игра, созданная одним человеком или небольшой группой людей. Её развитие происходит за счет средств её разработчиков.

Цель исследования – теоретически обосновать и разработать развивающую компьютерную игру для детей 6–7 лет на платформе Unity.

Объект исследования – процесс разработки компьютерных игр.

Предмет исследования – процесс разработки развивающих компьютерных игр для детей 6–7 лет на платформе Unity.

Для достижения поставленной цели необходимо выполнить следующие задачи:

– на основе анализа учебной, научно-технической литературы и предметной области сформулировать требования к разрабатываемой развивающей игре;

– на основе сравнительного анализа обосновать выбор инструментальных средств разработки игры;

– спроектировать структуру развивающей компьютерной игры и сценарий игрового процесса;

– разработать и протестировать компьютерную развивающую игру.

Методы исследования:

– теоретические: анализ учебной и научно-технической литературы по теме исследования; обобщение; сравнительный анализ;

– эмпирические: опрос; моделирование; тестирование программного продукта.

Практическая значимость исследования заключается в том, что данное приложение может быть использовано педагогами для проведения внеклассных мероприятий, помимо этого, полученный материал исследования может использоваться студентами при написании статей, рефератов, курсовых и дипломных работ.

Результаты исследования представлены на следующих научных мероприятиях:

1. VI Всероссийская научно-практическая конференция преподавателей, учителей, студентов и молодых ученых «Актуальные проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 7–12 ноября 2022 г., участие).

2. Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 11 апреля 2023 г., участие).

По результатам исследования опубликованы статьи:

1. Шорохов, Н. В. Анализ средств разработки компьютерных развивающих игр / Н. В. Шорохов // Актуальные проблемы преподавания дисциплин естественнонаучного цикла: тезисы докладов VI Всероссийской научно-практической конференции преподавателей, учителей, студентов и

молодых ученых, Лесосибирск, 14–15 ноября 2022 года / Сибирский федеральный университет. – Красноярск, 2022. – С. 81–84.

2. Шорохов, Н. В. Особенности разработки развивающих игр для детей младшего школьного возраста / Н. В. Шорохов // Современное педагогическое образование: Теоретический и прикладной аспекты: сборник научных статей II Всероссийский молодёжный научный форум, студентов и молодых ученых, Лесосибирск, 10–15 апреля 2023 года / Сибирский федеральный университет. – Лесосибирск – Красноярск, 2023. – С. 86–90.

Структура работы – работа состоит из введения, двух глав, заключения, списка использованных источников, включающего 41 наименование, и трех приложений. Результаты работы представлены в 4 таблицах, 25 рисунках. Общий объём работы – 65 страниц.

1 Проектирование развивающей компьютерной игры

1.1 Теоретические аспекты проектирования развивающих компьютерных игр

Рассмотрим некоторые теоретические аспекты проектирования развивающих компьютерных игр, которые являются необходимыми при разработке.

1.1.1 Анализ жанров развивающих компьютерных игр

В современном мире развитие детей играет важную роль, и игры могут стать эффективным инструментом в этом процессе. В своей работе С. М. Кокче указывает, что «развивающие игры помогают развивать у детей воспитывать интерес, способность к исследованию и творческому поиску, желание и умение учиться» [11, 29].

Развивающие игры приобретают все большую популярность как средство вовлечения детей в интерактивное обучение. В частности, игры, предназначенные для детей 6–7 лет, обычно играют решающую роль в содействии их когнитивному, моторному и эмоциональному развитию.

Если рассматривать развивающие игры, созданные в России, то это будет список всего из нескольких пунктов. Вдобавок при их подробном рассмотрении становится понятно, что большая часть даже не являются таковыми.

Опираясь на работу Д. В. Денисова [9], выделим различные жанры развивающих игр.

Обучающие игры – этот тип игр помогает игрокам изучать новые навыки, такие как правила грамматики, история, наука и т. д. Такие игры часто представлены в форме квестов или интерактивных историй.

Логические игры – этот жанр игр помогает игрокам развивать логическое мышление и способность к анализу. Эти игры могут включать в себя головоломки, шарады и т. д.

Симуляторы – это жанр игр, в которых игроки могут управлять реалистичными объектами и принимать решения, влияющие на процесс игры. Например, врачи, которые управляют больницами, или менеджеры, которые управляют бизнесом.

Спортивные игры – этот жанр игр помогает игрокам улучшить свои спортивные навыки, такие как координация движений, силовые упражнения, реакция и т. д.

Игры с геймификацией – этот тип игр объединяет различные игровые механики и элементы игрового дизайна, такие как система бонусов, достижений, система уровней и т. д., чтобы помочь игрокам оставаться мотивированными и заинтересованными в игре.

Таким образом, развивающие игры – это жанр компьютерных игр, который призван помочь игрокам развивать навыки и умения в интересной и увлекательной форме. Этот жанр может включать в себя различные поджанры, такие как логические игры, головоломки, обучающие игры, поэтому разработчик может объединить несколько поджанров для создания высококачественной и интерактивной игры с большим количеством функций.

1.1.2 Характеристика целевой аудитории развивающей игры

Целевая аудитория разрабатываемой развивающей игры – дети 6–7 лет. На этом этапе развития дети активно интересуются миром вокруг себя, находятся на стадии активного учения и исследования. Они начинают осваивать различные навыки, такие как чтение, письмо, математика и логика. Дети в этом возрасте могут уже понимать абстрактные понятия и решать простые логические задачи. Они могут использовать воображение и аналитические навыки для решения проблем. Дети в этом возрасте могут концентрироваться на задаче в течение некоторого времени, но их способность сосредоточиться может быть ограничена. Игры должны быть интересными и стимулирующими, чтобы держать их внимание.

Дети 6–7 лет имеют достаточно развитую моторику, что позволяет им легко управлять игровыми персонажами и выполнять простые движения с помощью мыши или сенсорного экрана. Яркие, красочные и привлекательные визуальные эффекты способствуют удержанию внимания детей этого возраста. Игры с простым и интуитивно понятным пользовательским интерфейсом будут легче пониматься и использоваться. Развивающие игры для детей 6–7 лет должны предлагать образовательные задания, которые помогут развить навыки чтения, письма, математики, логики и другие когнитивные способности. Игры для детей этого возраста должны быть безопасными и соответствовать нормам безопасности и конфиденциальности данных. Кроме того, наличие простой и понятной поддержки или подсказок поможет детям в случае затруднений. Учитывая эти характеристики целевой аудитории, разработчики игры могут создавать игровой контент, который соответствует интересам и потребностям детей 6–7 лет, способствуя их развитию и обучению.

1.1.3 Анализ существующих решений среди развивающих компьютерных игр для детей в возрасте 6–7 лет

Для анализа существующих решений был проведён опрос среди родителей, имеющих детей в возрасте от 6 до 8 лет, в котором респонденты указывали возраст ребёнка, пол, а также указывали игры, в которые предпочитают играть их дети. В опросе участвовали 67 респондентов. Результаты опроса приведены на рисунке 1.

Из диаграммы видно, что наибольшей популярностью пользуется игра Minecraft. На втором месте по популярности стоит онлайн проект Logiclike, который содержательно наиболее близок для нашего исследования.

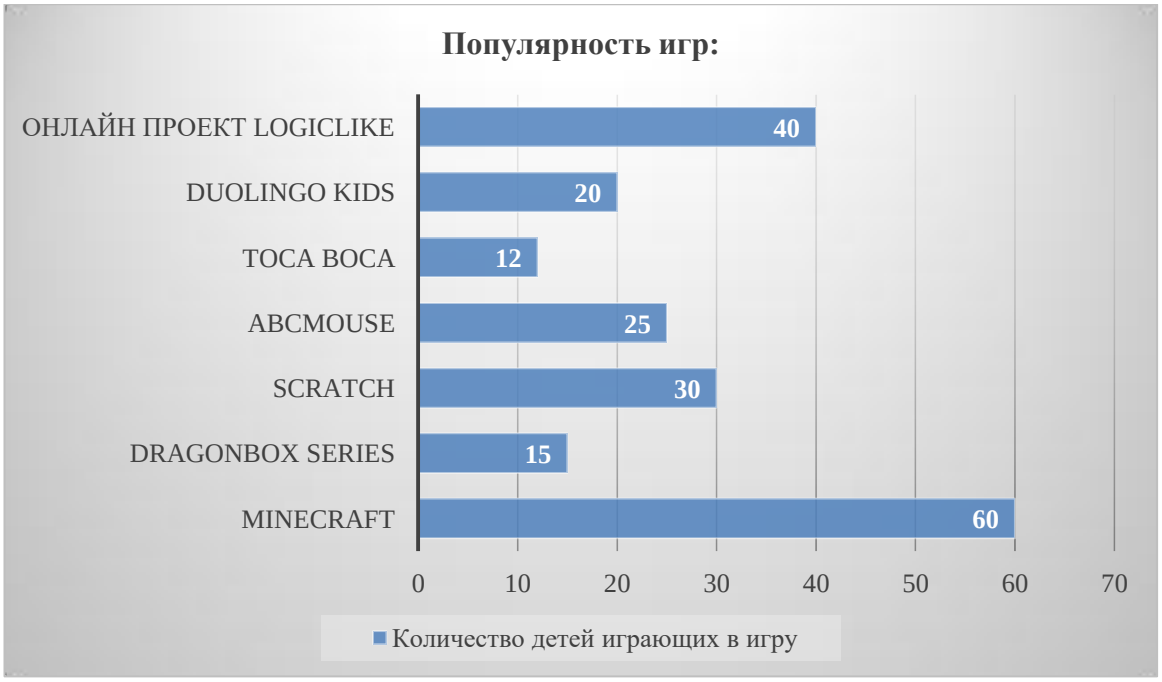

Рисунок 1 – Популярность игр по результатам опроса родителей

Исходя из результатов опроса, в ходе исследования были сформулированы характеристики, по которым проводился сравнительный анализ существующих решений. Результаты анализа отражены в таблице 1.

Таблица 1 – Анализ существующих решений

Характеристика	**Интерактивные составляющие задания**	**Образовательный аспект**	**Визуальные предпочтения**	**Адаптивность**	**Бесплатное решение**
Minecraft	–	–	+	+	–
DragonBox series	–	–	+	+	+
Scratch	+	±	+	±	+
ABCmouse	±	±	±	±	+
Toca Boca	–	±	±	+	+
Duolingo kids	+	+	–	+	–
Онлайн проект Logiclike	+	+	±	+	–

Разрабатываемая в ходе исследования игра должна обладать всеми характеристиками, указанными в таблице 1.

С проведенного анализа и характеристики целевой аудитории выделим требования к разрабатываемой игре.

1.1.4 Требования к развивающей компьютерной игре для детей 6–7 лет

При разработке развивающих игр для детей 6–7 лет важно тщательно подбирать игровую механику, соответствующую этапу их развития. Игровая механика должна быть увлекательной, доставляющей удовольствие и способствовать обучению через игру. Например, игры с решением головоломок, принятием решений и запоминанием могут помочь развить навыки критического мышления, в то время как игры, основанные на двигательных навыках, могут улучшить координацию рук и глаз. Крайне важно соблюдать баланс между вызовом и доступностью, гарантируя, что игры не будут слишком легкими или слишком сложными для целевой возрастной группы.

Следует учитывать уровни сложности игр, чтобы убедиться, что они достаточно сложные, чтобы поддерживать мотивацию детей, но не подавляющие их. Можно использовать постепенное повышение уровней сложности, чтобы создать у детей чувство выполненного долга и побудить их продолжать играть и учиться. Кроме того, игры должны обеспечивать индивидуализацию, позволяя игрокам прогрессировать в своем собственном темпе и адаптироваться к различным уровням мастерства.

Визуальный дизайн развивающих игр для детей 6–7 лет должен быть визуально привлекательным, красочным и соответствовать возрасту. Использование увлекательной графики, анимации и интерактивных элементов может улучшить общий игровой процесс и стимулировать вовлечение детей. Звуковые эффекты, такие как музыка, голос за кадром и звуковые подсказки, также могут способствовать погружению в игры и доставлять удовольствие. Важно учитывать, что дети в возрасте 6–7 лет ещё могут не уметь читать, для этого в игре должно быть предусмотрено озвучивание заданий. Например, при наведении на вариант ответа должно озвучиваться, что именно выбирает ребёнок.

В своих работах Ю. А. Алексеев [1], Д. А. Винокуров [5] указывают, что использование компьютера эффективно помогает в развитии творческих способностей детей. Изученный в процессе игровой деятельности материал забывается учащимися в меньшей степени, чем материал, при изучении которого игра не использовалась. Это объясняется, прежде всего, тем, что в игре органически сочетается занимательность, делающая процесс познания доступным и увлекательным для датей, и деятельность, благодаря участию в которой в процессе обучения, усвоение знаний становится более качественным и прочным. Авторы так же отмечают сложности проектирования разработки развивающей игры, которая требует глубокого понимания педагогических принципов и методов обучения детей. Одна из сложностей состоит в том, чтобы найти правильный баланс между обучением и развлечением. Игра должна быть достаточно интересной и захватывающей, чтобы привлечь внимание детей, но в то же время предоставлять обучающий контент и способствовать развитию навыков.

Развивающие игры должны соответствовать педагогическим принципам, чтобы максимально повысить их эффективность в качестве инструментов обучения и развития. Анализ педагогической литературы позволил выделить наиболее важные педагогические принципы, которые следует учитывать при разработке компьютерных развивающих игр: индивидуализация, мотивация и актуальность. Индивидуализация предполагает адаптацию игр к уникальным учебным потребностям и предпочтениям каждого ребенка, что позволяет осуществлять кастомизацию и персонализацию. Мотивацию можно стимулировать с помощью вознаграждений, стимулов и обратной связи, побуждая детей продолжать играть и учиться. Актуальность предполагает соответствие содержания игр учебной программе или конкретным целям обучения, придание играм осмысленности и применимости к реальным жизненным ситуациям.

Анализ работ Д. В. Денисова [9], Гейг Майка [6], В. Я. Платова [18] и существующих решений позволил выделить следующие функциональные

требования к развивающей игре:

1) Игра должна содержать несколько уровней, каждый из которых будет представлять собой новый уровень сложности для ребенка. Уровни могут быть связаны с тематикой, например такими как, математика, логика и т. д.

2) Игра должна включать обучающий контент, который помогает детям учиться новым вещам. Например, это может быть грамматика, математика, основы науки или пространственного мышления.

3) Игра должна содержать головоломки, которые помогают развивать логическое мышление, улучшать координацию движений, решать задачи, тренировать память и другие навыки.

4) Игра должна быть интерактивной и увлекательной. Дети должны иметь возможность взаимодействовать с игрой и управлять ею, чтобы они могли чувствовать, что контролируют процесс.

5) Игра может содержать систему наград и поощрений, которые мотивируют детей продолжать играть и учиться. Награды могут быть в виде медалей, значков, баллов, достижений и т. д.

6) Игра должна быть визуально привлекательной для детей. Это может означать использование ярких цветов, забавных персонажей, анимации и других элементов.

7) Игра должна быть адаптивной к уровню знаний и возрасту детей. Это означает, что игра должна быть достаточно легкой для маленьких детей, но достаточно сложной для более старших.

8) Игра должна иметь понятный и удобный интерфейс для детей. Он должен быть простым и интуитивно понятным, чтобы дети могли легко понимать, что происходит в игре.

9) В соответствии с рекомендациями Всемирной организации здравоохранения [23], дети в возрасте от 5 до 8 лет должны проводить не более 1 часа в день за экраном в целом, включая время на образовательные или развивающие деятельности. Игра должна подразумевать меню паузы и уведомление с озвучкой, что ребёнок слишком много времени провёл за игрой.

1.2 Анализ инструментальных средств разработки

В качестве среды разработки был выбран платформа Unity. В ходе сравнения она была выбрана, как самая подходящая для нашего проекта. В качестве сравнения были проанализированы следующие среды разработки.

Unity [35]

Одна из самых популярных платформ для создания игр – Unity. Она появилась в 2005 году и до сих пор не теряет спроса среди гейм-разработчиков. Первая причина, почему программный комплекс управления так долго держится на плаву – это огромное сообщество и множество официальных и неофициальных видео уроков, которые помогают создавать игры и решать многие проблемы.

Unreal Engine [39]

Еще один лидер в рейтинге игровых движков. Unreal Engine – это платформа, ориентированная на 3D, с системой визуального программирования. Unreal Engine использует специальный язык сценариев под названием Blueprint, который позволяет создавать визуальный скриптинг без написания кода. Кроме того, Unreal Engine поддерживает C++ для более продвинутого программирования и оптимизации, что делает его лучшим выбором для разработчиков с опытом работы с C++.

