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
Храмова Л.Н. Храмова
подпись инициалы, фамилия
« 14 » 06 2024 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии
код-наименование направления

РАЗРАБОТКА КОМПЬЮТЕРНОЙ 3D ИГРЫ ДЛЯ РАЗВИТИЯ ПРОСТРАНСТВЕННОГО МЫШЛЕНИЯ

Руководитель 14.06.24 доцент, канд. пед. наук А.В. Фирер
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник 14.06.24 В.Д. Русов
подпись, дата инициалы, фамилия

Нормоконтролер 14.06.24 А.В. Фирер
подпись, дата инициалы, фамилия

Лесосибирск 2024

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка компьютерной 3D игры, направленной на развитие пространственного мышления» содержит 76 страниц, 41 использованный источник, 2 таблицы, 28 иллюстраций, 8 приложений.

КОМПЬЮТЕРНЫЕ ИГРЫ, ГОЛОВОЛОМКА, ЛАБИРИНТ, ПРОСТРАНСТВЕННОЕ МЫШЛЕНИЕ, UNITY.

Компьютерные игры стали актуальной темой в современной науке, являясь продуктом технологического прогресса и инструментом для развития умственных способностей человека, в частности пространственного мышления.

Цель исследования – теоретически обосновать и разработать компьютерную 3D игру, направленную на развитие пространственного мышления для детей 6–7 лет.

Объект исследования – процесс разработки компьютерной игры.

Предмет исследования – процесс проектирования и разработки компьютерной 3D игры, направленной на развитие пространственного мышления для детей 6–7 лет.

Основные задачи исследования:

– на основе анализа научно-технической литературы и существующих компьютерных игр для развития пространственного мышления разработать концепцию и модель игры, а также выявить требования к ней;

– на основе анализа инструментальных средств выбрать платформу и среду разработки 3D игры;

– разработать и протестировать компьютерную 3D игру для развития пространственного мышления детей 6–7 лет.

В результате выпускной квалификационной работы была разработана компьютерная 3D игра для детей 6–7 лет, направленная на развитие пространственного мышления

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире информационные технологии играют ключевую роль в процессе информатизации общества и образования, включая сферу разработки компьютерных игр. Эти игры приобрели огромную популярность среди людей всех возрастов, и количество игроков постоянно растет. Пользователи могут выбирать из множества жанров, которые насчитывают сотни вариантов, благодаря разнообразию интересов людей.

Компьютерные игры стали актуальной темой в современной науке, являясь продуктом технологического прогресса и инструментом для развития умственных способностей человека. Они позволяют визуализировать различные жизненные ситуации и способы их преодоления, способствуя формированию личности и культуры общества. Игры, направленные на развитие пространственного мышления, в частности лабиринты, особенно полезны для тренировки ориентации в пространстве. Создание игр в этом жанре делает акцент на этих навыках, помогая игрокам улучшить их умственные способности и развить пространственное мышление.

Цель исследования – теоретически обосновать и разработать компьютерную 3D игру, направленную на развитие пространственного мышления для детей 6–7 лет.

Объект исследования – процесс разработки компьютерной игры.

Предмет исследования – процесс проектирования и разработки компьютерной 3D игры, направленной на развитие пространственного мышления для детей 6–7 лет.

Основные задачи исследования:

– на основе анализа научно-технической литературы и существующих компьютерных игр для развития пространственного мышления разработать концепцию и модель игры, а также выявить требования к ней;

– на основе анализа инструментальных средств выбрать платформу и среду разработки 3D игры;

– разработать и протестировать компьютерную 3D игру для развития пространственного мышления детей 6–7 лет.

Методы исследования включают в себя:

– теоретические: анализ учебной и научно-технической литературы, обобщение и сравнительный анализ.

– эмпирические: моделирование и тестирование программного продукта.

Результаты исследования были представлены на следующих научных мероприятиях:

1. VII Всероссийская научно-практической конференция «Актуальные проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 1-2 ноября 2023 г.).

2. III Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 9 апреля 2024 г.).

По результатам исследования принята к публикации следующая статья: Русов В. Д. / Разработка компьютерной 3D игры для развития пространственного мышления / В. Д. Русов // Информационные технологии в образовании, науке и производстве: материалы III Всероссийского молодежного научного форума «Современное педагогическое образование: теоретический и прикладной аспекты» / ЛПИ – филиал СФУ. – Лесосибирск, 2024.

Структура работы – работа состоит из введения, двух глав, заключения, списка использованных источников, включающего 41 наименование, и 8 приложений. Результаты работы представлены в 2 таблицах, 28 иллюстрациях. Общий объем работы – 76 печатных листов.

1 Теоретические аспекты проектирования компьютерной игры для развития пространственного мышления

1.1 Анализ существующих компьютерных игр для развития пространственного мышления

Пространственное мышление является критически важным когнитивным навыком, который включает в себя способность визуализировать и манипулировать объектами, ориентироваться в пространстве, используя проработанные навыки в уме. «Развитие пространственного мышления происходит в течение всей жизни, но наиболее интенсивно оно развивается у детей от 4 до 7 лет» [27]. Дети начинают осваивать базовые пространственные навыки через игры и взаимодействие с окружающим миром. Пазлы, конструкторы, рисование и другие активности способствуют развитию этих навыков. В наше время дети от 5 лет уже знакомы с компьютерными играми, отсюда берут начало исследования, посвященные воздействию компьютерных игр на развитие пространственного мышления.

Существуют различные игры, направленные на развитие этого навыка, каждая из которых предлагает уникальный подход и методологию. Проведем анализ существующих игр, предназначенных для развития пространственного мышления, с акцентом на их механики, целевую аудиторию и эффективность.

1) Portal и Portal 2. Portal и его сиквел Portal 2 от Valve Corporation являются одними из самых известных игр, направленных на развитие пространственного мышления. Игрокам нужно использовать устройство, создающее порталы, чтобы перемещаться по уровням и решать головоломки. Эффективность данной серии игр заключается в том, что они эффективно развивают навыки пространственного мышления, требуя от игроков думать о

взаимосвязях между различными пространственными точками и использовать их для достижения цели.

2) Tetris. Это одна из самых известных и классических головоломок, разработанная Алексеем Пажитновым. Игроки должны размещать падающие блоки различных форм так, чтобы они заполнили горизонтальные линии без пробелов. Эффективность данной игры заключается в том, что Tetris развивает навыки пространственного мышления, требуя от игроков быстро анализировать формы и находить оптимальные места для их размещения.

3) Minecraft. Это своеобразная игра-песочница, позволяющая игрокам строить и исследовать открытый мир, состоящий из блоков. Эффективность данной игры заключается в том, что Minecraft стимулирует пространственное мышление, позволяя игрокам визуализировать и строить сложные структуры, а также ориентироваться в большом открытом мире.

Данные игры, как и многие другие для развития пространственного мышления предлагают разнообразные подходы и механики. Они варьируются от манипуляции формами и объектами (Tetris, Portal) до исследования и строительства в открытых мирах (Minecraft). Эти игры доказали свою эффективность в развитии когнитивных навыков, предоставляя игрокам увлекательные и образовательные возможности.

Хотя многие из представленных игр эффективно способствуют развитию пространственного мышления, но их основная цель зачастую заключается в предоставлении развлекательного опыта. Следует отметить, что механики и задачи в этих играх не всегда оптимизированы для систематического и целенаправленного развития пространственного мышления.

На развитие пространственного мышления также направлены и интерактивные упражнения образовательной онлайн платформы ЛогикЛайк [15]. Платформа предлагает различные задачи и головоломки, которые помогают пользователям различных возрастных групп развивать внимание, логику, математические способности, 3D мышление и т. д. Данная

платформа зарекомендовала себя, как образовательный интерактивный онлайн ресурс, содержащий увлекательные задания и игры. Однако, для детей 6-7 лет предложены только задания в двухмерной проекции (например, графические диктанты, которые можно распечатать, и развивающая игра «Мой мир»).