CryEngine [30]

CryEngine позволяет создавать игры с фотореалистичной графикой. При должном умении проекты, разработанные с его помощью, превосходят по качеству картинки любые игры. CryEngine известен своими исключительными возможностями в области графики. Он предлагает передовые функции рендеринга, включая реалистичное глобальное освещение в реальном времени, динамические погодные системы и продвинутые частицы. У него сильное внимание к созданию визуально потрясающих и реалистичных окружений.

Godot [32]

Unity и Unreal Engine – большие имена в разработке игр. Они оба

свободны в использовании, но есть также и другие. Например, Godot – бесплатная платформа для разработки игр с открытым исходным кодом. Главным минусом Godot можно назвать малое в сравнении с Unity количество документации и видео уроков, особенно на русском языке.

На сегодняшний день все привыкли к существованию нескольких общепринятых программных решений для создания игр, а именно:

1. Unreal Engine 4;
2. Unity;
3. Godot;
4. CryEngine.

Сравнительный анализ средств разработки представлен в таблице 2.

Таблица 2 – Сравнение сред разработки

Критерии	**Unity**	**Unreal Engine 4**	**Godot**	**CryEngine**
Доступность	Бесплатный	Бесплатный	Проект создаётся бесплатно, но требует денег для реализации	3 088 рублей
Набор инструментов	Большой	Огромный	Большой	Огромный
Порог вхождения	Низкий	Средний	Низкий	Высокий
Сообщество (доступность обучения и видео уроки)	Огромное	Огромное	Низкое	Низкое
Язык программирования	C#	C++	GML	C++, C#, HSL

Например, Unity чаще всего используют для разработки мобильных игр на IOS, Android и Windows, а Unreal Engine 4 для разработки игр для таких платформ, как PlayStation, MacOS и Xbox. Встречается мнение, что легче всего начинать развиваться в сфере GameDev посредством изучения такого способа взаимодействия со средой разработки как Visual Scripting.

Естественно, пользователь, услышав о необходимости писать код, непременно взглянет в сторону чего-то более простого и понятного на первый взгляд. И этот пользователь откроет для себя очень гибкую систему,

позволяющую создавать игры с наименьшими усилиями.

Unity – это универсальный игровая платформа для разработки, предназначенная для всех. С другой стороны, CryEngine хороша для определенных групп людей. Люди, желающие создать игру с модифицированной игровой платформой, могут выбрать CryEngine. Unity – лучший вариант во всех остальных случаях.

Adobe Photoshop [26]

Adobe Photoshop может использоваться для создания и редактирования графических ресурсов для игр, в том числе для Unity. С помощью Photoshop можно создавать и редактировать спрайты, текстуры, интерфейсы и другие элементы, используемые в игровой разработке в Unity.

Photoshop предоставляет широкие возможности для редактирования графических изображений, включая различные инструменты рисования, редактирования цвета и фильтры. Это позволяет создавать высококачественные и детализированные графические элементы, которые могут использоваться в Unity.

Одним из главных преимуществ использования Photoshop в Unity является возможность экспортировать графические ресурсы в форматах, которые поддерживаются Unity, таких как PNG, JPG, TIF, BMP, PSD и GIF. Это делает возможным быстрое и простое импортирование созданных в Photoshop графических ресурсов в Unity.

Gimp [31]

GIMP (GNU Image Manipulation Program) – это бесплатный и открытый графический редактор, который также может быть использован для создания спрайтов для игр, включая для Unity. Он предоставляет множество инструментов и функций для рисования, редактирования и экспорта графических изображений, что делает его удобным инструментом для создания спрайтов в Unity.

CorelDraw [29]

CorelDRAW – это векторный графический редактор, который, в

принципе, можно использовать для создания некоторых элементов интерфейса в Unity, таких как кнопки, иконки, логотипы и т.д. Однако, для создания спрайтов, которые используются для отображения игровых объектов, более подходящим является растровый графический редактор, такой как Adobe Photoshop или GIMP.

Inkscape [33]

Inkscape – это свободно распространяемый векторный графический редактор, который позволяет создавать и редактировать различные векторные изображения, включая спрайты.

Inkscape предлагает различные инструменты и возможности для создания спрайтов, и конкретная методика работы может зависеть от предпочтений и требований проекта.

Сравнение конечных стоимостей лицензий сравниваемых редакторов за минимальный временной период, предоставляемый разработчиком, представлено в таблице 3.

Таблица 3 – Цена использования программного обеспечения

Название	**Лицензия**	**Цена, руб./месяц**
«CorelDraw»	проприетарная	2 823,98
«Adobe Photoshop»	проприетарная	1 620,43
«Inkscape»	свободная	–
«GIMP»	свободная	–

GIMP и Inkscape обладают всем необходимым функционалом создания моделей и анимации для разрабатываемого проекта. Главное преимущество выбранных пакетов – свободная модель распространения. Поэтому для разработки спрайтов игры будем использовать именно эти графические редакторы.

1.3 Моделирование компьютерной развивающей игры

Unified Modeling Language (UML) является широко используемым стандартом для визуализации и описания различных аспектов системы. Он предоставляет набор графических элементов и нотаций для создания диаграмм, которые помогают визуализировать структуру и поведение системы.

Как отмечает Крег Ларман [13], UML диаграммы последовательности являются одним из видов диаграмм, используемых в разработке программного обеспечения для визуализации взаимодействия между объектами или компонентами системы во времени. Они помогают представить последовательность выполнения операций или сообщений между различными элементами системы. Так в диаграмме существуют различные объекты, представляющие сущности или компоненты системы, которые взаимодействуют между собой. Объекты обычно изображаются в виде прямоугольников с именами. Жизненные линии, которые представляют собой время существования объектов, изображаются в виде вертикальных линий, простирающихся вдоль оси времени. Жизненные линии имеют точки активации, которые показывают периоды, когда объект активен и выполняет операции. Сообщения, которые представляют собой взаимодействие между объектами или компонентами системы, показывают, как объекты обмениваются информацией или вызывают операции друг у друга. Сообщения обычно изображаются в виде стрелок, направленных от отправителя к получателю, с указанием имени сообщения и параметров. Операции представляют собой действия, которые выполняются объектами или компонентами системы. Они указываются внутри фигуры объекта и могут быть связаны с временем выполнения. Диаграммы последовательности могут включать условия, ветвления и циклы, чтобы показать различные сценарии выполнения операций или сообщений в системе.

По мнению Д. Пайлон [17], диаграммы последовательности полезны для анализа и проектирования системы, позволяют лучше понять и визуализировать

последовательность взаимодействия между объектами и компонентами. Они также могут быть использованы в качестве документации и коммуникационного средства между разработчиками и заинтересованными сторонами.

В соответствии с вышесказанным создадим с помощью языка моделирования UML диаграмму последовательности для разрабатываемой развивающей компьютерной игры «Познай себя», которая представлена на рисунке 2.

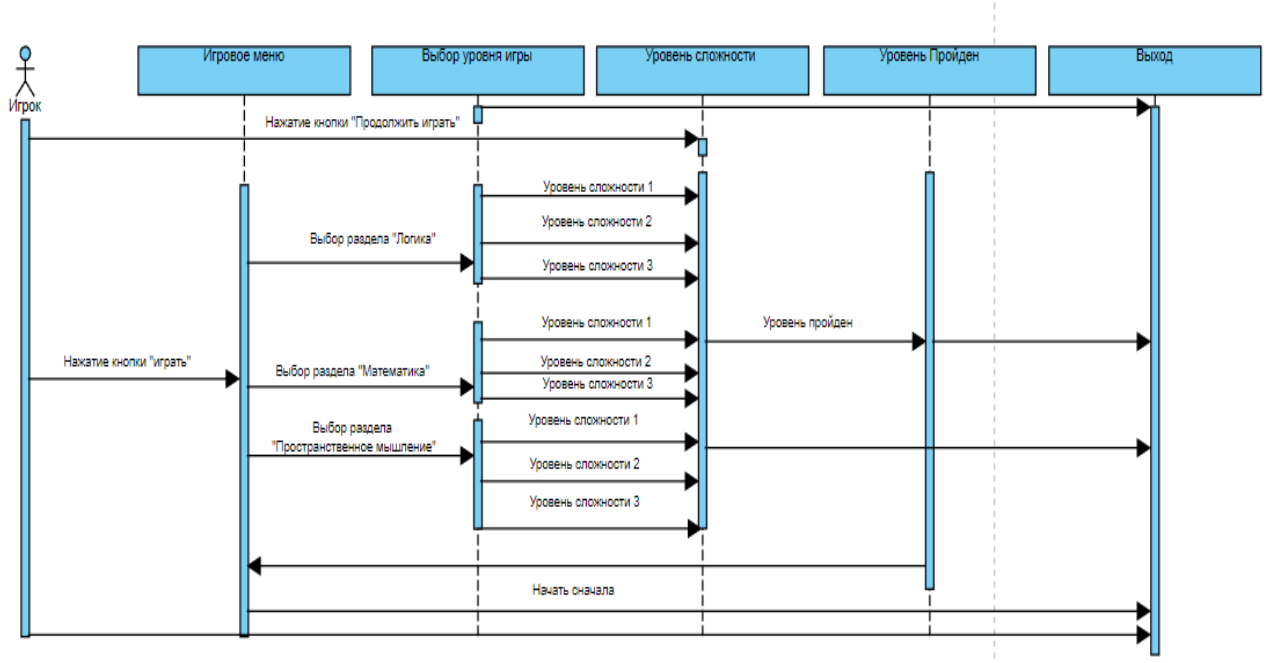

Рисунок 2 – Диаграмма последовательности игры «Познай себя»

UML диаграммы состояний являются одним из видов диаграмм, используемых в разработке программного обеспечения для визуализации поведения системы в различных состояниях [17]. Они представляют собой графическое представление состояний, переходов между состояниями и действий, связанных с этими переходами. Диаграммы состояний представляют собой определенные условия или фазы, в которых может находиться объект или система. Переходы представляют собой изменение состояния объекта или системы и обычно связаны с определенными событиями или условиями.

Переходы изображаются в виде стрелок, указывающих направление изменения состояния. Они могут иметь условия, которые должны быть выполнены, чтобы произошел переход. События на диаграмме бывают внешними или внутренними событиями, которые инициируют переходы между состояниями. События обычно представляются в виде маленьких эллипсов, помещенных на переходы или рядом с состояниями. Примерами событий могут быть: нажатие кнопки; получение сообщения; изменение внутреннего состояния объекта. Действия представляют собой действия или операции, которые выполняются во время перехода между состояниями. Действия могут быть связаны с определенными событиями или переходами и указываются рядом с соответствующими элементами диаграммы.

При выходе из игры игрок возвращается к главному меню и может сохранить свой прогресс для дальнейшей игры без необходимости проходить снова уже пройденные уровни. Сообщение нажатие кнопки «Продолжить играть» на диаграмме выше, представим в виде диаграммы состояний на рисунке 3:

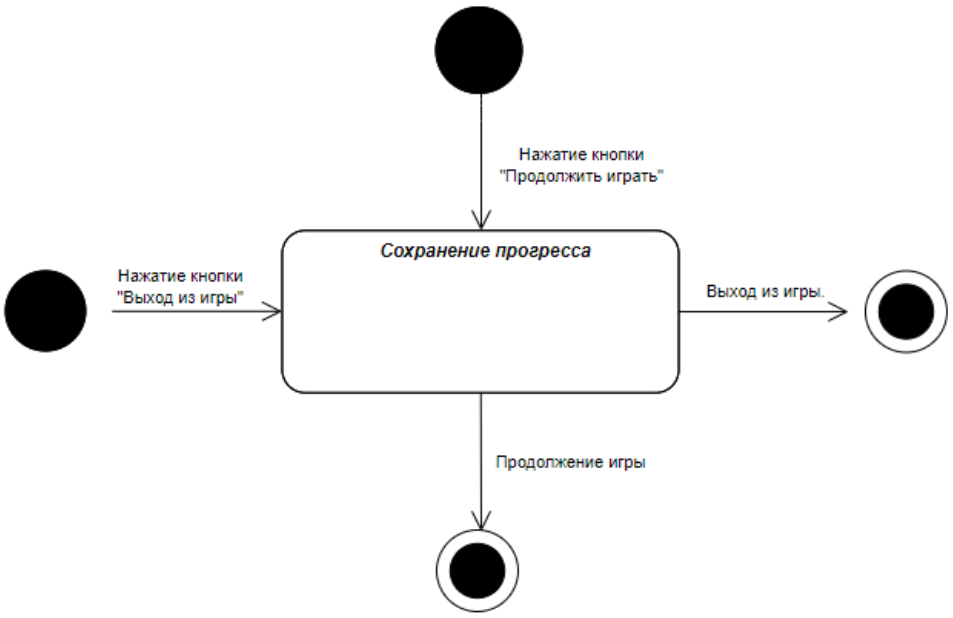

Рисунок 3 – Диаграмма состояний игры «Познай себя»

Как отмечает Крег Ларман [13], диаграмма активности в UML (Unified

Modeling Language) используется для визуализации последовательности действий и потоков управления в системе или процессе. Она позволяет моделировать бизнес-процессы, алгоритмы, взаимодействия между объектами и другие динамические аспекты системы.

Диаграмма активности состоит из узлов (например, действий, начального и конечного узлов, разветвлений и объединений) и дуг, которые связывают узлы и определяют порядок выполнения действий. Объект «Игрок», представленный на рисунке 2, может осуществлять три действия:

– запустить игру;
– продолжить игру;
– выйти из игры.

Модель диаграммы активности представлена на рисунке 4 и отображает выбор игрока в главном меню, состоящий из трёх вариантов событий:

– запустить игру с учётом сохранённого прогресса;
– начать новую игру;
– выйти из игры.

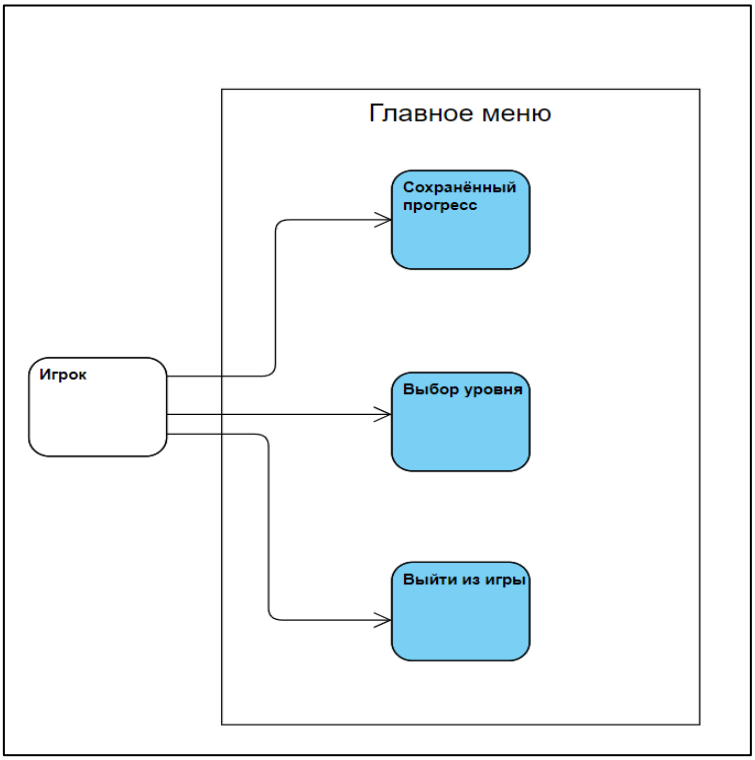

Рисунок 4 –Диаграмма активности объекта «Игрок»

В данном разделе была определена актуальность выбранной темы с точки зрения соответствия требованиям разработки компьютерных игр, основные жанры компьютерных игр, определены характеристики целевой аудитории. Был проведен сравнительный анализ средств разработки, на основе которого выбраны следующие программные компоненты: платформа Unity, интегрированная среда разработки Microsoft Visual Studio Code и графические редакторы Gimp и Inkscape для проектирования спрайтов. Была определена концепция разрабатываемой игры и созданы ее модели в нотации UML, которые отображают процесс игры с точки зрения взаимодействия игрока и игры.

2 Разработка компьютерной развивающей игры на платформе Unity

2.1 Установка программного обеспечения Unity и UnityHub

Для начала разработки было необходимо скачать специальное программное обеспечение. В первую очередь, скачать платформу Unity с официального сайта. Для этого нужно перейти на официальный сайт [35] и выбрать раздел «Скачать Unity» (Download Unity).

Далее необходимо выбрать версию Unity. Рекомендуется выбирать стабильную последнюю версию для получения всех последних обновлений и функций. На странице загрузки нужно выбрать операционную систему, на которой планируется использовать Unity. Последняя поддерживает различные операционные системы, включая Windows и macOS. В рамках нашего исследования будем использовать версию для Windows.

Может потребоваться выбрать модули или плагины, которые соответствуют потребностям разработки игр. После завершения загрузки нужно запустить установочный файл Unity и следовать инструкциям на экране, чтобы закончить процесс установки. Во время установки может потребоваться выбрать компоненты, которые нужно установить, и указать путь установки. После завершения установки можно запускать Unity. Его можно найти в меню «Пуск» (Windows) или в папке «Приложения» (macOS). При первом запуске Unity может потребоваться войти в свою учетную запись или создать новую. Создание учётной записи продемонстрировано на рисунке 5.

После прохождения регистрации учётной записи и установки Unity, можно приступать к созданию проекта в UnityHub. UnityHub позволяет найти полезные статьи, обзоры, уроки, советы и примеры, которые помогут разработчикам освоить различные аспекты работы с Unity. Здесь можно задать вопросы и получить помощь от опытных разработчиков, а также поделиться своими проектами и идеями.

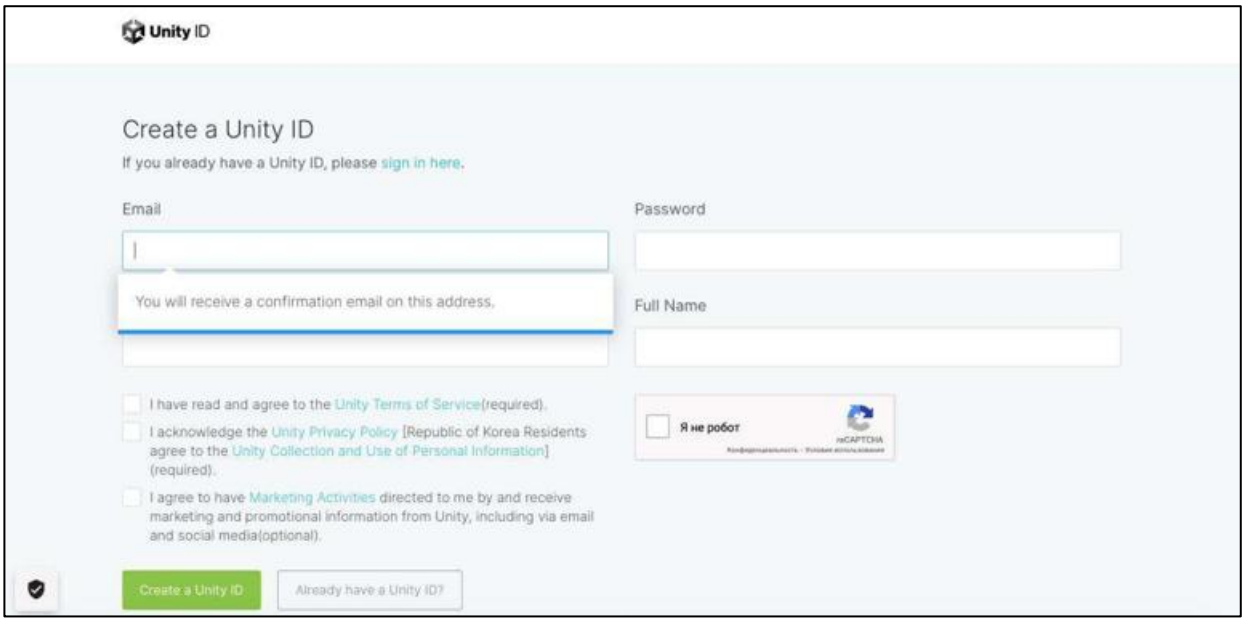

Рисунок 5 – Создание учётной записи UnityID

UnityHab является площадкой для обмена знаниями и опытом в сообществе разработчиков Unity. Здесь можно получить поддержку, найти вдохновение и быть в курсе последних новостей и трендов в мире разработки игр на платформе Unity. Так же UnityHub служит для создания и сохранения игрового проекта игры, что представлено на рисунке 6.

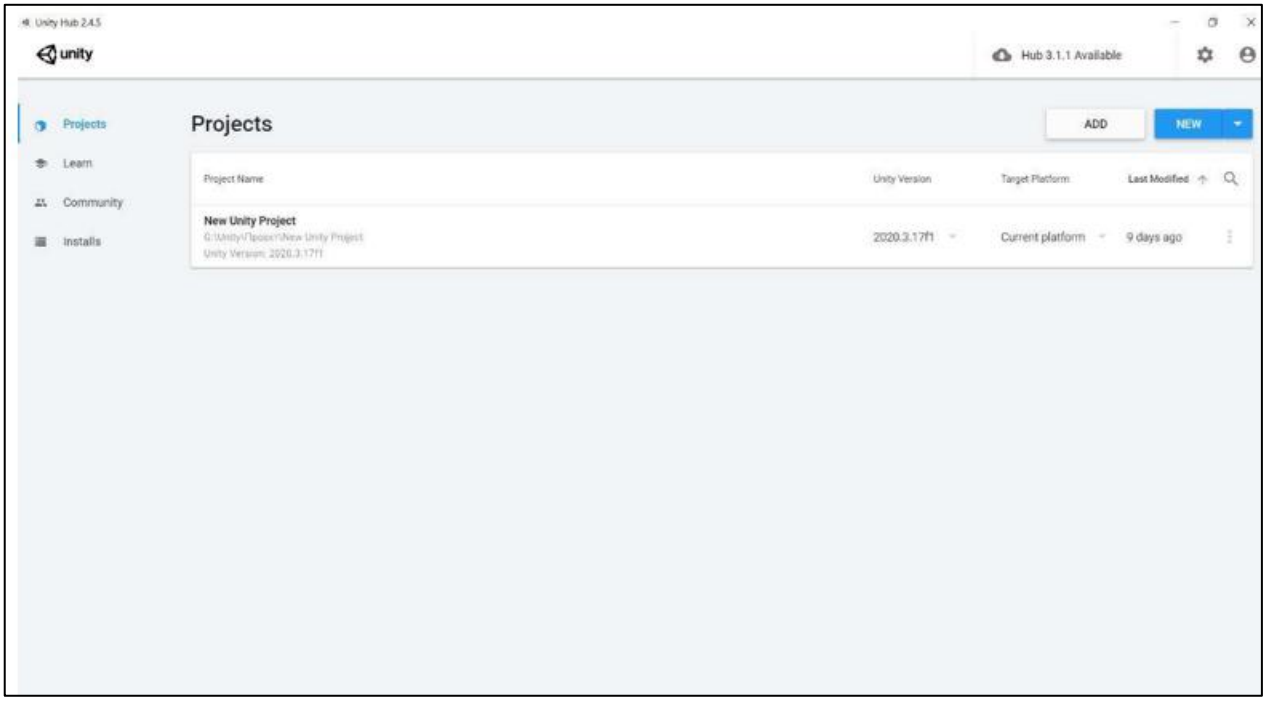

Рисунок 6 – Окно UnityHub

Visual Studio Code [40]

Visual Studio Code является одним из популярных редакторов кода, который можно использовать в среде Unity для разработки игр. Visual Studio Code можно бесплатно скачать как утилиту для работы в Unity, после выбора операционной системы скачиваем Visual Studio Code, что представлено на рисунке 7.

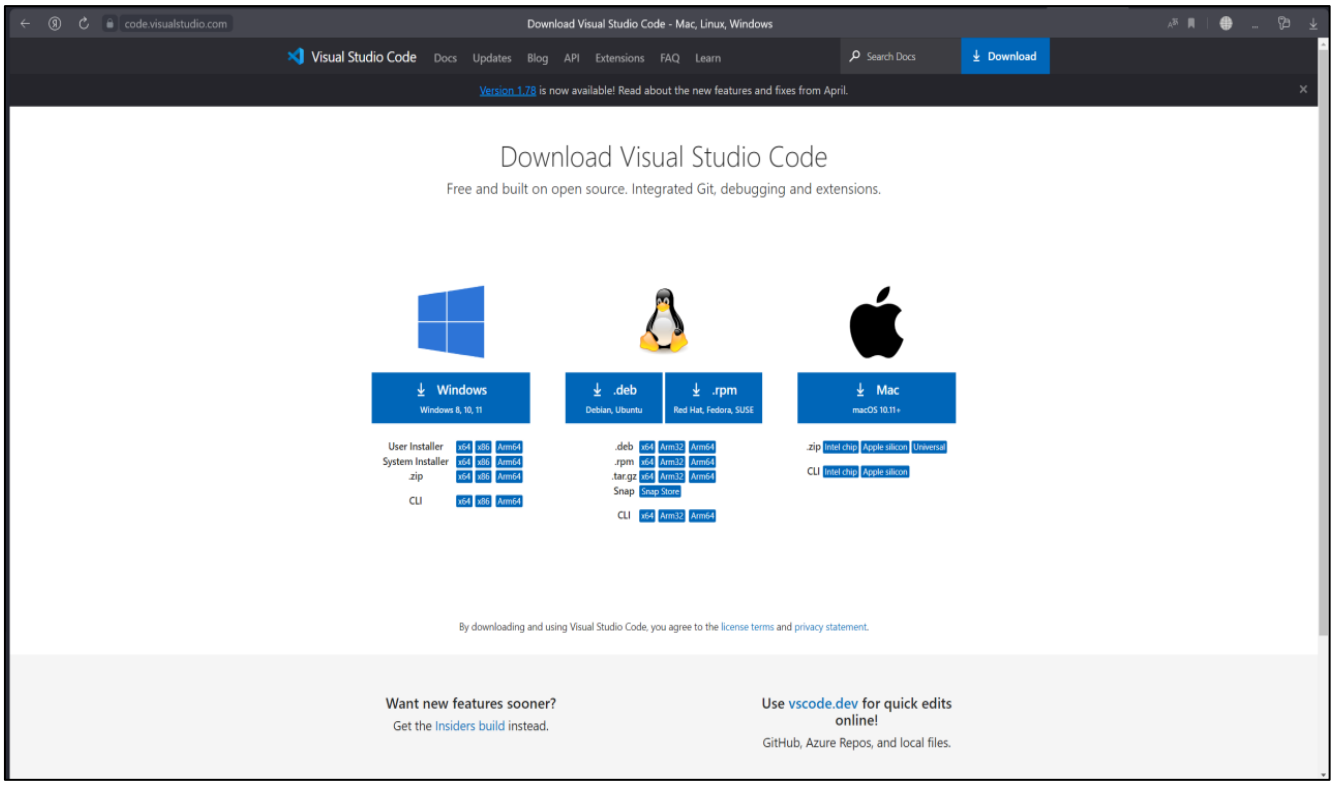

Рисунок 7 – Установка Visual Studio Code

Audacity [27]

Audacity – это бесплатное и открытое программное обеспечение для редактирования аудио. Оно предоставляет различные инструменты и функции для записи, обработки и манипулирования звуковыми файлами.

В игре предусмотрено озвучивание заданий и выбранных ответов для помощи детям, не умеющим читать. Эта программная утилита поможет создать озвучивание. Установка Audacity, как продемонстрировано на рисунке 8.

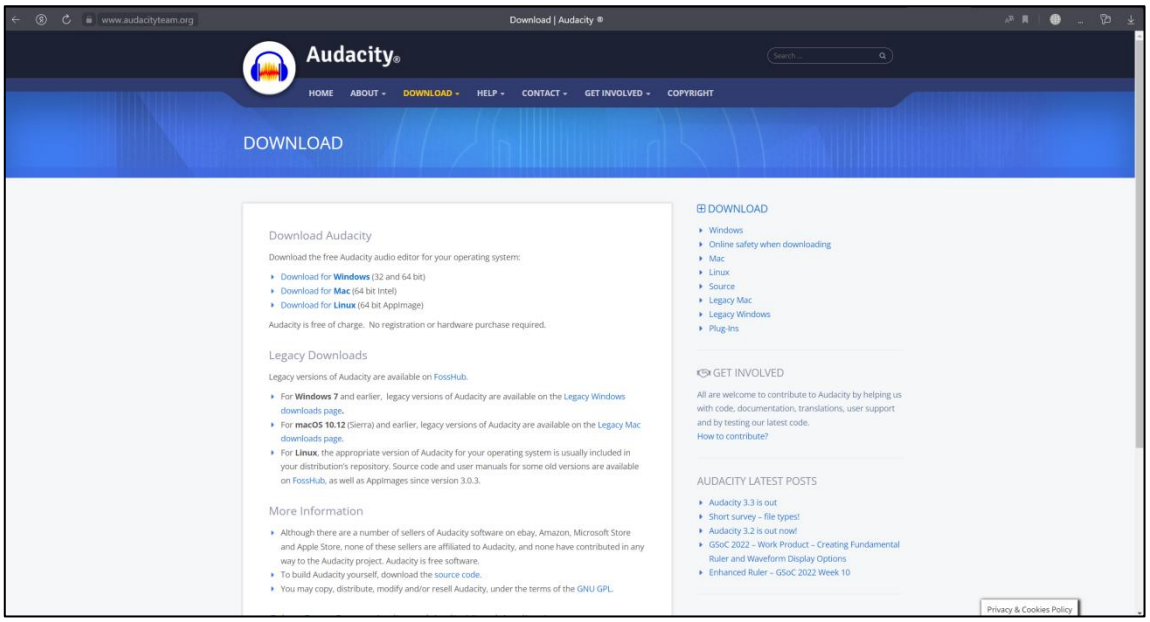

Рисунок 8 – Установка с официального сайта Audacity

После установки всего необходимого можно приступать к созданию проекта в Unity. Для этого в ранее установленном UnityHub нужно нажать кнопку «New project» для создания нового проекта. Далее необходимо выбрать, сколько пространств будет в разрабатываемой игре и написать имя проекта, как представлено на рисунке 9.

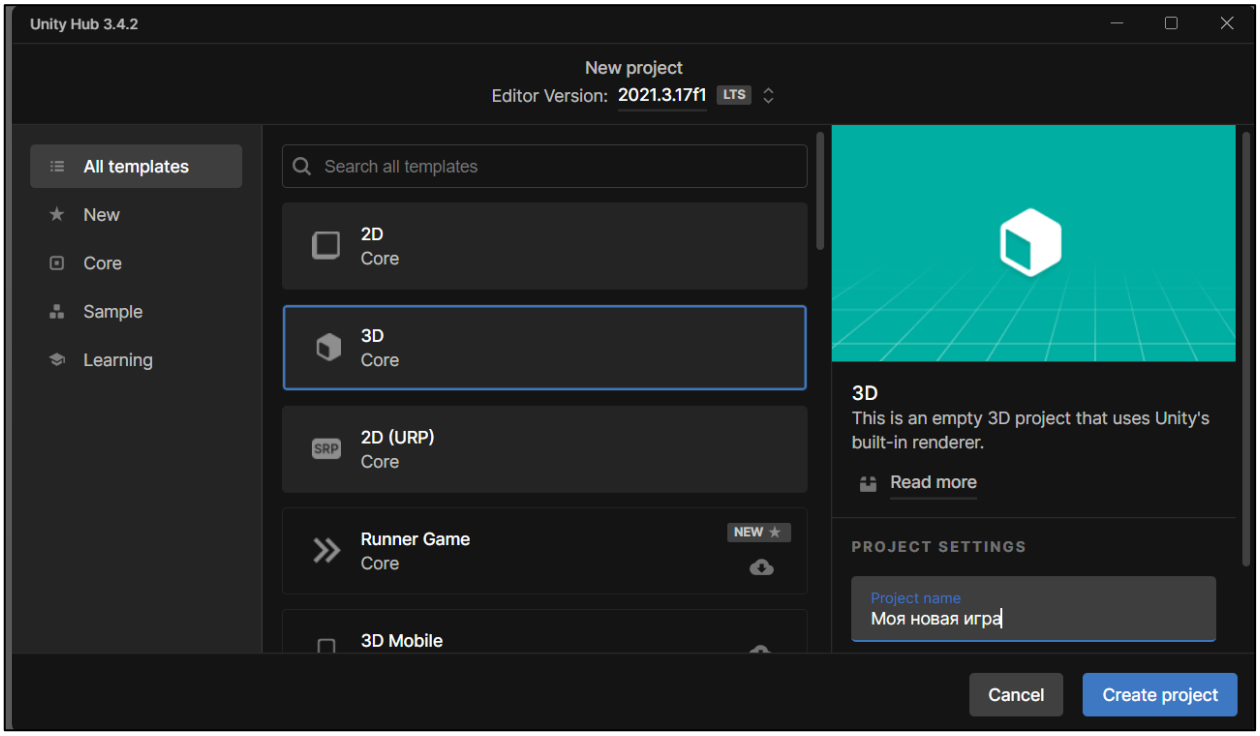

Рисунок 9 – Создание проекта игры

После нажатия на кнопку «Create project» создастся пустой проект Unity, как показано на рисунке 10.