Всё выше сказанное указывает на необходимость разработки специализированных игр, в которых все элементы дизайна и игровая механика будут направлены на развитие пространственного мышления.

Для того, чтобы определить в каком жанре будет разработана игра, нужно определить понятие компьютерной игры.

Подвальный М. А. [22] считает, что «Видеоигра (компьютерная игра), в широком смысле – любая игра, для функционирования которой необходима электронно-вычислительная машина: персональный компьютер, телевизионная приставка, портативная игровая система, игровой автомат, смартфон, планшет и т. д.».

Анализ источников по теме исследования [8; 23; 27] позволил выделить следующие характеристики компьютерных игр:

– интерактивность: игрок взаимодействует с игровым миром, и его действия непосредственно влияют на ход игры;

– виртуальность: игровой мир создан и не существует в реальности;

– поставленная цель: у игры есть цель или задачи, которые игрок должен выполнить;

– воспроизводимость: в игру можно играть многократно, при этом каждый раз результат может быть разным.

Компьютерные игры выполняют разнообразные функции:

– развлекательная: игры позволяют людям расслабиться, получить удовольствие и отвлечься от повседневных забот;

– образовательная: игры могут использоваться для обучения и развития различных навыков, таких как логическое мышление, память, внимание, пространственное мышление;

– тренировочная: игры могут использоваться для тренировки различных навыков, таких как реакция, координация движений, скорость принятия решений;

– социальная: игры могут использоваться для общения и взаимодействия с другими людьми, способствуя развитию социальных навыков и установлению новых связей;

– профессиональная: игры могут использоваться для подготовки специалистов в различных областях, например, военных, летчиков, диспетчеров и т. д. Они помогают моделировать сложные ситуации и развивать профессиональные навыки в безопасной и контролируемой среде.

Анализ источников [1, 2; 6] позволил выделить следующие виды компьютерных игр, представленных в таблице 1:

Таблица 1 – Виды компьютерных игры

Жанр	**Описание**	**Основные поджанры**
Экшен (Action)	Игры, требующие быстрой реакции и координации движений	Шутеры (FPS/TPS), Платформеры, Файтинги,
Приключения (Adventure)	Игры, сосредоточенные на исследовании	Графические квесты, Интерактивные фильмы
Ролевые игры (RPG)	Игры, где игрок управляет персонажем, развивая его способности и историю	Японские RPG (JRPG), Западные RPG (WRPG), MMO
Стратегии (Strategy)	Игры, требующие тактического и стратегического мышления	Реального времени (RTS), Пошаговые (TBS)
Спортивные (Sports)	Игры, имитирующие различные виды спорта	Футбольные, Баскетбольные, Гонки
Симуляторы (Simulation)	Игры, имитирующие реальные или вымышленные сценарии	Градостроительные, Полетные, Жизненные, Спортивные
Головоломки (Puzzle)	Игры, где внимание уделяется решению задач и головоломок	Логические, Лабиринты, Математические

Анализируя данную таблицу, можно сделать вывод, что жанр головоломки (puzzle) более целенаправленно способствует развитию пространственного мышления, особенно поджанр лабиринт, который может быть использован в качестве инструмента для развития пространственного мышления, улучшения концентрации, повышения уровня внимания и представления различных объектов, путем переноса их из 2D в 3D проекции.

Жанр головоломок в компьютерных играх прошел длинный путь развития, от простых логических задач до сложных, многоуровневых испытаний, которые требуют глубокого анализа и креативности. Этот процесс можно разделить на несколько ключевых этапов.

В начале своего пути головоломки в компьютерных играх представляли собой простые логические задачи. В 1980 году японский разработчик Хироюки Имабаяси создал игру Sokoban, где игрок должен был передвигать ящики по лабиринту, направляя их в определенные места. Эта игра стала одной из первых, показавших потенциал головоломок в цифровом формате.

В 1985 году советский программист Алексей Пажитнов разработал Тетрис, игру, которая быстро стала глобальным феноменом. Тетрис, с его простыми, но увлекательными механиками, продемонстрировал, что головоломки могут быть не только сложными, но и чрезвычайно затягивающими.

В 1990-е годы головоломки начали эволюционировать, предлагая игрокам более разнообразные и сложные задачи. В 1993 году вышла игра Myst, разработанная братьями Робином и Рэндом Миллерами. Эта игра сочетала в себе элементы приключений и головоломок, предлагая игрокам исследовать таинственный остров и решать головоломки для продвижения сюжета. Myst стала одной из самых продаваемых компьютерных игр своего времени и положила начало новому поджанру — приключенческим головоломкам.

С развитием технологий головоломки начали интегрироваться в другие жанры, создавая уникальные гибриды. Примером такой интеграции является игра Portal от компании Valve. Portal сочетала в себе элементы головоломок и шутеров от первого лица. Игрокам предлагалось использовать портальную пушку для создания пространственных дыр и решения головоломок, основанных на физике. Эта игра получила высокие оценки за инновационный подход и оригинальность.

С появлением смартфонов и планшетов в 2010-х годах жанр головоломок получил новый импульс для развития. Мобильные платформы предоставили разработчикам возможность создавать игры, в которые можно было играть в короткие сессии, делая головоломки доступными для широкой аудитории. Примером успешной мобильной головоломки является Angry Birds (2009 год), которая предложила игрокам разрушать структуры с помощью птичек, стреляя ими из рогатки.

Сегодня жанр головоломок продолжает развиваться, предлагая игрокам всё более инновационные и сложные задачи. Современные головоломки часто используют возможности виртуальной и дополненной реальности, создавая более глубокие и интерактивные игровые миры. Игры, такие как The Witness, предлагают игрокам исследования и решение сложных головоломок в открытом мире, сочетая в себе элементы классических логических игр и современных технологий.

Исследования показывают, что регулярное прохождение лабиринтов, относящихся к жанру головоломок, способствует активации определенных участков мозга, что приводит к развитию навыка пространственного мышления и других когнитивных способностей.

Разрабатываемая игра направлена на развитие пространственного мышления, что делает её ценным инструментом для образовательных целей. Образовательные игры объединяют элементы обучения и развлечения, создавая интерактивный опыт, который способствует освоению определенных навыков. В свою очередь, игры-головоломки, требуют от

игроков решения различных задач, связанных с логическим мышлением, стратегией и креативностью. Лабиринт идеально вписывается в этот жанр, так как прохождение лабиринта требует ориентации в пространстве, планирования маршрута и принятия решений на каждом этапе.

Подводя итог анализа существующих компьютерных игр для развития пространственного мышления, было принято решение разработать компьютерную 3D игру в жанре головоломка и её поджанра лабиринт.

Проектирование и разработку компьютерной 3D игры для развития пространственного мышления будет осуществляться в соответствии с алгоритмом, который состоит из следующих этапов:

1. Определение концепции игры:
   – определение жанра, стиля и основной механики игры;
   – создание базового сценария компьютерной игры, моделирование.
2. Исследование и планирование:
   – оценка технических возможностей и выбор инструментов разработки (движок, язык программирования).
3. Разработка компьютерной 3D игры для развития пространственного мышления:
   – подготовка среды разработки;
   – моделирование 3D объектов и текстурирование;
   – написание кода для основных механик игры (движение, взаимодействие с объектами, работа камеры);
   – разработка логики лабиринта;
   – разработка пользовательского интерфейса (UI);
   – интеграция звука в игру;
4. Тестирование
   – тестирование отдельных модулей и компонентов на корректность работы;
   – проверка взаимодействия между различными частями игры;

– внесение исправлений и оптимизация кода (по надобности).