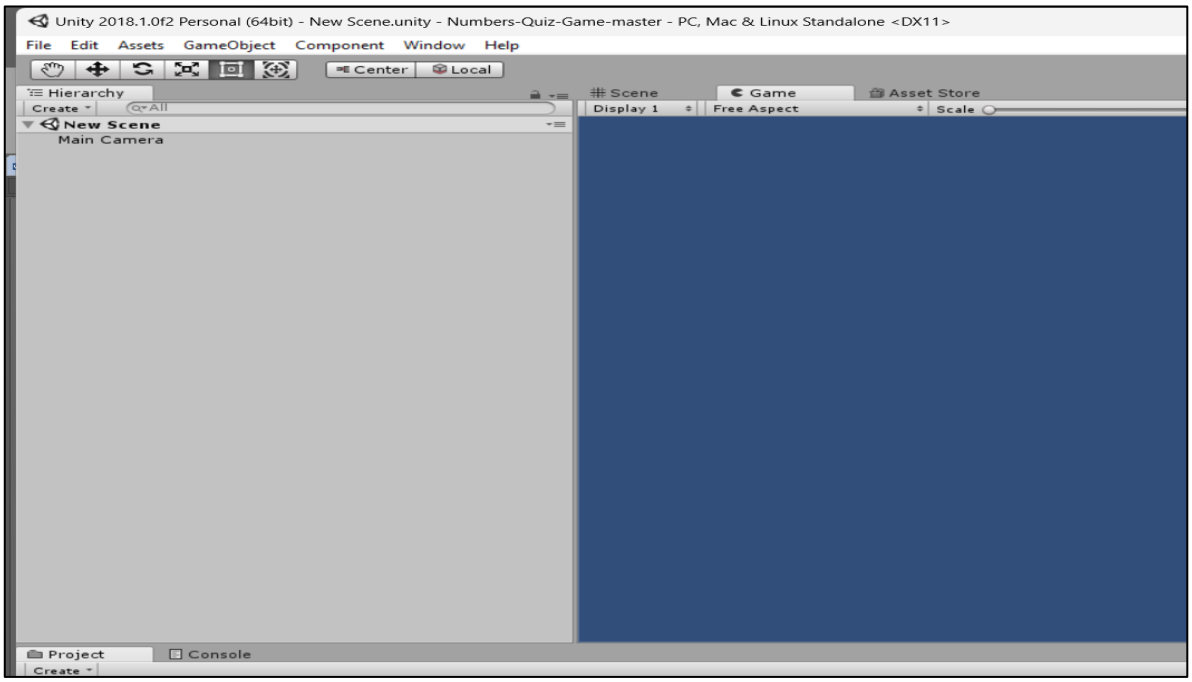

Рисунок 10 – Пустой проект игры

После создания пустого проекта нужно подключить Visual Studio Code для работы со скриптами в проекте. Что бы это сделать нужно зайти в меню «Edit», а после в «Project settings», как показано на рисунке 11.

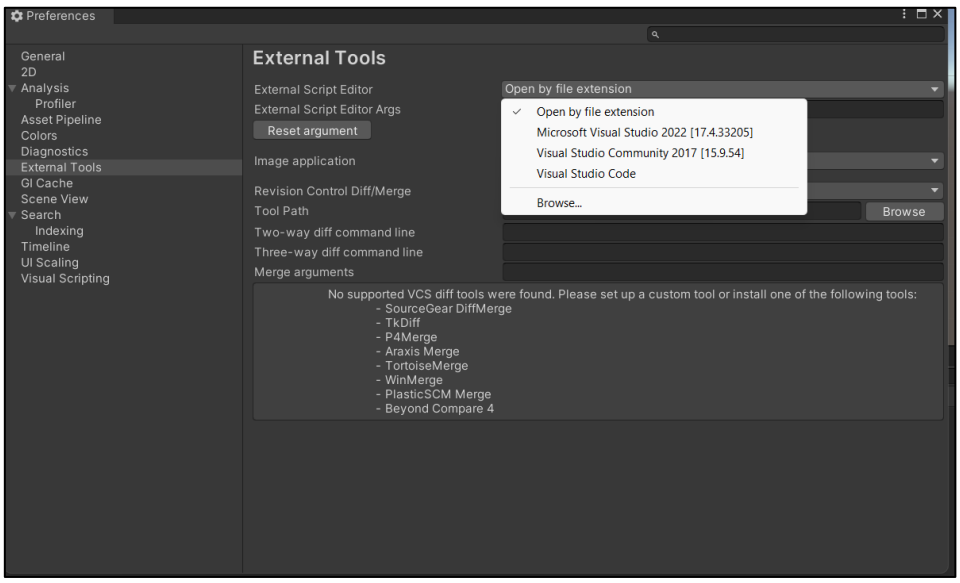

Рисунок 11 – Настройка работы скриптов в Visual Studio Code

2.2 Разработка спрайтов игры и их интегрирование в проект

Разработка спрайтов игры в программе GIMP может быть увлекательным и творческим процессом, но при создании спрайтов следует учитывать следующие особенности:

1. Работа с прозрачностью: GIMP поддерживает прозрачность, что является важной особенностью для спрайтов игр.

2. Работа со слоями: GIMP предлагает мощные функции работы со слоями. При создании следует создавать отдельные слои для различных элементов спрайта, что позволит легко редактировать и управлять каждой частью отдельно. Это особенно полезно при создании анимации спрайтов.

3. Использование кистей и инструментов рисования: GIMP предоставляет разнообразные кисти и инструменты для рисования, что позволит создавать уникальные детали и эффекты на спрайтах.

4. Анимация спрайтов: GIMP поддерживает создание анимированных спрайтов путем использования кадров и временных слоев. Это полезно для создания движения или анимации спрайтов.

5. Использование фильтров и эффектов: GIMP имеет множество встроенных фильтров и эффектов, которые могут быть применены к спрайтам для добавления текстур, освещения, тени и других эффектов.

6. Экспорт в различные форматы: GIMP позволяет экспортировать спрайты в различные форматы, такие как PNG, GIF, JPEG и другие поддерживаемые спрайты для unity.

На рисунке 12 представлено создание спрайта для фона.

На рисунке 13 представлены слои, с помощью которых был создан фон игры.

Остальные спрайты развивающей компьютерной игры «Познай себя» были созданы аналогично.

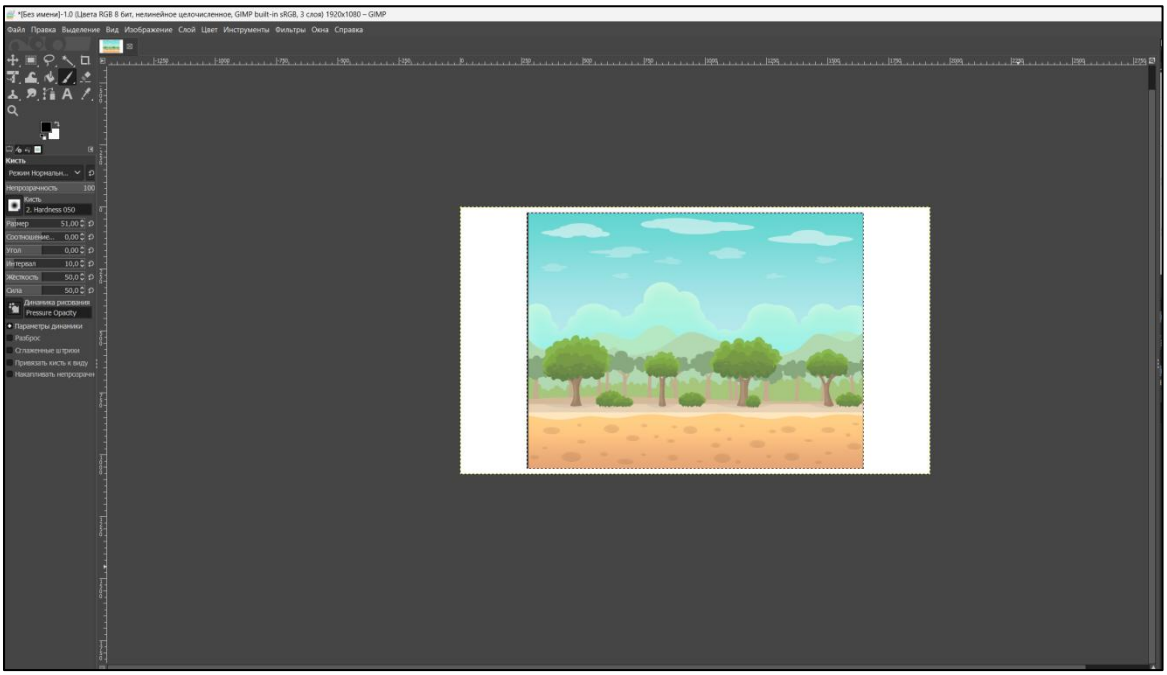

Рисунок 12 – Создание спрайта фона игры

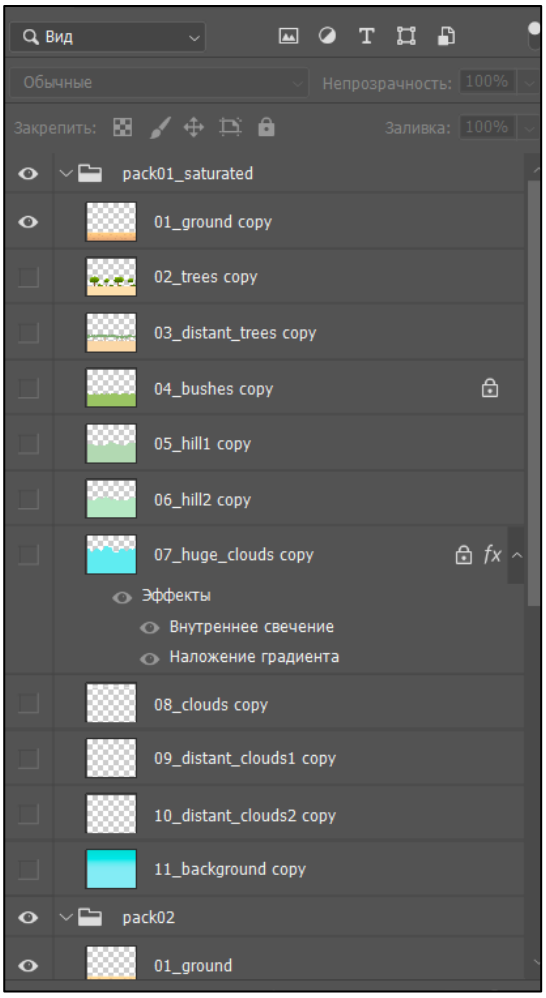

Рисунок 13 – Создание фона игры по слоям

Для интеграции спрайтов в проект необходимо выполнить следующие шаги:

1. Подготовить спрайты к загрузке: для этого необходимо убедиться, что спрайты имеют правильные размеры и формат. Рекомендуемый формат для спрайтов – PNG, так как он поддерживает прозрачность. Если спрайты имеют различные анимационные кадры, нужно разделить их на отдельные изображения или использовать спрайтовые атласы («sprite atlas»), которые объединяют несколько спрайтов в одном изображении.

2. Далее нужно создать папку для спрайтов в проекте Unity. Для этого нужно щёлкнуть правой кнопкой мыши на панели «Project» в Unity, выбрать «Create» и «Folder», чтобы создать новую папку. Назовём ее, к примеру, «Sprites».

3. Добавляем спрайты в созданную папку: откроем папку со спрайтами на компьютере, а затем переместим их в созданную папку «Sprites» в Unity. Unity автоматически импортирует спрайты в проект, как показано на рисунке 14.

4. Редактирование спрайтов так же доступно в Unity.

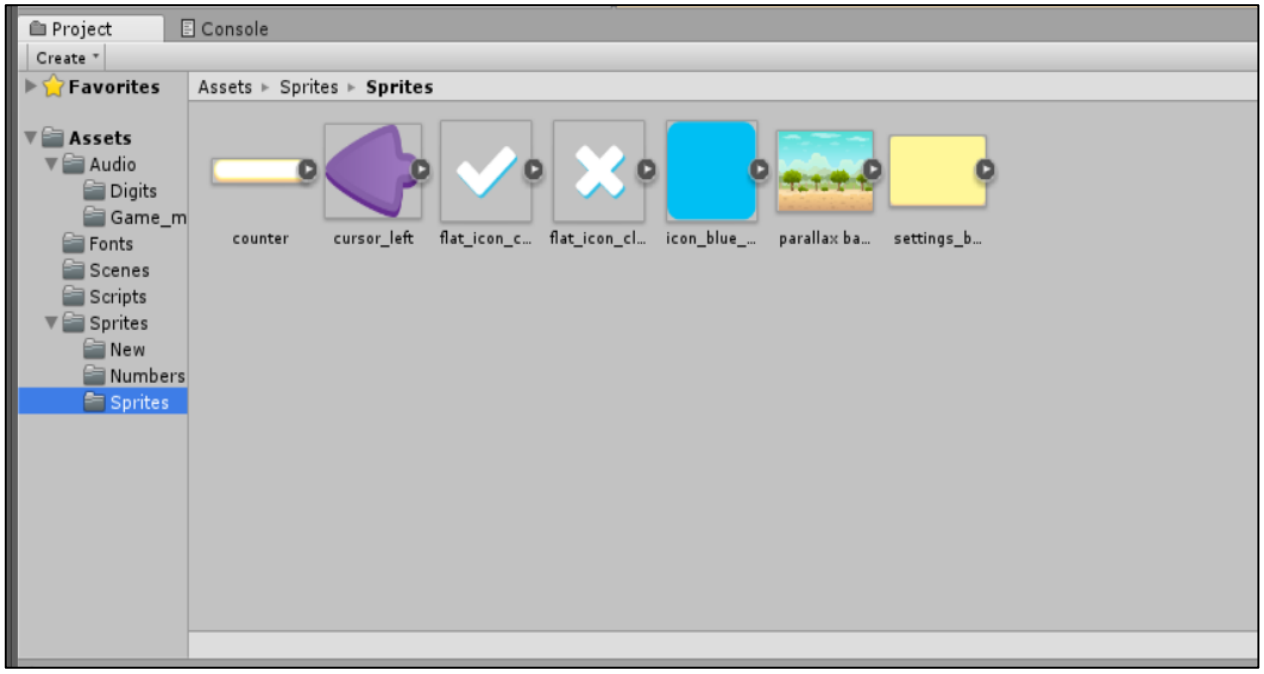

Рисунок 14 – Добавление спрайтов в проект Unity

5. Создание спрайтовых объектов: в окне «Project» в Unity, выбираем спрайт, который необходимо использовать, и перемещаем его в сцену или в иерархию объектов. Это создаст спрайтовый объект (Sprite Renderer) в сцене.

6. Настройка спрайтовых объектов: можно изменить свойства спрайта, такие как размер, масштаб, положение и поворот, в инспекторе объекта. Также можно настроить анимацию спрайта, если имеется анимированный спрайт. Unity предлагает различные компоненты и системы анимации для работы со спрайтами, что продемонстрировано на рисунке 15.