Создание альфа-версии компьютерной 3D игры для развития пространственного мышления (лабиринт) является ключевым этапом, который позволяет убедиться в работоспособности основных игровых механик и выявить критические проблемы на ранней стадии. Этот процесс требует тщательного планирования, проектирования и разработки, а также тесного взаимодействия между различными участниками команды. Важным аспектом является проведение внутреннего тестирования и получение обратной связи, что позволяет внести необходимые корректировки и подготовить проект к следующим этапам разработки. Альфа-версия служит фундаментом для дальнейшего совершенствования игры, обеспечивая надежную основу для достижения конечной цели — создания увлекательного и полезного продукта.

1.2 Разработка концепции и требований к компьютерной игре

Отбор целевой аудитории является важным этапом разработки любой компьютерной игры и влияет на требования к ней. Целевая аудитория — группа людей, которые будут заинтересованы в продвижении игры, её геймплее, целях и функциональности. Определение целевой аудитории помогает разработчикам точнее сфокусироваться на нужных механиках игры, что поможет повысить её популярность среди выбранного сегмента игроков, а также исключит попадание нецелесообразного контента в саму игру.

Для разработки компьютерной 3D игры для развития пространственного мышления была определена целевая аудитория – дети 6–7 лет. Ведь в данном возрасте дети активно интересуются окружающим их миром, активно учатся новому и стремятся к исследованию неизвестного. Дети в этом возрасте уже способны решать простые логические задачи, а

моторика рук уже позволяет управлять компьютерной мышью и управлять персонажем с помощью клавиатуры. Они способны концентрироваться на поставленных задачах не так долго, как взрослые, поэтому разрабатываемая игра должна заинтересовать их и задержать в ней с целью выработки навыков пространственного мышления.

Также важно отметить особенности игр, направленных на развитие пространственного мышления, при выборе целевой аудитории. В данных играх часто приходится проводить анализ местности, в которую поместили игрока, от него требуется запоминать всю зримую информацию и составлять план дальнейших действий для продвижения по данной ему локации.

Необходимо выбрать платформу, на которой будет разрабатываться игра, которая позволит создавать игры с обширными мирами и графическими эффектами, при этом не будет нагружать восприятие ребенка.

Стиль игры должен включать в себя визуальное оформление и звуковое сопровождение. Для данной игры, с учетом характеристики целевой аудитории, выбран минималистический и интуитивно понятный стиль, который способствует фокусировке на задачах и улучшению когнитивных навыков игрока.

За визуальное оформление будет отвечать следующее:

– Графика. Игра будет иметь 3D-графику с упрощенными моделями и текстурами. Такой подход позволяет не перегружать внимание игрока лишними деталями и концентрироваться на решении задач. Цветовая палитра будет сбалансированной и ненавязчивой, с использованием пастельных тонов для снижения утомляемости глаз.

– Интерфейс. Пользовательский интерфейс (UI) будет минималистичным, с четкими и понятными элементами управления. Важные элементы, такие как мини-карта и время будут размещены так, чтобы не мешать обзору игрового пространства.

Общая атмосфера игры будет нацелена на создание условий для комфортного и увлекательного обучения. Лабиринты будут располагаться в различных тематических зонах, похожих на офис или институт, что добавит разнообразия и интереса к игровому процессу.

С каждым новым уровнем будет увеличиваться сложность их прохождения, за счет сокращения времени и увеличения площади уровня. График сложности уровней представлен на рисунке 1.

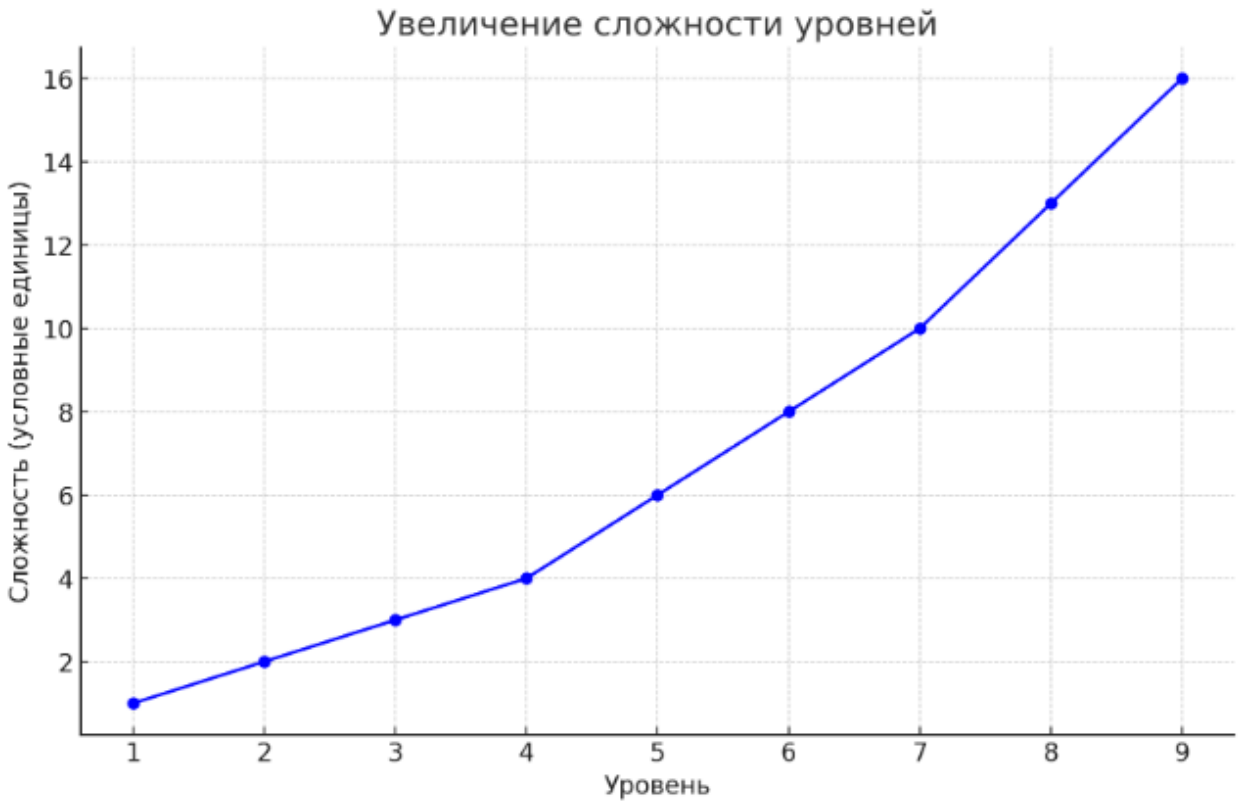

Рисунок 1 – График сложности уровней

На данном графике видно, что увеличение уровня сложности вводит игрока в постепенно увеличивающиеся трудности без сильных перепадов. Сложность определяется размером лабиринта, временем его прохождения и наличием препятствий.

1.3 Моделирование компьютерной 3D игры для развития пространственного мышления

Unified Modeling Language (UML) — это стандартный язык визуального моделирования, который используется для спецификации, визуализации, разработки и документирования программных систем. Он был разработан для стандартизации методов создания диаграмм в объектно-ориентированном программировании и для того, чтобы помочь разработчикам систем и приложений четко представлять структуру и поведение программного обеспечения.

В своей работе Крег Ларман [14] отмечает, что UML диаграммы последовательности являются одним из видов диаграмм, используемых в разработке программного обеспечения для визуализации взаимодействия между объектами или компонентами системы во времени. Эти диаграммы особенно полезны для детального представления потоков сообщений и событий, происходящих между участниками системы, что помогает разработчикам понять и документировать динамическое поведение системы. К. Ларман [14] также подчеркивает, что диаграммы последовательности могут быть эффективным инструментом для выявления и устранения ошибок на ранних этапах разработки, а также для улучшения координации между членами команды, обеспечивая единое видение процесса взаимодействия компонентов системы.

Поэтому было принято решение разработать диаграмму последовательности для компьютерной 3D игры для развития пространственного мышления, которая представлена на рисунке 2.