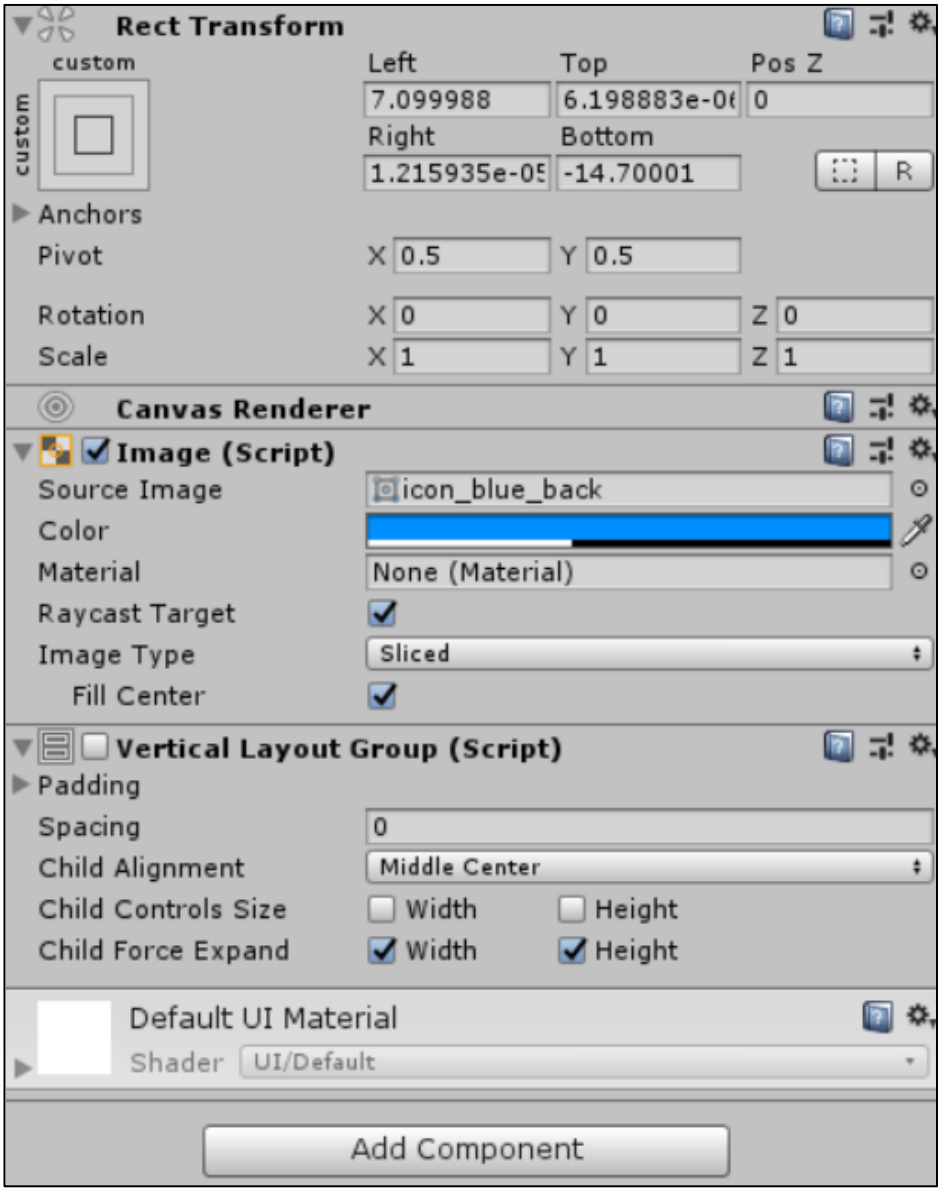

Рисунок 15 – Изменение свойств спрайтового объекта

2.3 Создание интерфейса компьютерной развивающей игры на платформе Unity

2.3.1 Описание структуры интерфейса

Общая структурная схема интерфейса игры «Познай себя» изображена на рисунке 16.

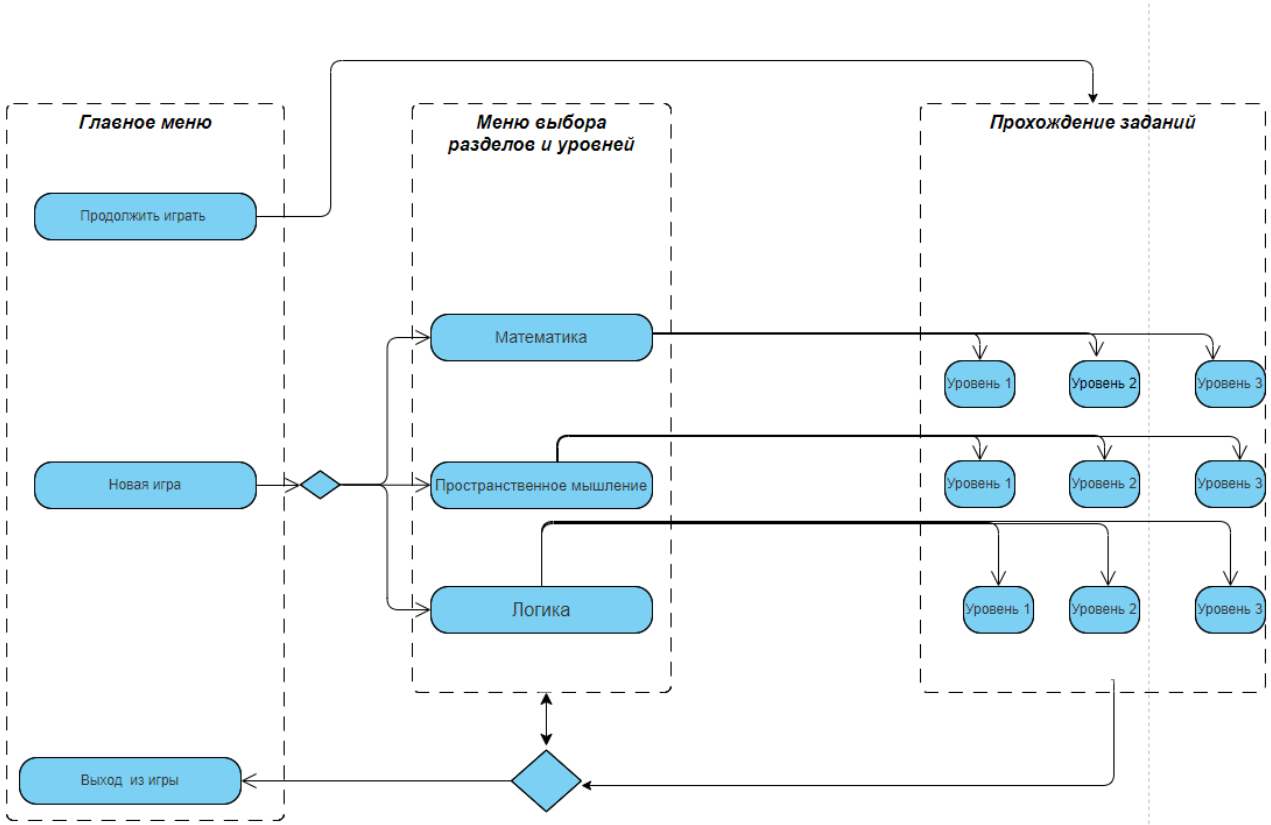

Рисунок 16 – Общая структурная схема интерфейса игры «Познай себя»

При запуске игры отображается главное меню, в котором должны быть отображены три кнопки: «Продолжить играть», «Новая игра», «Выйти из игры». Так же в меню необходимо отобразить название игры, которое будет располагаться в верхней части экрана и выполнено в ярких цветах. Соответствующие цвета должен иметь и сам интерфейс. Черновой прототип главного меню представлен на рисунке 17.

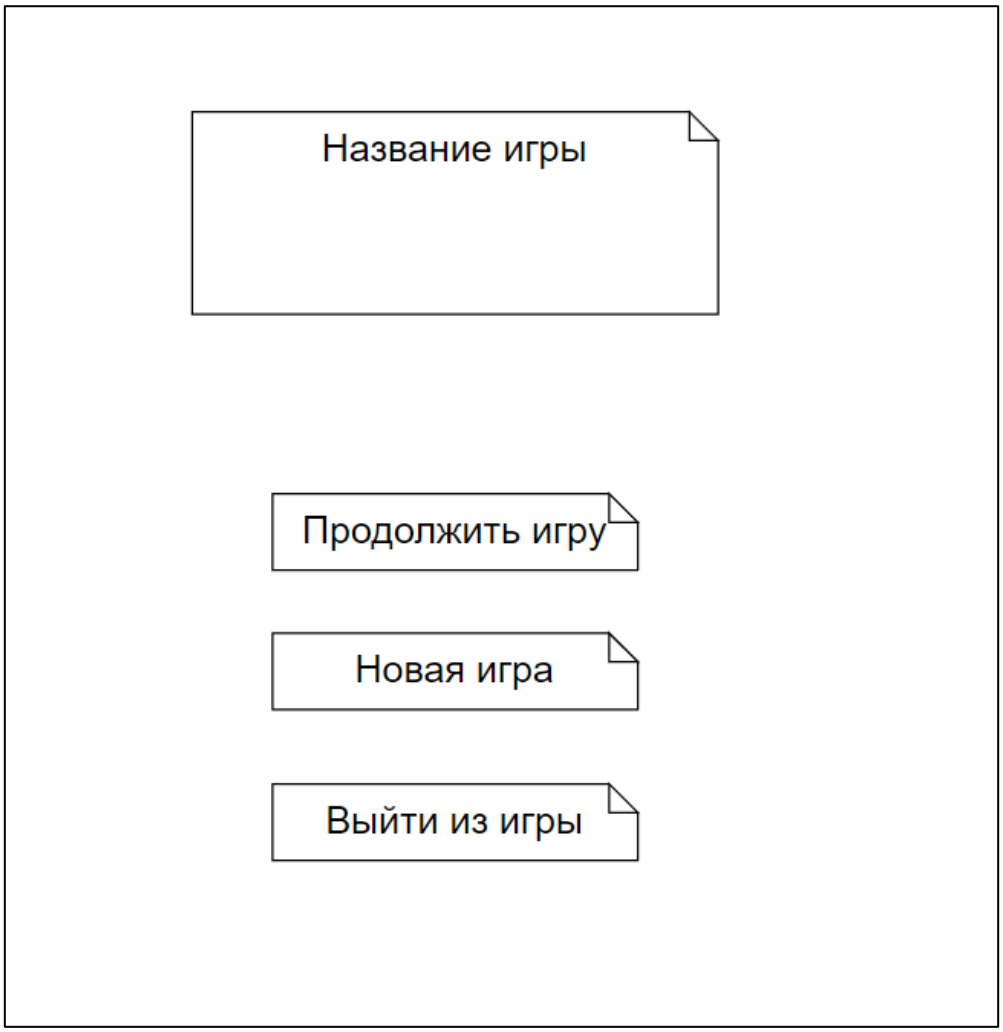

Рисунок 17 – Черновой прототип главного меню

В процессе начала новой игры предусмотрен выбор трёх видов направления прохождения (разделов), таких как «Математика», «Логика», «Пространственное мышление». Для каждого раздела предусмотрены три уровня сложности. Черновой прототип выбора раздела и уровней сложности игры представлен на рисунке 18.

При нажатии на соответствующий уровень игрок переходит к экрану с заданиями. Чтобы перейти к другому уровню, игрок возвращается в меню выбора разделов и уровней.

Рисунок 18 – Черновой прототип выбора раздела и уровней сложности игры

Рассмотрим схему расположения элементов заданий уровней. Уровни в развивающей игре должны усложняться по мере повышения, что описано в требованиях к игре в пункте 1.1.4 параграфа 1.1 настоящей работы. На каждом уровне игроку предлагается тестовое задание в закрытой форме с выбором одного правильного ответа. Нажатие на один из вариантов ответа определяет сюжетную развилку игры, которая зависит от уровня и правильности ответов. Схемы расположения элементов в момент выбора варианта ответа представлена на рисунках 19–20.

Отметим, что количество вариантов ответов может варьироваться от двух до четырёх. Это зависит от типа задания и никак не связано с выбором раздела или уровня.

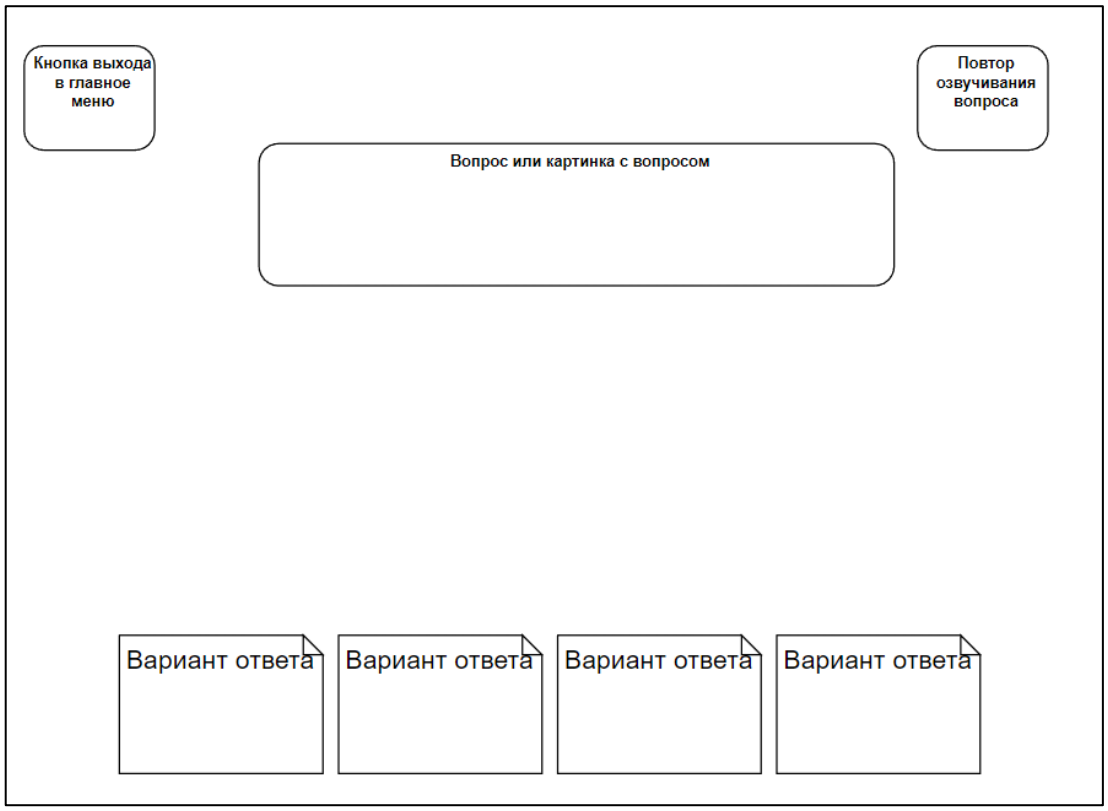

Рисунок 19 – Схема задания уровня один

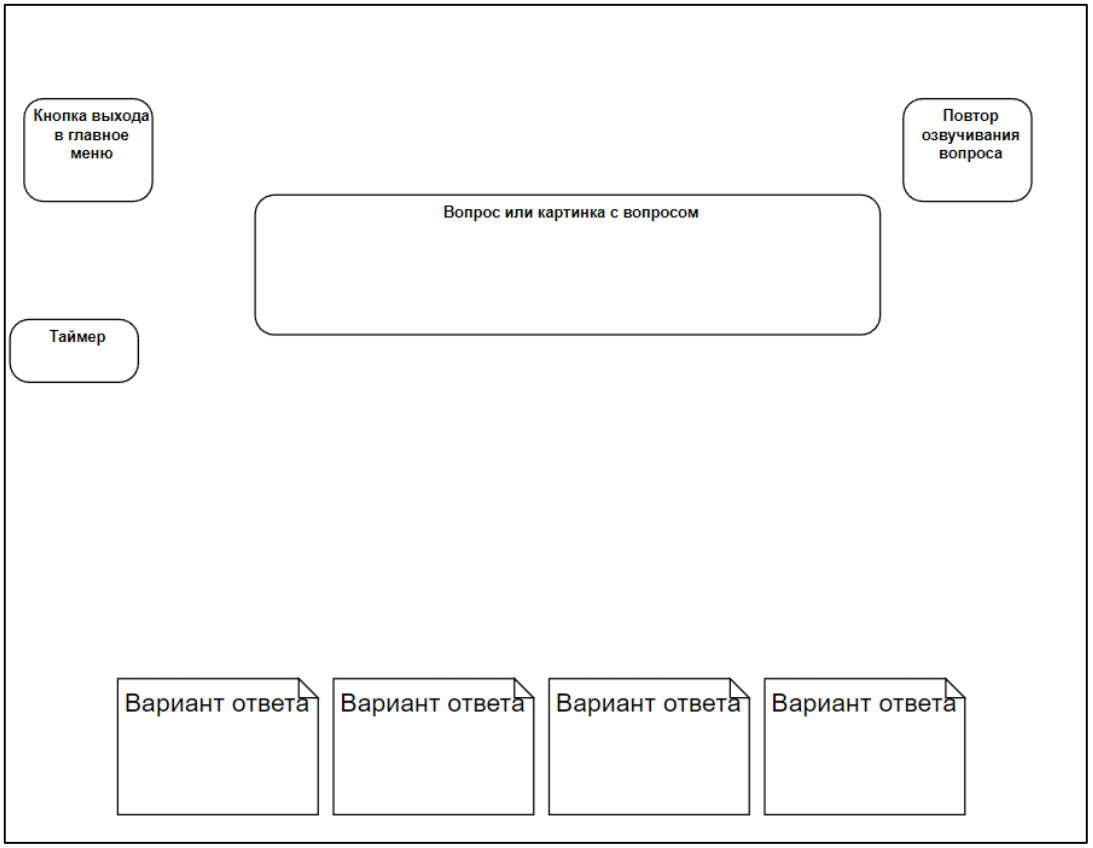

Рисунок 20 – Схема задания с таймером

В результате работы над проектированием интерфейса были созданы схемы расположения элементов игры (черновые прототипы). Данные схемы помогут в процессе реализации игры при расположении спрайтов элементов на различных экранах.

2.3.2 Реализация интерфейса игры «Познай себя» на платформе Unity

Интерфейс развивающей игры спроектирован с помощью модуля Unity Users Interface. Unity UI предоставляет мощные инструменты для создания пользовательского интерфейса игры.

При создании сцены меню было написано несколько скриптов, воспринимающих нажатие кнопок пользователем, а также применяемых ранее нарисованных в программе Gimp спрайтов для создания игрового меню, представленные на рисунке 12 параграфа 2.2. Главное меню открывается при запуске игры и позволяет пользователю начать новую игру, продолжить играть (при наличии сохранения) и выйти из приложения. Название игры «Познай себя» отображается в пользовательском меню при помощи UI.Text. Кнопки в главном меню созданы при помощи UI.Button. Экран главного меню представлен на рисунке 21.

Рисунок 21 – Главное меню развивающей игры «Познай себя»

Меню выбора разделов и уровней, представленное на рисунке 22, включает в себя предложенные на выбор три совершенно разные направления игры, такие как «Математика», «Логика» и «Пространственное мышление». Для каждого направления созданы три уровня сложности.

Рисунок 22 – Меню выбора разделов и уровней игры «Познай себя»

Проектирование интерфейса игровых уровней отличается в зависимости от желаемого конечного результата. Для реализации игровых уровней были написаны C# скрипты, взаимодействующие с пользователем. Скрипты позволили реализовать в игре такие сущности, как: повтор воспроизведения задания, меняющиеся кнопки в зависимости от задаваемого вопроса.

Для реализации игровых уровней был настроен Canvas-элемент, позволяющий создать пользовательское меню разрабатываемого уровня. Он содержит следующие компоненты:

1) Canvas scaler – это компонент, который позволяет настраивать масштабирование и адаптивность элементов пользовательского интерфейса.

Этот компонент даёт возможность пользовательскому интерфейсу масштабироваться и адаптироваться под различные разрешения экрана и устройства.

2) Graphic Raycaster – это компонент в Unity, который позволяет определить, какие элементы графики (Graphic) в canvas будут реагировать на события пользовательского ввода, такие как нажатие кнопки мыши или касание экрана на мобильных устройствах.

3) Image используется для отображения 2D графики в Canvas. Он позволяет отображать спрайты, текстуры и другие изображения на экране.

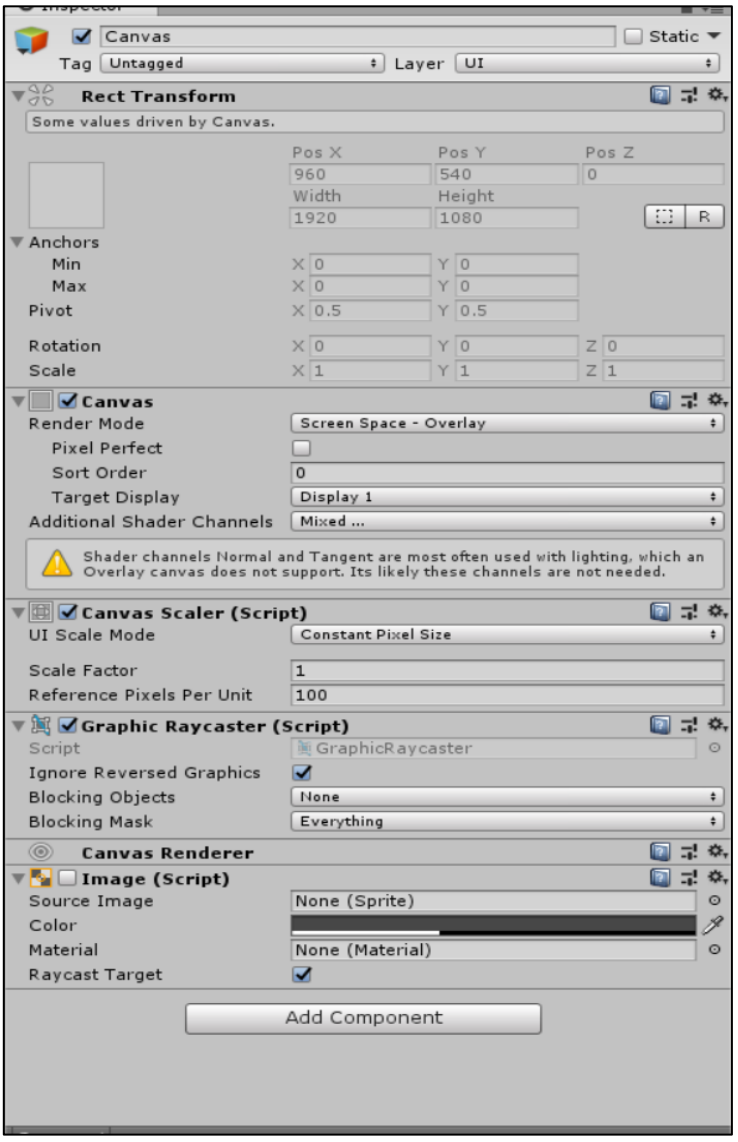

Рисунок 23 – Настройка Canvas элементов пользовательского интерфейса

Для реализации выбора правильного ответа была добавлена панель ответов (Answer Panel), в которой добавлено от двух до четырёх кнопок для выбора пользователем правильного ответа. Для настройки был написан скрипт воспроизведения выбора ответа и замены выбранных вариантов в зависимости от ответа, как показано на рисунке 24.

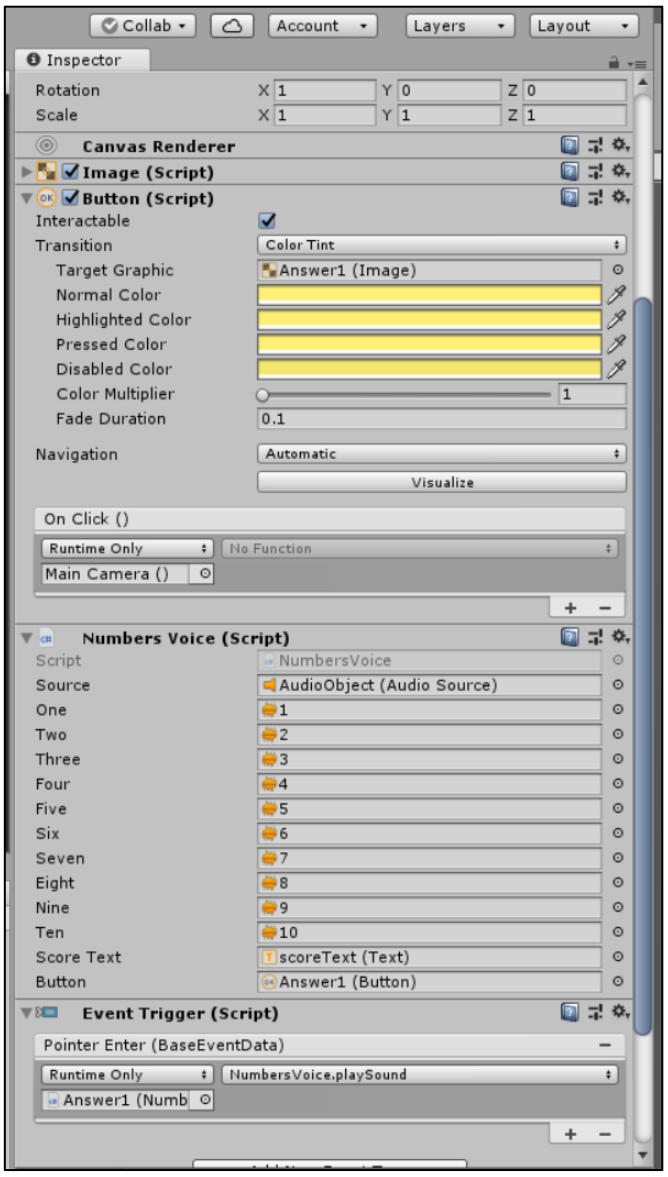

Рисунок 24 – Настройка Answer Panel элементов пользовательского интерфейса

Воспроизведение происходит посредством наведения на кнопку выбора варианта ответа. Скрипт Number voice добавляет воспроизведение 10 дорожек, представленных на рисунке 25 и сделанных при помощи программы Audacity о которой было написано в параграфе 2.1.

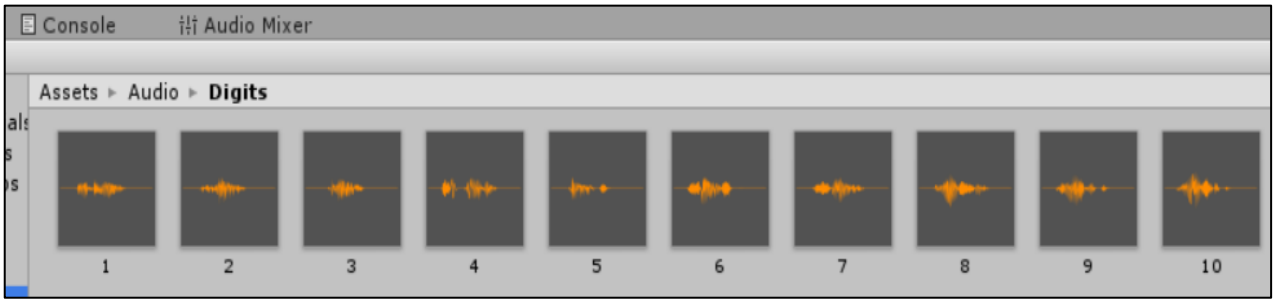

Рисунок 25 – Воспроизведение нажатия кнопок

Воспроизведение текста можно повторить, что обязательно для реализации в игре для детей 6–7 лет. В игре предусмотрено как воспроизведение задания при переходе на уровень, так и при нажатии на соответствующую кнопку. Панель настройки изображена на рисунке 26.

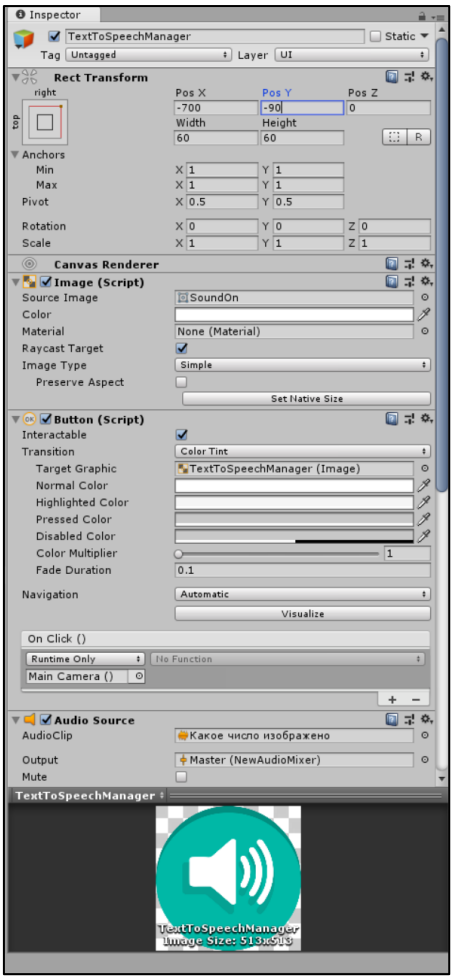

Рисунок 26 – Панель настройки кнопки озвучивания

2.4 Реализация системы заданий компьютерной развивающей игры «Познай себя» на платформе Unity

Рассмотрим реализацию системы заданий на платформе Unity в соответствии с тематикой разделов.

2.4.1 Реализация системы заданий по разделу «Математика»

Система заданий по математике учитывает характеристики целевой аудитории и представляет собой совокупность тестовых вопросов закрытого типа с выбором одного правильного ответа.

Для реализации трёх уровней по разделу «Математика» было спроектировано меню Canvas, содержащее в себе панель отображения вопроса («QuestionPanel»), кнопку выхода из уровня («ExitButton»), панель с вариантами ответов («AnswerPanel»), панель, позволяющая озвучить задание для игрока и при наведении на вариант ответа цифру «AudioObject». Так же были созданы вспомогательные элементы, продемонстрированные на рисунке 27.

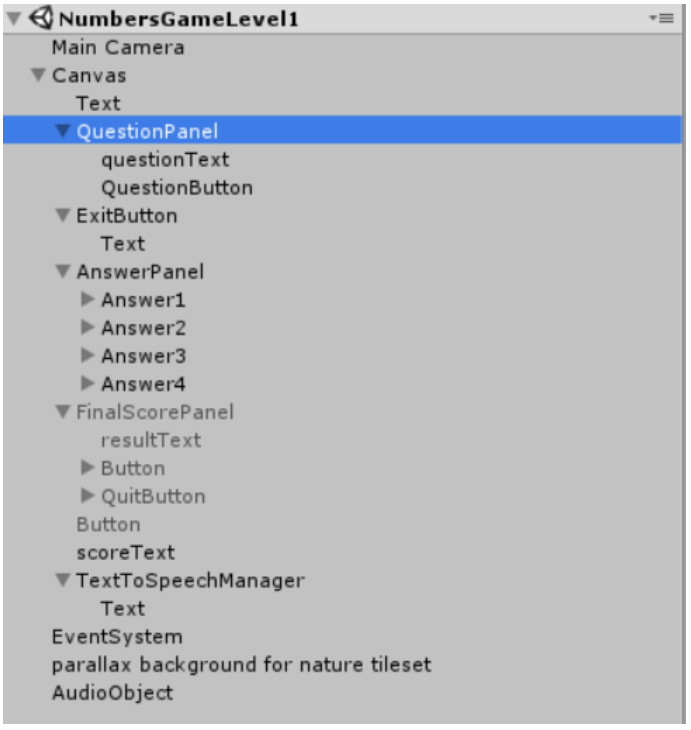

Рисунок 27 – Canvas-меню уровней раздела «Математика»

Три уровня сложности реализованы следующим образом.

На первом уровне сложности игроку нужно выбрать из предложенных вариантов какое число изображено перед ним. В реализации предусмотрена возможность ошибки: игровой процесс покажет с помощью соответствующего спрайта, что данный ответ не верен и игрок может ответить повторно. Задание первого уровня сложности по разделу «Математика» представлено на рисунке 28. Как было описано ранее, для детей, не умеющих читать, при наведении на вариант ответа происходит его озвучивание.