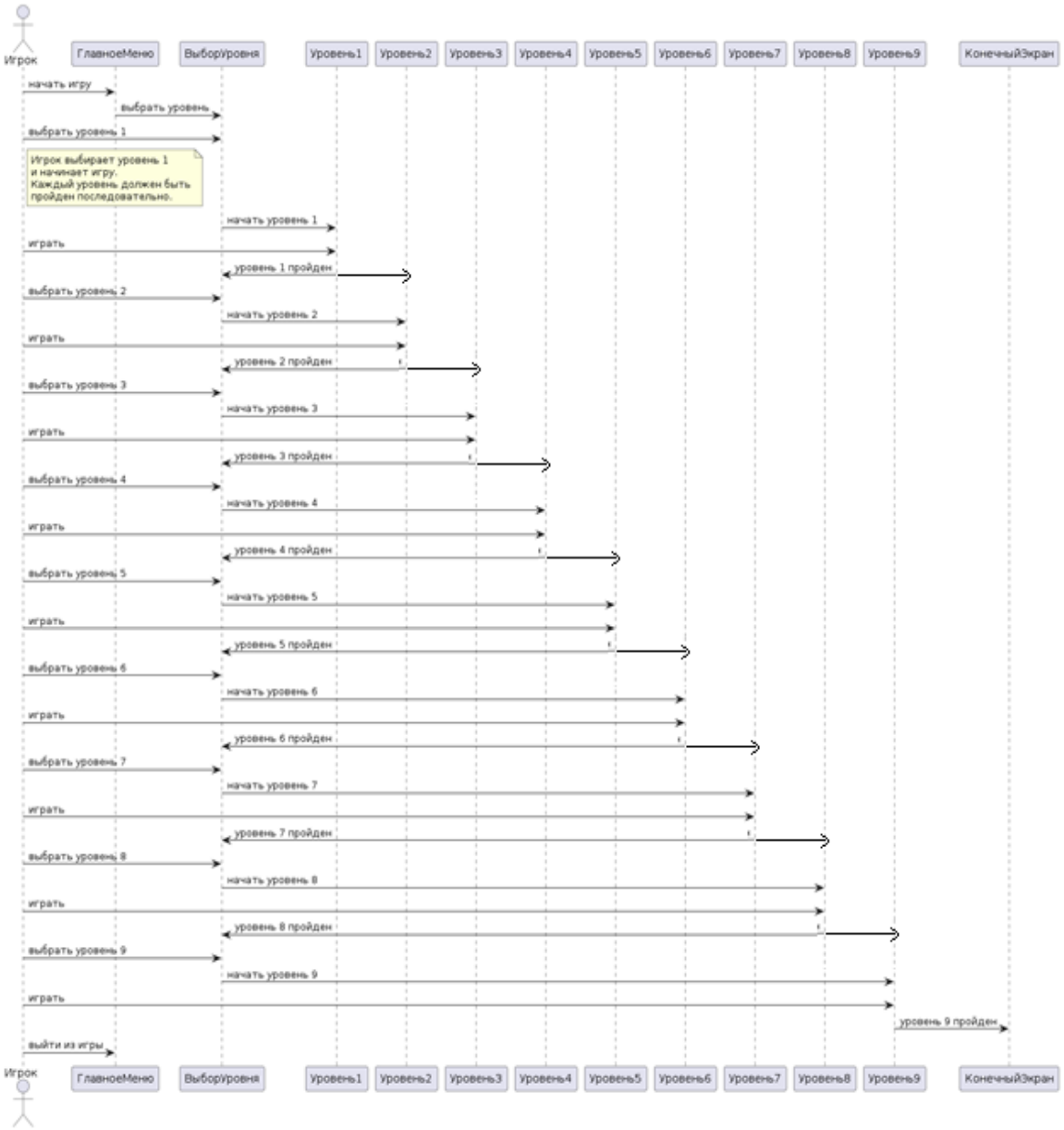

Рисунок 2 – Диаграмма последовательности

Как отмечает К. Ларман [14], 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 3, может осуществлять два действия:

– выбрать уровень;

– выйти из игры.

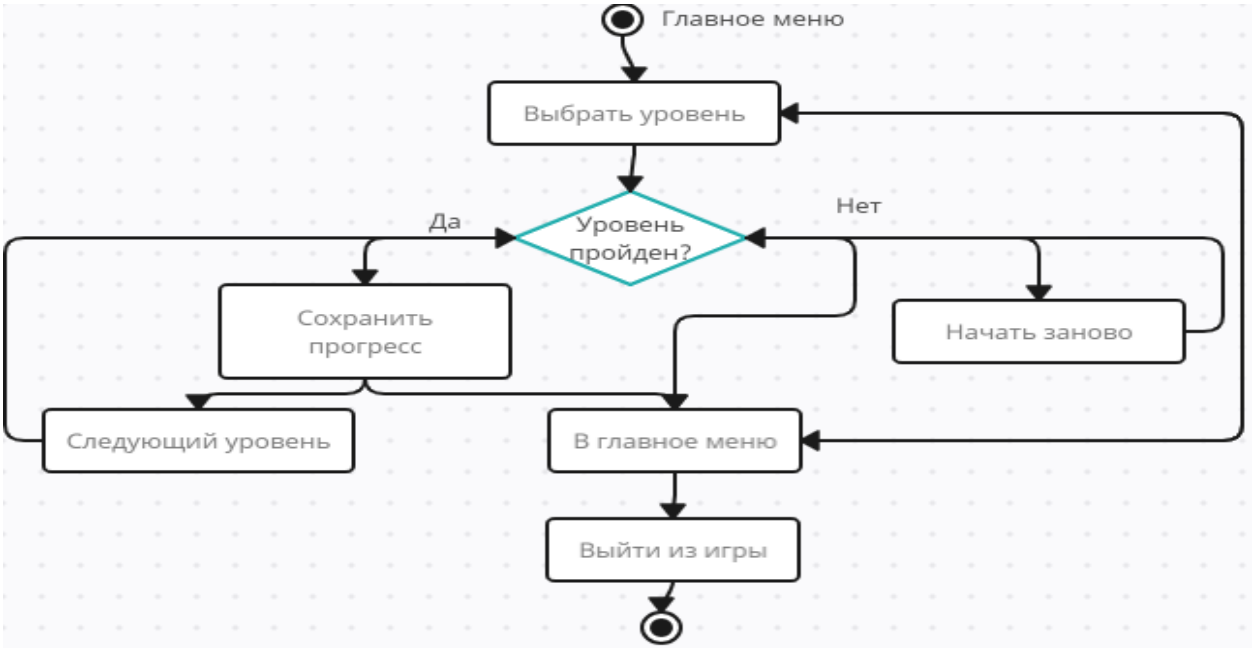

Рисунок 3 – Диаграмма активности

В данном параграфе были смоделированы диаграммы последовательности и активности, которые помогут при дальнейшей разработке компьютерной 3D игры для развития пространственного мышления.

1.4 Сравнительный анализ и выбор инструментальных средств разработки компьютерных игр

Выбор инструментальных средств разработки компьютерных игр (далее ИСДК) является важным этапом в процессе создания игры. ИСДК должны соответствовать потребностям проекта, обеспечивая функциональность, удобство использования и производительность.

При выборе ИСДК следует обратить внимание на их функциональность. ИСДК должен содержать следующую функциональность:

– редактор сцены и уровня;

– редактор ресурсов;

– интеграция с языками программирования;

– инструменты для работы с UI;

– систему создания анимаций;

– физический движок для настройки ассетов;

– тестирование и отладку;

– возможность управления аудио и звуковыми эффектами.

По мнению И. С. Кудряшова [11], «Игровой движок (англ. game engine) – комплекс программных средств (ПО), обеспечивающих работу графических, звуковых, геймплейных и других элементов видеоигры. Игровой движок может быть отнесён к промежуточному ПО (англ. middleware – связующее, или подпрограммное, обеспечение), поскольку предоставляет программным приложениям дополнительные возможности по обеспечению связи и управлению вводом и выводом, не заложенные в основной функционал платформы и операционной системы».