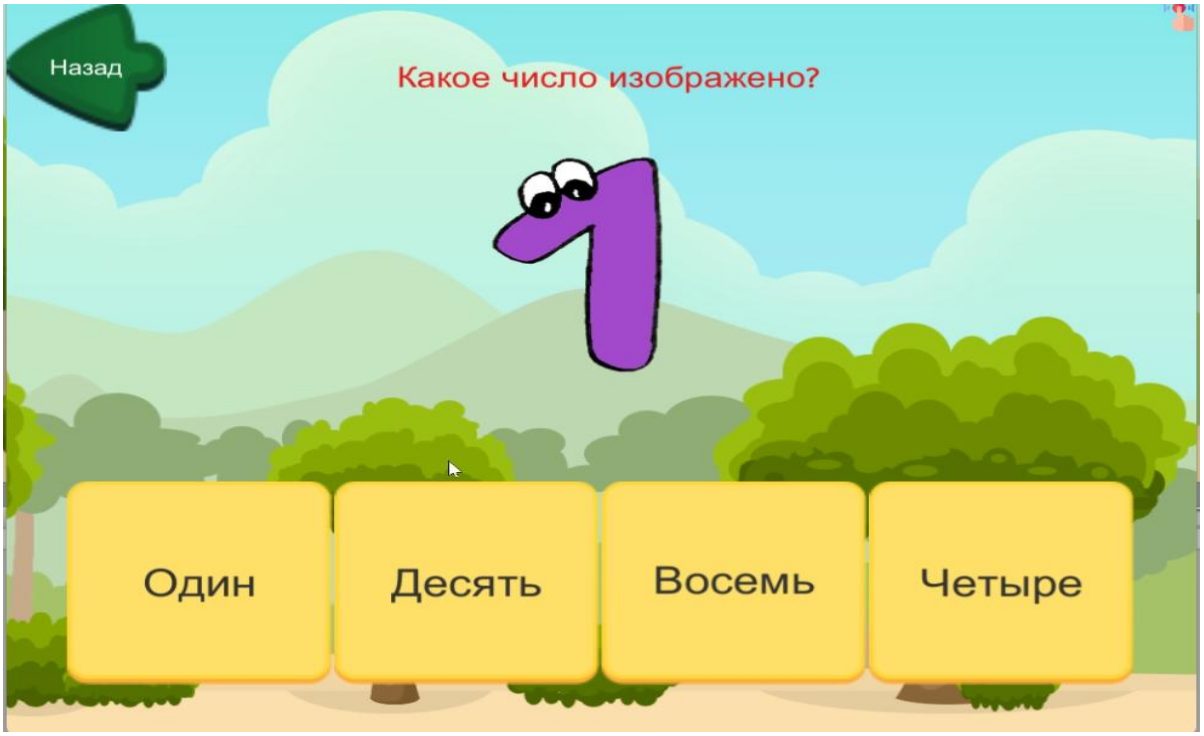

Рисунок 28 – Первый уровень сложности, раздел «Математика»

На втором уровне сложности игроку нужно посчитать, сколько фруктов изображено перед ним, что продемонстрировано на рисунке 29. Права на ошибку на этом уровне не дается, а вводится система подсчета правильных ответов.

Рисунок 29 – Второй уровень сложности, раздел «Математика»

На третьем уровне сложности игроку, так же, как и на втором, нужно посчитать количество фруктов, но вводится счётчик времени, который изображен в левой части рисунка 30.

Рисунок 30 – Третий уровень сложности, раздел «Математика»

2.4.2 Реализация системы заданий по разделу «Логика»

В своей работе В. Ю. Лыскова [15], Е. А. Ракитина [15] указывают, что логика – это наука о законах мышления и рассуждения, которая изучает правильные методы и принципы вывода. Она занимается формальным анализом аргументов и структуры рассуждений с целью определить, являются ли они логически верными. На основе этой работы были спроектированы уровни раздела «Логика» в развивающей компьютерной игре «Познай себя». На рисунке 31 представлен первый уровень сложности раздела «Логика».

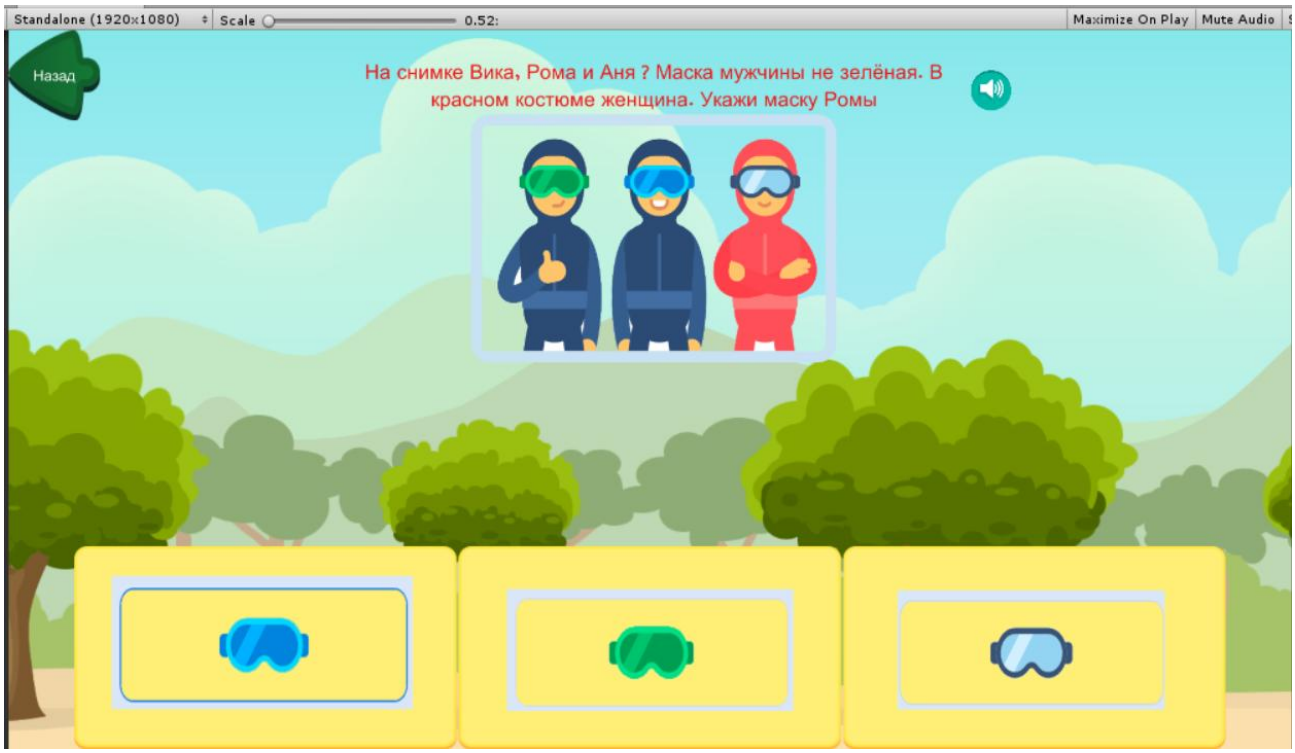

Рисунок 31 – Первый уровень сложности, раздел «Логика»

Второй уровень сложности содержит задания на соотношение правильной формы с тенью от предмета, так же у игрока появляется счётчик правильных ответов, что показано на рисунке 32. Отметим, что это задание развивает и пространственные представления.

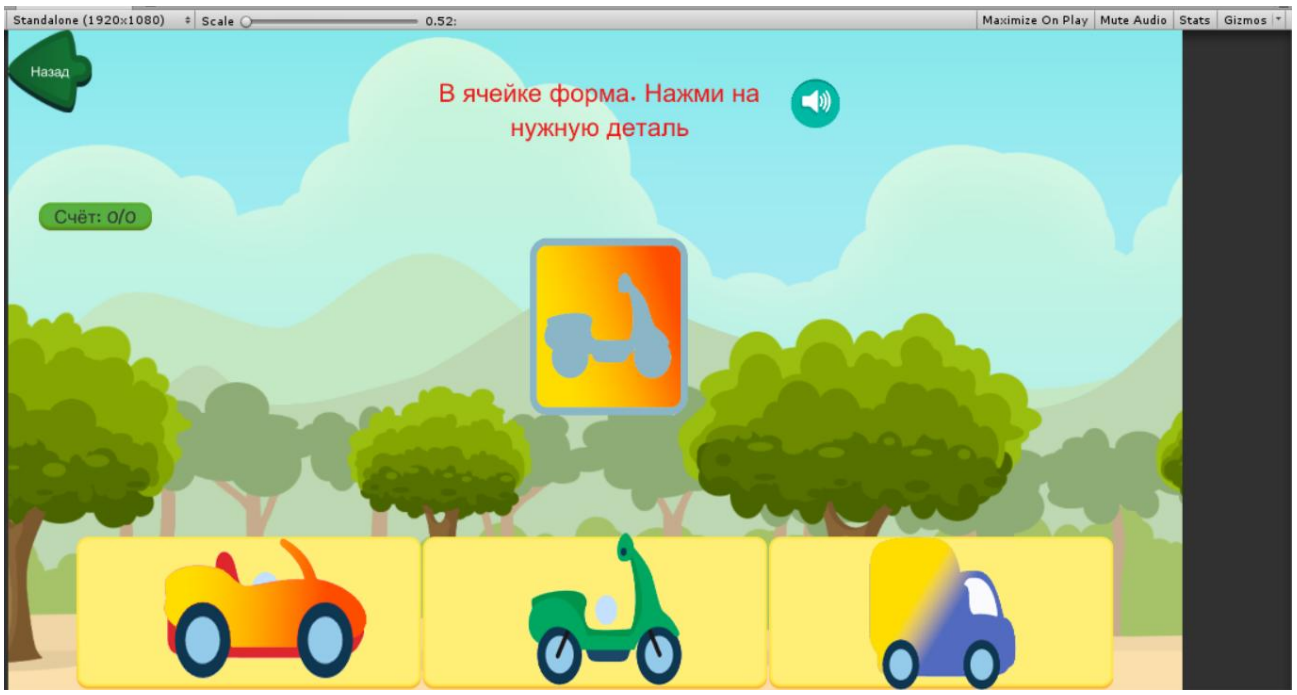

Рисунок 32 – Второй уровень сложности, раздел «Логика»

На третьем уровне сложности игроку предстоит выполнить задания из первого и второго уровней, но к счётчику правильных ответов добавляется время.

2.4.3 Реализация системы заданий по разделу «Пространственное мышление»

Первый уровень сложности по разделу «Пространственное мышление» содержит задание, представленное на рисунке 33. Игроку нужно выбрать, какой объект длиннее.

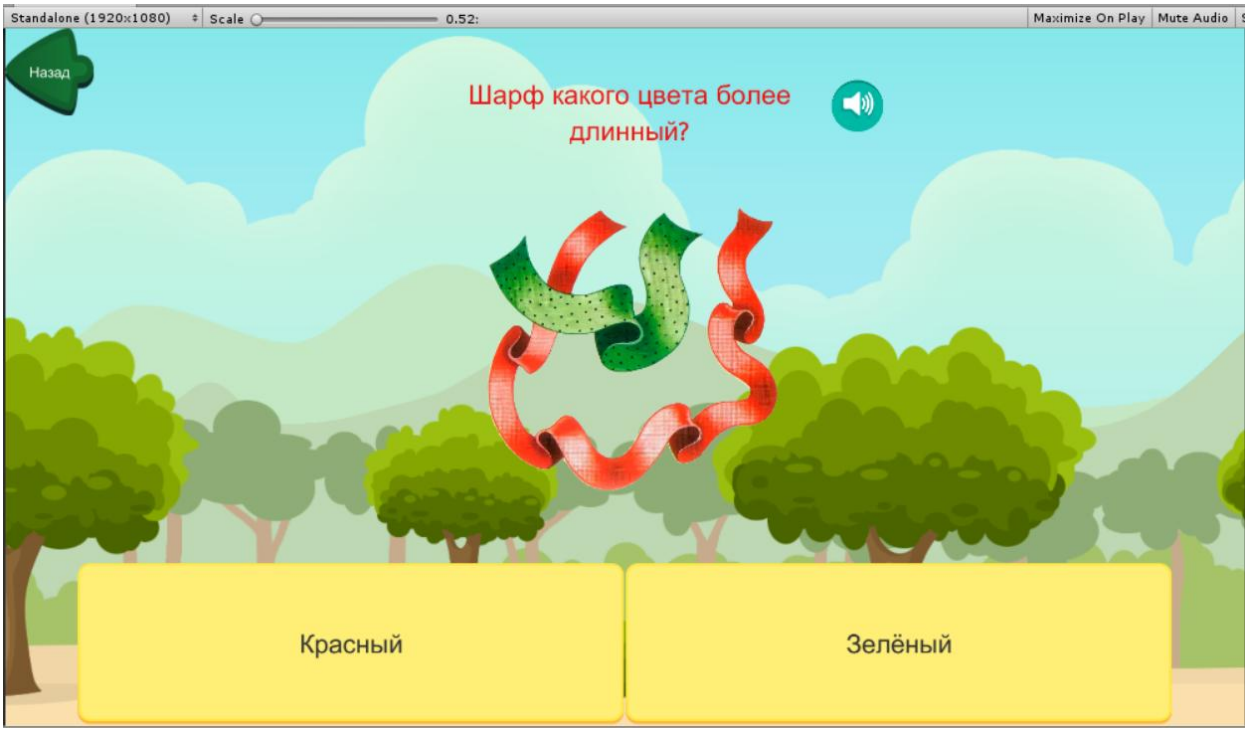

Рисунок 33 – Первый уровень сложности, раздел «Пространственное мышление»

Второй уровень сложности по разделу «Пространственное мышление» предлагает игроку указать вид сверху для объёмной фигуры, как показано на рисунке 34.

Рисунок 34 – Второй уровень сложности, раздел «Пространственное мышление»

Уровень сложности три предполагает собой комбинацию уровней сложности один и два, но добавляется время.

2.4.4 Функциональное тестирование

Функциональное тестирование – это процесс проверки функциональности программного продукта или системы для убеждения в том, что они работают согласно заданным требованиям и ожиданиям. Цель функционального тестирования заключается в проверке, что система выполняет функции, для которых она предназначена, и что она делает это правильно.

Во время функционального тестирования тестируются различные функции продукта, такие как ввод данных, обработка данных, вывод результатов и взаимодействие с другими системами или компонентами. Это может включать тестирование пользовательского интерфейса, работы с базами данных, обработки файлов, выполнения бизнес-логики и других функциональных аспектов. Результаты функционального тестирования игры «Познай себя» описаны в таблице 4.

Таблица 4 – Функциональное тестирование

№	Тест	Действие тестировщика	Ожидаемый результат	Прохождение теста
1	Запуск игры	Запустить игру, первый раз	Запуск сцены с игрой	Да
2	Запуск и выбор различных уровней игры	Запустить различные уровни игры	Запуск всех видов уровней	Да
3	Остановка игры	Нажать на кнопку выхода, для прерывания прохождения	Нажатие на кнопку выведет игрока на главный экран	Да
4	Возврат в главное меню	Нажать на кнопку «назад», что возвращает игрока в главное меню	Нажатие кнопки «назад» вернет игрока в главное меню	Да
5	Перезапуск уровня	Свободно можно начать уровень сначала	Уровень начнётся заново при пере заходе в него	Да

Окончание таблицы 4

№	Тест	Действие тестировщика	Ожидаемый результат	Прохождение теста
6	Прохождение уровня	Пройти уровень	Свободно и без ошибок пройти уровень игры	Да
7	Сохранение прогресса	При выходе из игры, игра сохраняет прогресс за пользователем	При пере заходе в игру можно продолжить свой прогресс	Да
8	Выход из игры	Кнопка выхода из игры, закрывает приложение	Выход из игры	Да

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе было теоретически обоснована и разработана развивающая компьютерная игра «Познай себя» для детей 6–7 лет на платформе Unity.

Для достижения поставленной цели были решены следующие задачи:

1. На основе анализа предметной области, а также учебной и научно-технической литературы были раскрыты теоретические аспекты проектирования развивающих компьютерных игр, на основе которых сформулированы требования к разработанной игре.

2. На основе сравнительного анализа был обоснован выбор средства разработки игры – платформа Unity.

3. Были спроектированы структура развивающей игры, ее интерфейс и сценарий игрового процесса на основе языка моделирования UML.

4. На платформе Unity была разработана компьютерная развивающая игра «Познай себя», осуществлено ее функциональное тестирование, позволяющее оценить качество проделанной работы.

Данное приложение может быть использовано педагогами для проведения внеклассных мероприятий в школе или проведения занятий в подготовительной группе дошкольного образовательного учреждения. Полученный материал исследования может использоваться студентами при написании статей, рефератов, курсовых и выпускных квалификационных работ.

Результаты исследования представлены на следующих научных мероприятиях:

1. VI Всероссийская научно-практическая конференция преподавателей, учителей, студентов и молодых ученых «Актуальные проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 7–12 ноября 2022 г., участие).

2. Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 11 апреля 2023 г., участие).

По результатам исследования опубликованы статьи:

1. Шорохов, Н. В. Анализ средств разработки компьютерных развивающих игр / Н. В. Шорохов // Актуальные проблемы преподавания дисциплин естественнонаучного цикла: тезисы докладов VI Всероссийской научно-практической конференции преподавателей, учителей, студентов и молодых ученых, Лесосибирск, 14–15 ноября 2022 года / Сибирский федеральный университет – Красноярск, 2022. – С. 81–84.

2. Шорохов, Н. В. Особенности разработки развивающих игр для детей младшего школьного возраста / Н. В. Шорохов // Современное педагогическое образование: Теоретический и прикладной аспекты: сборник научных статей II Всероссийский молодёжный научный форум, студентов и молодых ученых, Лесосибирск, 10–15 апреля 2023 года / Сибирский федеральный университет. – Лесосибирск – Красноярск, 2023. – С. 86–90.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алексеев, И. Ю. Разработка развивающих игр для детей на платформе Unity / И. Ю. Алексеев // Молодежная наука как фактор и ресурс инновационного развития : сборник статей II Международной научно-практической конференции.–2020.–С. 94–97. URL:https://www.elibrary.ru/item.asp?id=44537273 (дата обращения: 29.10.2022)

2. Бартон, Д. Р. Игры без риска / Д. Р. Бартон, С. Сьюггеруд. – СПб: Питер, 2019. – 400 с.