Unity [36] — многофункциональный игровой движок, используемый для создания 2D- и 3D-игр, интерактивных приложений, VR/AR-проектов и других интерактивных продуктов. Бесплатная версия Unity поддерживает Android, Web Player и PC платформы, а полная версия позволяет публиковать проекты на PC, Linux, Mac, Windows Store, iOS, Android, Windows Phone 10 Store, Blackberry 10, Wii U, PS3, Xbox 360, PS4 и Xbox One. Кроме того, Unity поддерживает разработку VR-приложений для Hololens, Oculus Rift, StarVR и других платформ.

Unity можно адаптировать под конкретные нужды разработки. Интерфейс можно настроить, убрав лишние элементы меню и добавив собственные настройки для упрощения процесса разработки. Unity обладает

интуитивным интерфейсом с функцией Drag and Drop и поддерживает два языка программирования: C# и JavaScript.

Отметим ключевые особенности Unity, опираясь на работу М. Гейга [7]:

– простота использования: интуитивно понятный интерфейс и drag-and-drop функционал делают Unity доступным как для опытных разработчиков, так и для новичков;

– универсальность: подходит для создания различных проектов, от простых мобильных игр до высокопроизводительных консольных игр [7].

– производительность: оптимизирован для высокой производительности на различных платформах, включая ПК, Mac, мобильные устройства, игровые консоли и VR/AR-гарнитуры;

– активное сообщество: большое и активное сообщество разработчиков обеспечивает доступ к обширным ресурсам, обучающим материалам и поддержке.

Анализ публикаций, посвященных разработке компьютерных игр на платформе Unity [5; 7; 9; 13; 17], позволил выделить следующие преимущества использования этой платформы:

– сокращение времени разработки: позволяет быстро и эффективно создавать прототипы и готовые игры благодаря обширным библиотекам ассетов и инструментов;

– многоплатформенная разработка: легко портировать игры на различные платформы, экономя время и ресурсы;

– доступ к обширным ресурсам: обширная библиотека обучающих материалов, документации и примеров кода облегчает процесс обучения и разработки;

– снижение затрат: бесплатное программное обеспечение, что делает его доступным для разработчиков любого бюджета.

К недостаткам Unity при выборе как ИСДК можно отнести то, что у движка проблемы с оптимизацией занимаемой памяти, контент Unity требует гораздо больше памяти для запуска и разработки, что может стать проблемой и замедлить процесс разработки.

Unreal Engine [40] (UE) — мощный игровой движок, разработанный компанией Epic Games, предназначенный для создания высококачественных 3D-игр, VR/AR-проектов, визуализаций, симуляций и других интерактивных приложений. Движок написан на языке C++ и поддерживает разработку игр для различных операционных систем и платформ.

Unreal Engine использует модульную систему зависимых компонентов, что упрощает процесс портирования игр на разные платформы. Движок поддерживает различные системы рендеринга, такие как Direct3D, OpenGL, и Pixomatic. Для воспроизведения звука используются EAX, OpenAI, DirectSound3D, а ранее поддерживались A3D. Также включены возможности голосового воспроизведения текста и распознавания речи. UE предоставляет модули для работы с сетью и поддержку различных периферийных устройств.

Ключевые особенности UE:

– реалистичная графика: UE обеспечивает передовые возможности рендеринга, позволяя создавать фотореалистичные изображения и визуальные эффекты;

– физический движок: включает продвинутый физический движок, обеспечивающий реалистичное взаимодействие объектов в игровом мире;

– система искусственного интеллекта: позволяет создавать сложных и реалистичных NPC (неигровых персонажей) и врагов;

– инструменты для разработки: широкий набор инструментов, включая редактор уровней, систему анимации, систему освещения и многое другое.

Недостатком при выборе UE как ИСДК, является то, что UE по сравнению с Unity рассчитан на крупные команды разработчиком с огромным функционалом, что порой одному разработчику не удастся создать качественную компьютерную игру.

Преимущества использования UE:

– создание игр мирового класса: UE используется для разработки некоторых из самых популярных и визуально впечатляющих игр.

– реалистичные визуальные эффекты: позволяет создавать фотореалистичные изображения и визуальные эффекты, близкие к реальности.

– расширяемость: UE можно расширять с помощью различных плагинов и инструментов, что позволяет создавать уникальные и индивидуальные проекты.

– доступ к обширным ресурсам: обширная библиотека обучающих материалов, документации и примеров кода облегчает процесс обучения и разработки.

Godot Engine [35] — многофункциональный игровой движок с открытым исходным кодом, предназначенный для создания 2D- и 3D-игр, VR/AR-проектов и других интерактивных приложений.

Ключевые особенности Godot Engine:

– бесплатность и открытый исходный код: Godot полностью бесплатен и доступен под открытой лицензией MIT, что делает его привлекательным для разработчиков с ограниченным бюджетом или тех, кто хочет иметь полный контроль над своим кодом;

– простота использования: интуитивно понятный интерфейс и drag-and-drop функционал делают Godot доступным как для опытных разработчиков, так и для новичков;

– универсальность: подходит для создания широкого спектра проектов, от простых мобильных игр до высокопроизводительных 3D-игр;

– производительность: оптимизирован для обеспечения высокой производительности на различных платформах, включая ПК, Mac, мобильные устройства, игровые консоли и VR/AR-гарнитуры.

Преимущества использования Godot Engine:

– сокращение времени разработки: позволяет быстро и эффективно создавать прототипы и готовые игры, используя обширные библиотеки ассетов и инструментов;

– многоплатформенная разработка: легко портировать игры на различные платформы, экономя время и ресурсы;

– снижение затрат: полностью бесплатен, что делает его доступным для разработчиков любого бюджета;

– доступ к обширным ресурсам: обширная библиотека обучающих материалов, документации и примеров кода облегчает процесс обучения и разработки.

Главным недостатком Godot Engine при выборе ИСДК будет являться то, что движок поддерживает только свой созданный язык программирования, тем самым увеличив время разработки до момента изучения языка.

На основании проведенного анализа можно сделать вывод, что Unity является наиболее подходящим ИСДК. Unity больше подходит для создания компьютерной 3D игры, направленной на развитие пространственного мышления, по следующим параметрам:

– легко осваивается и позволяет быстро начать разработку;

– подходит для создания игр на различных платформах;

– обеспечивает высокую производительность игры;

– имеет большое и активное сообщество, которое может оказать помощь в процессе разработки.

Для написания скриптов в Unity используется язык программирования C# [32] – это язык программирования, разработанный компанией Microsoft.

C# является одним из основных языков разработки в платформе Microsoft.NET и используется для создания различных типов приложений, включая настольные приложения, веб-приложения, игры, мобильные приложения и многое другое.

Преимущества использования C# для разработки игр в Unity:

– простота изучения: C# имеет относительно простой синтаксис, что делает его доступным как для опытных, так и для начинающих разработчиков;

– большое сообщество: C# имеет большое и активное сообщество разработчиков, что означает, что вы найдете множество ресурсов и поддержки, если вам понадобится помощь;

– производительность: C# оптимизирован для обеспечения высокой производительности, что важно для создания плавных и отзывчивых игр;

– гибкость: C# позволяет разработчикам использовать различные методы программирования, что дает им свободу творчества при разработке игр.

Среди популярных средств разработки на C# выделяются Microsoft Visual Studio и Visual Studio Code. C# предоставляет разработчикам доступ к множеству сторонних библиотек и фреймворков, что ускоряет процесс разработки и расширяет функциональность приложений.

Так как C# является основным языком программирования для Unity, рекомендуется использовать Visual Studio для более удобной работы. Visual Studio — это не только IDE (интегрированная среда разработки), но и комплексный инструмент для разработки приложений, охватывающий весь жизненный цикл создания программного обеспечения.

Основные функциональные возможности Visual Studio [41]:

– написание и редактирование кода: удобный редактор с подсветкой синтаксиса, автодополнением и множеством других функций, облегчающих процесс написания и редактирования кода;

– отладка кода: мощные инструменты для пошагового выполнения кода, отслеживания значений переменных, установки точек останова и выявления ошибок;

– сборка кода: возможность компиляции кода в различные форматы, такие как EXE, DLL и библиотеки;

– развертывание приложений: упрощенный процесс развертывания готовых приложений на локальных и удаленных серверах;

– дополнительные возможности: поддержка множества расширений и дополнительных функций, позволяющих адаптировать IDE к потребностям разработчика.

Инструменты для работы с графикой и звуком

GIMP [34] — бесплатный растровый графический редактор с широким набором функций для работы с изображениями.

FL Studio [33] — цифровая звуковая рабочая станция, предназначенная для написания и производства музыки с помощью встроенных инструментов и эффектов.

В этом параграфе проведён сравнительный анализ различных инструментальных средств разработки игр с целью определить наиболее подходящие для проектирования компьютерной игры в жанре головоломка. Изучены основные движки Unity, Unreal Engine, Godot Engine и рассмотрены их достоинства и недостатки. Кроме выбора движка, также определены инструменты для работы с графикой и звуком.

2 Разработка компьютерной 3D игры, направленной на развитие пространственного мышления

Цель данной главы — описать процесс разработки компьютерной 3D игры для развития пространственного мышления в соответствии с алгоритмом, приведенным в параграфе 1.1.

2.1 Установка программного обеспечения

Для начала разработки необходимо скачать и установить программное обеспечение Unity. Нужно перейти на официальный сайт [36] и выбрать раздел «Download Unity».

Далее необходимо выбрать последнюю стабильную версию для получения всех последних обновлений и функции приложения.

Во время первого запуска приложения UnityHub потребуется войти в учетную запись UnityID или зарегистрироваться. Для регистрации потребуется ввести электронную почту, пароль и полное имя. Окно создания учетной записи UnityID представлено на рисунке 4.

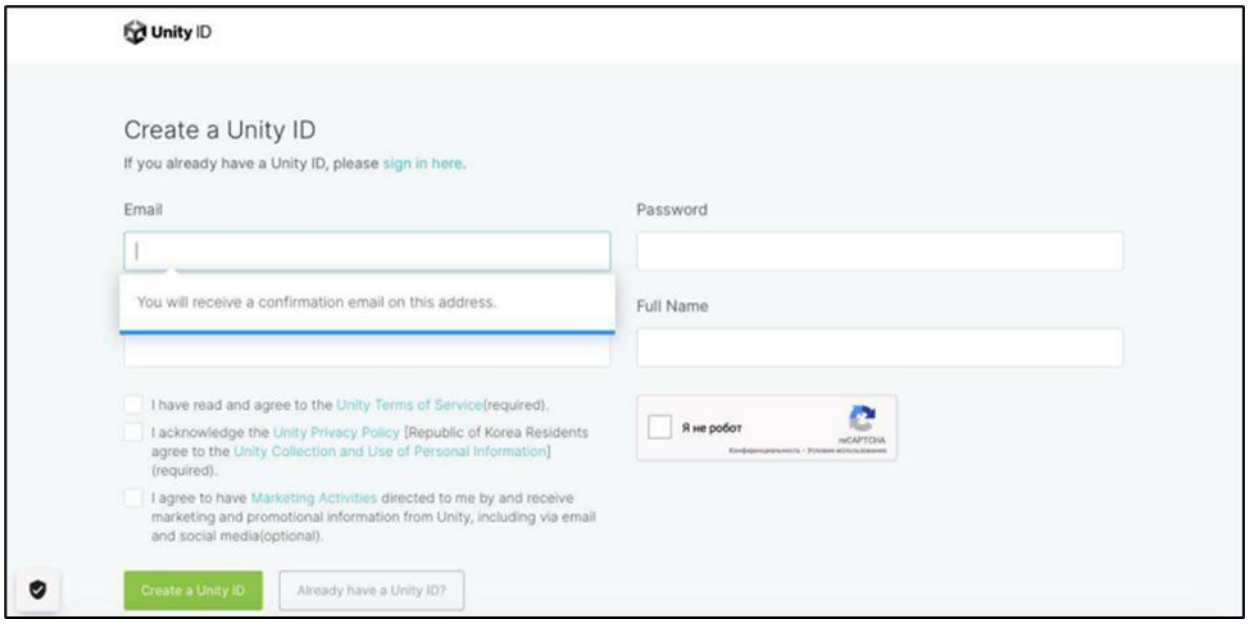

Рисунок 4 – Создание учетной записи UnityID

После успешного входа в учетную запись, откроется окно с проектами, библиотекой, плагинами и другими продуктами Unity, которое представлено на рисунке 5.

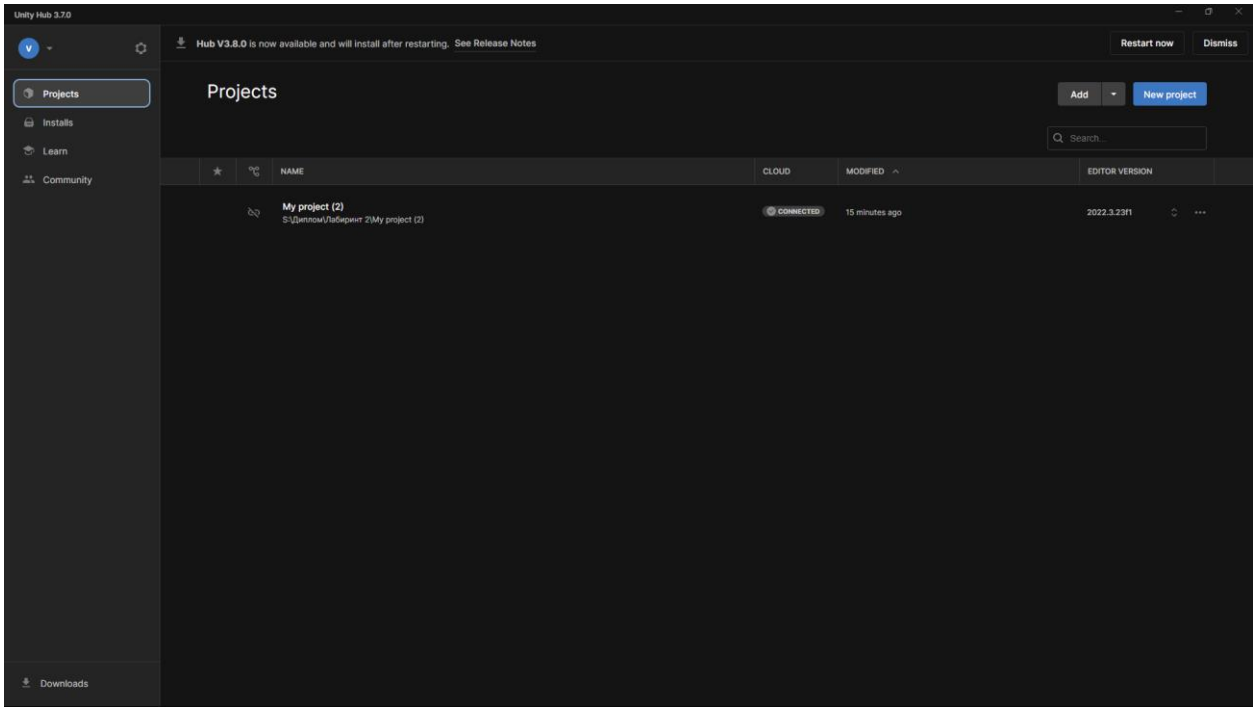

Рисунок 5 – Окно UnityHub

Для удобного редактирования кода необходимо скачать и установить редактор кода Microsoft Visual Studio 2022 (далее – MVS 2022) с официального сайта [41], с которым удобно работать в среде Unity для разработки игр. Загрузка MVS 2022 представлена на рисунке 6.