3. Богуславская, З. М. Развивающие игры для детей дошкольного возраста / З. М. Богуславская, Е. О. Смирнова. – Москва : Просвещение, 2018. – 143 с.

4. Васильев, А. В. Термин «компьютерная игра»: опыт междисциплинарного анализа / А. В. Васильев, Ю. В. Печатнов // Пролог: журнал о праве. – 2021. – №2. – С. 131–138. – URL: https://clck.ru/34mTEK (дата обращения: 20.11.2022).

5. Винокуров, Д. А. Разработка развивающей мобильной игры на платформе Unity / Д. А. Винокуров // Наука молодых - наука будущего. – 2023. – URL: https://www.elibrary.ru/item.asp?id=53950874 (дата обращения: 09.11.2022)

6. Гейг, Майк. Разработка игр на Unity за 24 часа / Майк Гейг. – Москва : Бомбора, 2020. – 466 с.

7. Гогоберидзе, А. Г. Теория и методика воспитания детей дошкольного возраста / А. Г. Гогоберидзе. – М.: Издательский центр «Академия», 2005. – 320 с.

8. Гундольф, Фрейермут С. Игры. Геймдизайн. Исследование игр / Фрейермут С. Гундольф. – Москва : Гуманитарный центр, 2021. – 250 с. – ISBN 978-617-7528-04-2.

9. Денисов, Д. В. Разработка игры на Unity. С нуля и до реализации / Д. В. Денисов. – Москва : Эксмо, 2021. – 180 с. – ISBN 978-5-532-94186-1.

10. Использование Unity для разработки приложений // Android tools : [сайт]. – 2018. – 07 апр. – URL: https://clck.ru/34mSxm (дата обращения:10.09.2022).

11. Кокче, С. М. Передовой опыт работы: «Интеллектуальное развитие детей посредством развивающих игр» / С. М. Кокче // Муниципальное бюджетное дошкольное образовательное учреждение детский сад № 35 муниципального образования Темрюкский район. – 15.08.2014 / URL: https://clck.ru/34k8em (дата обращения: 10.11.2022)

12. Ларкович, С. Н. Unity на практике. Создаем 3D-игры и 3D-миры / С. Н. Ларкович. – Россия : Наука и техника, 2022. – 384 с. – ISBN 978 07592 02-5.

13. Ларман, Крэг. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / Крэг Ларман. – М. : Вильямс, 2020. – 736 с.

14. Леоненков, А. В. Самоучитель UML / А. В. Леоненков. – СПб. : БХВ-Петербург, 2019. – 432 с. ISBN 5-94157-342-1.

15. Лыскова В. Ю. Логика в информатике / В. Ю. Лыскова, Е. А. Ракитина. – Москва : Лаборатория Базовых Знаний, 2004. – 160 с. – ISBN 5-93208-105-8 дата обращения: 09.11.2022)

16. Мэннинг, Д. Unity и для разработчика / Д. Мэннинг. – Санкт-Петербург: Питер, 2018. – 352 с.

17. Пайлон, Д. UML 2 для программистов / Д. Пайлон. – М. : Питер, 2018. – 797 с.

18. Платов, В. Я. Деловые игры: разработка, организация, проведение / В. Я. Платов. – Москва : Профиздат, 1991. – 192 с. – ISBN 5-255-00129-5.

19. Попов, А. А. Воспитание доброжелательности у детей 6–7 лет в процессе социо-игры / А. А. Попов, Е.В. Горшков, А.З Асроров // Тенденции развития науки и образования : сборник статей II Международной научно-практической конференции. – 2019. – С. 8-13. – URL: https://www.elibrary.ru/item.asp?id=50106522 (дата обращения: 09.11.2022)

20. Рябова, Т. В. Воспитание доброжелательности у детей 6–7 лет в процессе социо-игры / Т. В. Рябова // Молодежная наука как фактор и ресурс инновационного развития : сборник статей II Международной научно-практической конференции. – 2023. – С. 99–110. – URL: https://www.elibrary.ru/item.asp?id=50106522%20 (дата обращения: 09.11.2022)

21. Сергеев, Е. С. Разработка профориентационной vr-игры на платформе unity / Е. С. Сергеев, А. Е. Сухова, И. С Максимов, Н. А. Сенаторов // Научное обозрение. Технические Науки: сборник статей II Международной научно-практической конференции. – 2021. – С. 38–42. – URL: https://www.elibrary.ru/item.asp?id=45691799 (дата обращения: 09.11.2022)

22. Торн, А. Основы анимации в Unity : пер. с англ. Р. Рагимова / А. Торн. – М.: ДМК Пресс, 2016.– 176 с. – ISBN 978-5-97060-377-2.

23. Устав (Конституция) Всемирной организации здравоохранения: принят в г. Нью-Йорке 22 июля 1946 г. // Международные организации: материалы и док. / Сиб. ин-т междунар. отношений и регионоведения; сост.: О. В. Плотникова, Ю. И. Дубровин, В. С. Плотников. – Новосибирск, 2004. – С. 121–131.

24. Хорхе, Паласиос. Unity 5.x. Программирование искусственного интеллекта в играх. Руководство / Паласиос Хорхе. – М. : ДМК Пресс, 2017. – 427 с.

25. Что такое игровой движок? – Режим доступа: https://clck.ru/bmKfN (дата обращения: 24.11.2022).

26. Adobe Photoshop: официальный сайт / Copyright, 2023 – . – / URL:https://clck.ru/FTqJa (дата обращения: 15.09.2022).

27. Audiocity: официальный сайт / Copyright, 2023 – . – / URL: https://www.audacityteam.org/download/ (дата обращения: 15.09.2022).

28. C# Programming Guide. URL: https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx (дата обращения: 15.08.2022).

29. CorelDraw: официальный сайт / Copyright, 2023 – . – / URL: https://www.coreldraw.com/en/pages/free-download/ (дата обращения: 15.09.2022).

30. CryEngine: официальный сайт / Crytek GmbH, 2023– . – / URL: https://www.cryengine.com/download (дата обращения: 15.09.2022).

31. GIMP: официальный сайт / Creative Commons, 2023 – . – / URL: https://www.gimp.org/downloads/ (дата обращения: 15.09.2022).

32. Godot: официальный сайт / Software Freedom Conservancy, 2023 – . – / URL: https://godotengine.org/download/windows/ (дата обращения: 15.09.2022).

33. Inkscape: официальный сайт / Creative Commons, 2023 – . – / URL: https://inkscape.org/ru/release/all/windows/ (дата обращения: 15.09.2022).

34. Unity : URL: https://ru.wikipedia.org/wiki/Unity (дата обращения: 09.01.2023).

35. Unity : официальный сайт / Unity Technologies, 2023 – . – URL: https://unity.com/download (дата обращения: 15.09.2022).

36. Unity 5 tutorial for beginners: 2D Platformer - Moving Platform, YouTube URL: Режим па: https://www.youtube.com/watch?v=4R_AdDK25kQ (дата обращения:02.02.2023).

37. Unity в действии. Мультиплатформенная разработка на C# : Издательский дом. – Санкт-Петербург [и др.] / Д. Хокинг; под редакцией С. М. Черников. – Санкт-Петербург, 2019. – 351 с. – ISBN 978-5-4461-0816-9.

38. Unity и C#. Геймдев от идеи до реализации / Д. Г. Бонд. – Питер : ООО Издательство «Питер», 2019. – 928 с.

39. Unreal Engine: официальный сайт / Epic Games, 2004–2023 URL: https://www.unrealengine.com/en-US/download (Дата обращения:09.08.2022)

40. Visual Studio Code : официальный сайт / Microsoft, 2023 – . – URL: https://code.visualstudio.com/ (дата обращения: 15.09.2022).

41. Will Goldstone, W. G. Основы разработки игр на Unity / W. G. Will Goldstone. – США : Packt Publishing Ltd, 2019. – 298 с. – ISBN 9781847198198.

ПРИЛОЖЕНИЕ А

Листинг C# кода, создание публичных переменных классов кнопок

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class NumbersVoice : MonoBehaviour {
    public AudioSource source;
    public AudioClip one;
    public AudioClip two;
    public AudioClip three;
    public AudioClip four;
    public AudioClip five;
    public AudioClip six;
    public AudioClip seven;
    public AudioClip eight;
    public AudioClip nine;
    public AudioClip ten;
    public Text scoreText;
    public Button button;
    public void playSound()
    {
        string info = button.GetComponentInChildren<Text>().text;
        scoreText.text = info;
        switch (info)
        {
            case «Один»:
                source.clip = one;
                break;
            case «Два»:
```

```
            source.clip = two;
            break;
        case «Три»:
            source.clip = three;
            break;
        case «Четыре»:
            source.clip = four;
            break;
        case «Пять»:
            source.clip = five;
            break;
        case «Шесть»:
            source.clip = six;
            break;
        case «Семь»:
            source.clip = seven;
            break;
        case «Восемь»:
            source.clip = eight;
            break;
        case «Девять»:
            source.clip = nine;
            break;
        case «Десять»:
            source.clip = ten;
            break;
    }
    source.Play();
  }
}
```

ПРИЛОЖЕНИЕ Б

Листинг C# кода, создание уровней два и три

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

public class QuizGameLevel3 : MonoBehaviour
{

    public Button[] answerButtons = new Button[4];
    public Text questionText;

    public int[] answers = new int[4];
    public string[] answersInEnglish = new string[4];
    private int questionsDone = 0;
    int correctAnswerLocation;
    List<int> questionsUsed = new List<int>();

    public GameObject[] QuestionPanels;
    public GameObject finalResultPanel;
    public Text resultText;
    private int score = 0;
    private bool gameActive = true;
    public GameObject feedbackText;
    public Text scoreText;
    public Button questionButton;
    public Sprite one;
    public Sprite two;
    public Sprite three;
    public Sprite four;
    public Sprite five;
    public Sprite six;
    public Sprite seven;
    public Sprite eight;
    public Sprite nine;
    public Sprite ten;
```

```
public GameObject feedbackButton;
public Sprite wrong;
public Sprite correct;

int timeLeft;
public Text timeText;
public Button timeButton;

public IEnumerator StartCountdown(int countdownValue)
{
   timeLeft = countdownValue;
   byte a = 44;
   int aInt = a;
   byte b = 255;
   int bInt = b;
   while (timeLeft > -1)
   {

      timeText.text = «Время: « + timeLeft;
      timeButton.GetComponent<Image>().color = new Color32(a, b, 52, 255); ;
      yield return new WaitForSeconds(1.0f);
      timeLeft--;

      int halfTime = 15;
      if (timeLeft >= halfTime) {

         aInt = aInt + 14;
         a = (byte)aInt;

      } else  {

         bInt = bInt - 14;
         b = (byte)bInt;

      }

   }
   gameActive = false;
   foreach (GameObject p in QuestionPanels)
   {
      p.SetActive(false);
   }
   finalResultPanel.SetActive(true);
   DisplayResults();
}
```

```
// Use this for initialization
void Start()
{
    score = 0;
    questionsDone = 0;
    newQuestion();
    scoreText.text = «Счёт: 0/0»;
    StartCoroutine(StartCountdown(30));

}
//Generating question
void newQuestion()
{
    int numberAsked = UnityEngine.Random.Range(1, 11);
    while (questionsUsed.Contains(numberAsked))
    {
        numberAsked = UnityEngine.Random.Range(1, 11);
    }
    switch (numberAsked)
    {
        case 1:
            questionButton.GetComponent<Image>().overrideSprite = one;
            break;
        case 2:
            questionButton.GetComponent<Image>().overrideSprite = two;
            break;
        case 3:
            questionButton.GetComponent<Image>().overrideSprite = three;
            break;
        case 4:
            questionButton.GetComponent<Image>().overrideSprite = four;
            break;
        case 5:
            questionButton.GetComponent<Image>().overrideSprite = five;
            break;
        case 6:
            questionButton.GetComponent<Image>().overrideSprite = six;
            break;
        case 7:
            questionButton.GetComponent<Image>().overrideSprite = seven;
            break;
        case 8:
            questionButton.GetComponent<Image>().overrideSprite = eight;
            break;
```

```
            case 9:
                questionButton.GetComponent<Image>().overrideSprite = nine;
                break;
            case 10:
                questionButton.GetComponent<Image>().overrideSprite = ten;
                break;

        }
        questionsUsed.Add(numberAsked);
        correctAnswerLocation = UnityEngine.Random.Range(0, 4);


        for (int i = 0; i < answers.Length; i++)
        {
            if (i == correctAnswerLocation)
            {
                answers[i] = numberAsked;
            }
            else
            {
                int wrongAnswer = UnityEngine.Random.Range(1, 11);
                while (wrongAnswer == numberAsked || Array.IndexOf(answers,
wrongAnswer) > -1)
                {
                    wrongAnswer = UnityEngine.Random.Range(1, 11);
                }
                answers[i] = wrongAnswer;
            }
        }

        for (int i = 0; i < answers.Length; i++)
        {
            switch (answers[i])
            {
                case 1:
                    answersInEnglish[i] = «Один»;
                    break;
                case 2:
                    answersInEnglish[i] = «Два»;
                    break;
                case 3:
                    answersInEnglish[i] = «Три»;
                    break;
                case 4:
                    answersInEnglish[i] = «Четыре»;
```

```
                break;
            case 5:
                answersInEnglish[i] = «Пять»;
                break;
            case 6:
                answersInEnglish[i] = «Шесть»;
                break;
            case 7:
                answersInEnglish[i] = «Семь»;
                break;
            case 8:
                answersInEnglish[i] = «Восемь»;
                break;
            case 9:
                answersInEnglish[i] = «Девять»;
                break;
            case 10:
                answersInEnglish[i] = «Десять»;
                break;

        }
    }
    questionText.text = numberAsked.ToString();
    for (int i = 0; i < answerButtons.Length; i++)
    {
        answerButtons[i].GetComponentInChildren<Text>().text =
answersInEnglish[i];
    }

}

//Checking answer and giving feedback
public void checkAnswer(int buttonNumber)
{
    if (gameActive)
    {
        if (buttonNumber == correctAnswerLocation)
        {
            score++;
            feedbackButton.GetComponent<Image>().overrideSprite = correct;
            feedbackButton.GetComponent<Image>().color = new Color32(33, 171, 0,
255);
            answerButtons[buttonNumber].GetComponent<Image>().color =
Color.green;
```

```
            print(«Правильно»);

        }
        else
        {
            answerButtons[buttonNumber].GetComponent<Image>().color =
Color.red;
            answerButtons[correctAnswerLocation].GetComponent<Image>().color =
Color.green;
            print(«Неправильно»);


            feedbackButton.GetComponent<Image>().overrideSprite = wrong;
            feedbackButton.GetComponent<Image>().color = Color.red;
        }
        StartCoroutine(«ContinueAfterFeedback»);
    }
  }
  //pause before moving to next question, showing feedback, updating score
  IEnumerator ContinueAfterFeedback()
  {
    gameActive = false;
    feedbackButton.SetActive(true);
    //feedbackText.SetActive(true);
    yield return new WaitForSeconds(1.7f);
    for (int i = 0; i < 4; i++)
    {

      answerButtons[i].GetComponent<Image>().color = new Color32(254, 238,
220, 255);
    }
    //feedbackText.SetActive(false);
    feedbackButton.SetActive(false);
    questionsDone++;
    scoreText.text = «Счёт: « + score.ToString() + «/» + questionsDone.ToString();

    newQuestion();


    gameActive = true;
  }

  //displaying final result
  void DisplayResults()
  {
```

```
    resultText.text = «Счёт: « + score;
}



//restart level
public void restartLevel()
{
    SceneManager.LoadScene(«NumbersGameLevel3»);
}


public void QuitGame()
{

    SceneManager.LoadScene(«NumbersLearning»);

}
     }
```

ПРИЛОЖЕНИЕ В

Листинг C# кода, адаптация экрана

```
using UnityEngine;

namespace Code.Scripts
{
        public class expansion : MonoBehaviour
        {
                [SerializeField] private FullScreenMode _screenMode =
FullScreenMode.MaximizedWindow;

       private void Awake()
                {
                        int width = 1920;
                        int height = 1080;

          Screen.SetResolution(width, height, _screenMode);
                }
        }
                                                        }
```