Для работы с аудио необходимо скачать и установить приложение FL Studio с официального сайта [33]. Загрузка FL Studio представлена на рисунке 7.

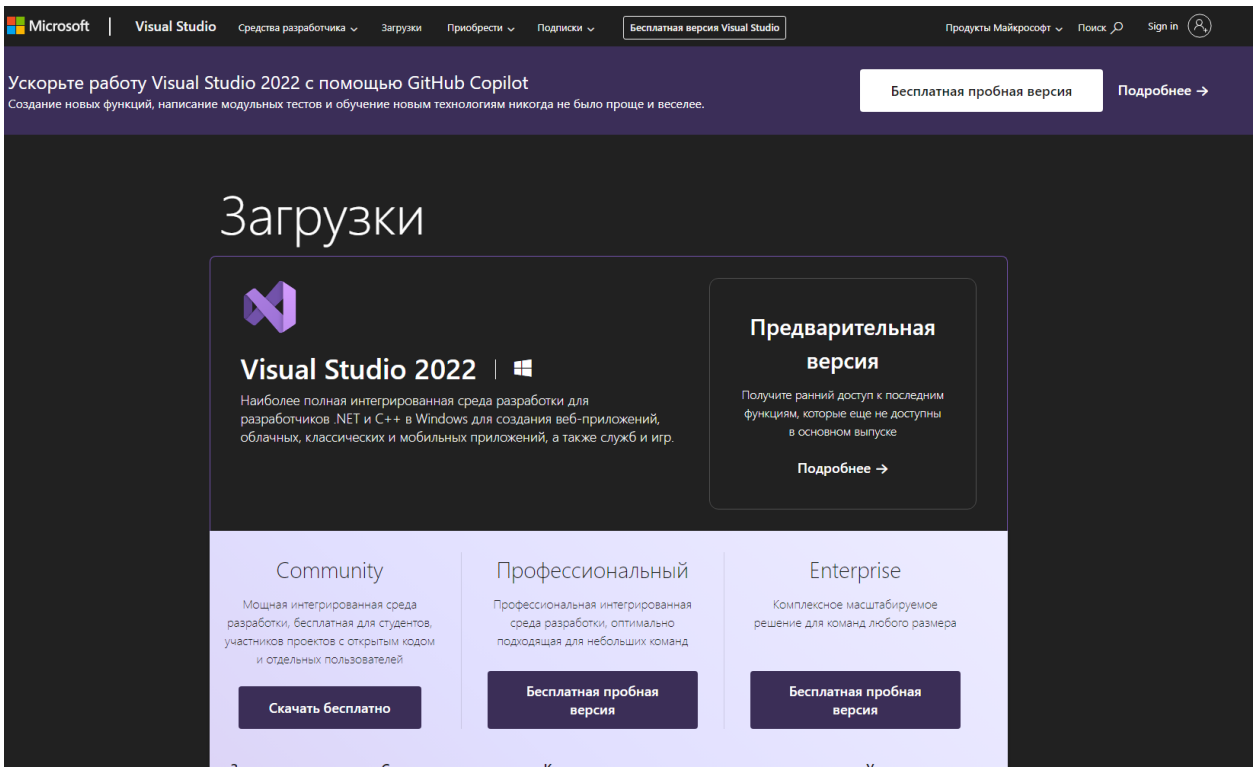

Рисунок 6 – Загрузка MVS 2022

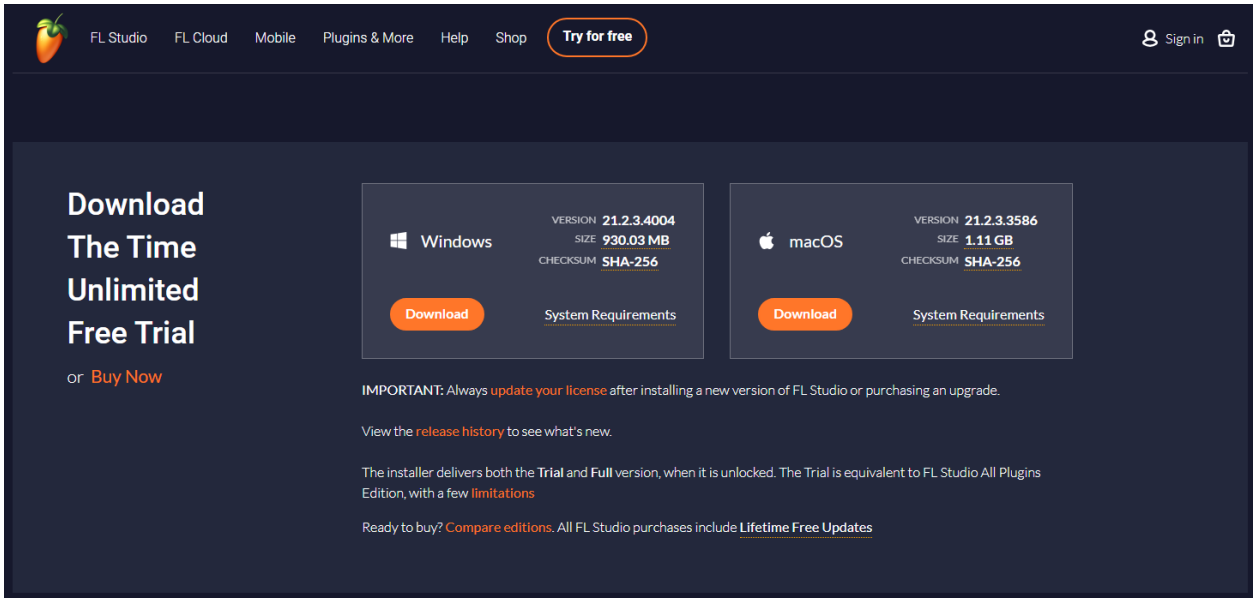

Рисунок 7 – Загрузка FL Studio

Для работы с графикой был выбран растровый редактор Gimp. Для его загрузки необходимо зайти на официальный сайт [34] и выбрать удобный способ скачивания файла. Загрузка Gimp представлена на рисунке 8.

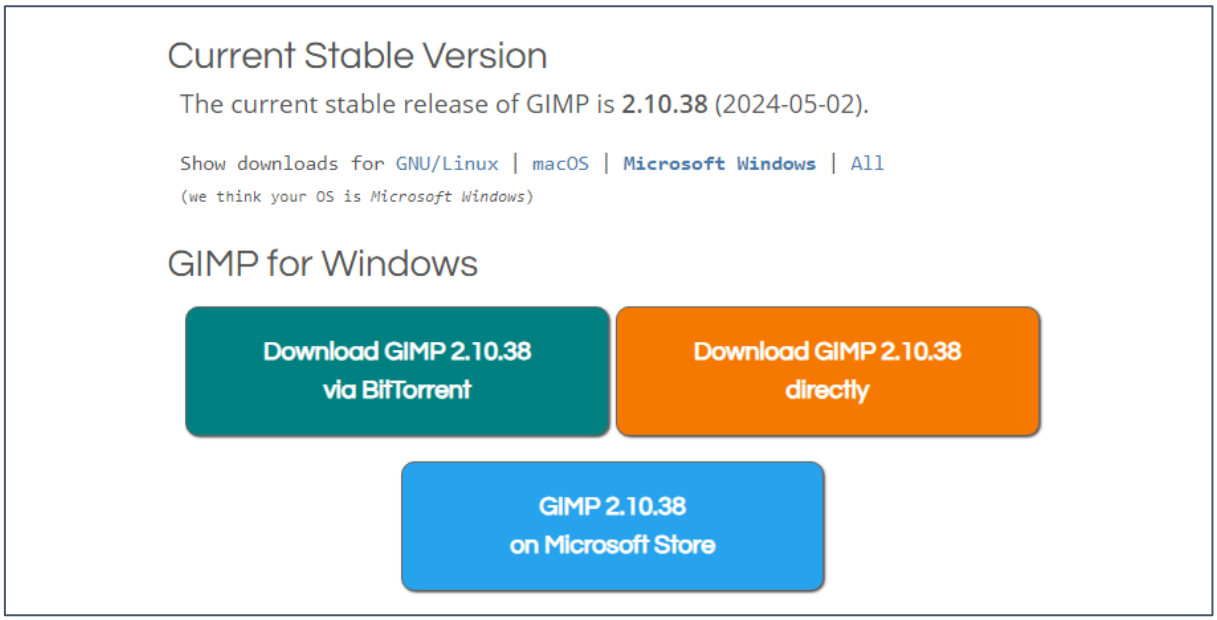

Рисунок 8 – Загрузка Gimp

В данном параграфе было установлено всё программное обеспечение, которые будет использоваться при создании компьютерной 3D игры для развития пространственного мышления.

2.2 Создание нового проекта

После установки необходимого программного обеспечения, нужно перейти в UnityHub, для создания игрового проекта. Необходимо в разделе «Projects» нажать на кнопку «New project», после чего откроется новое окно с выбором необходимых параметров и шаблонов разработки. Для создания компьютерной 3D игры для развития пространственного мышления необходимо выбрать шаблон разработки «3D (Built-in Render Pipeline) и нажать на кнопку «Create project», как представлено на рисунке 9.

После создания нового проекта, появится окно с пустой сценой. В центре окна будет находиться рабочая зона «Scene», слева – «Hierarchy», окно, которое использует концепцию родительско-дочерних иерархий или родительских элементов для группировки игровых элементов. Справа находится окно «inspector», в котором содержится информация о выбранном

объекте сцены. Так же в этом окне снизу находится две вкладки: «Project», в которой содержатся все объекты, которые загружены в данную игру для использования, и вкладка «Console», которая используется для определения ошибок, вывода сообщений и исключений, возникающих во время разработки. Всё это изображено на рисунке 10.

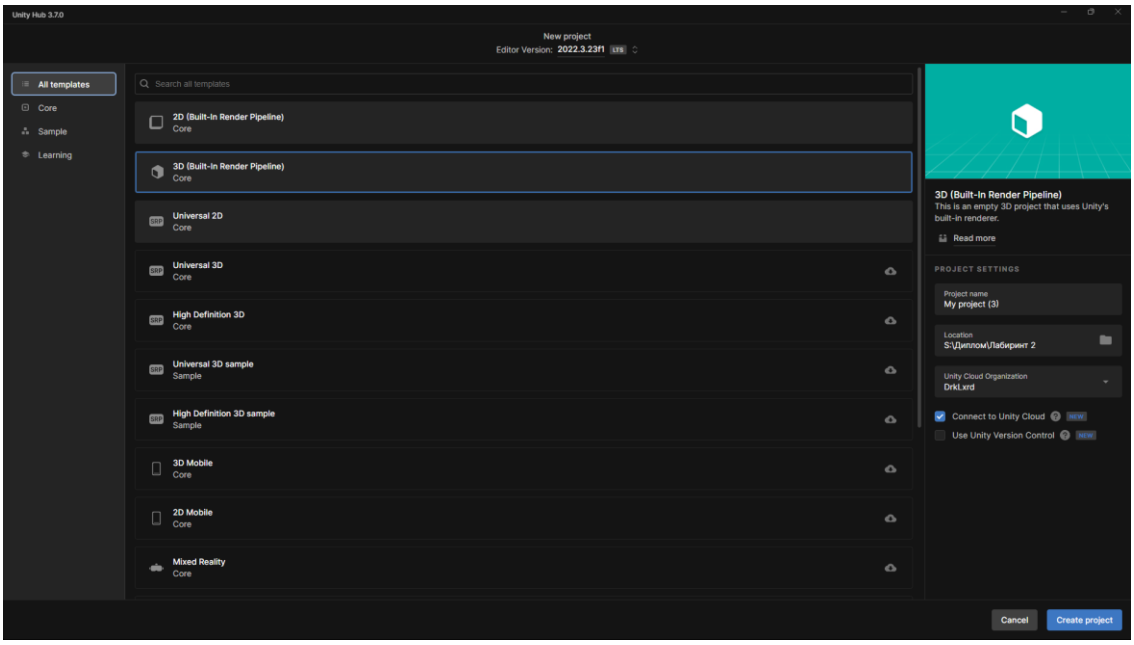

Рисунок 9 – Окно выбора шаблона разработки UnityHub

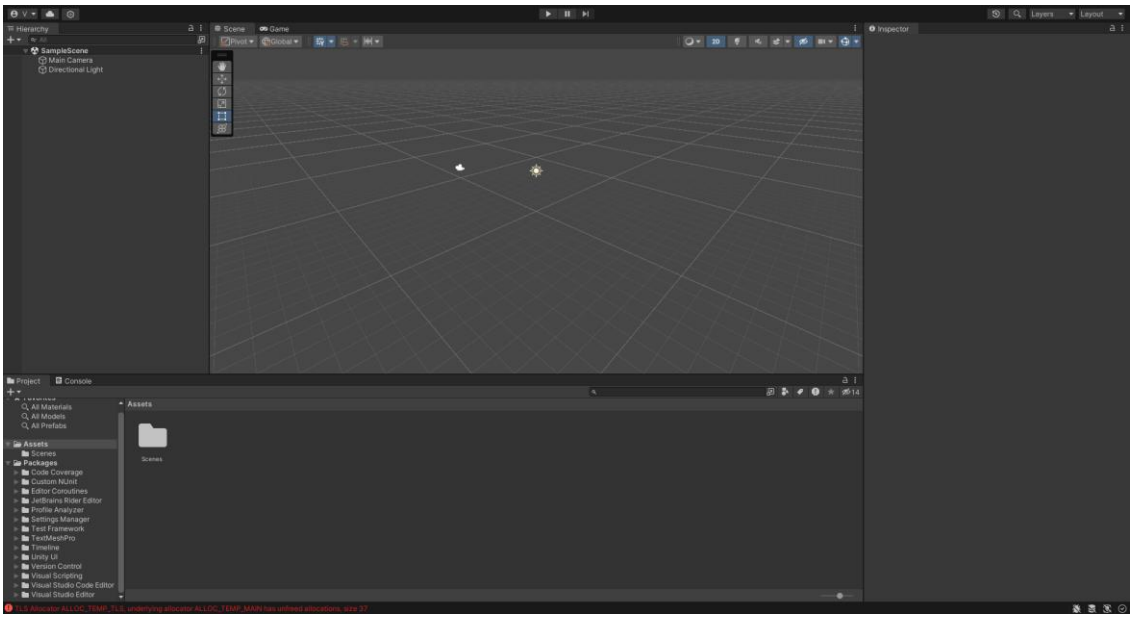

Рисунок 10 – Окно проекта UnityHub

2.3 Создание игрового персонажа

2.3.1 Создание формы игрового персонажа

Для создания игрового персонажа используется бесплатная библиотека Unity. Взятого игрового аватара можно изменять по своему усмотрению, например, взятую бесплатную форму персонажа, которая представлена на рисунке 11.

Рисунок 11 – Форма игрового персонажа

С помощью графического редактора Gimp нужно создать для него текстуру тела, изображенную на рисунке 12. Отметим, что к форме игрового персонажа прилагается шаблон текстуры, которую разработчик может заполнить по своему усмотрению.

Аналогичным образом персонажу необходимо нарисовать текстуру головы, которая продемонстрирована на рисунке 13.

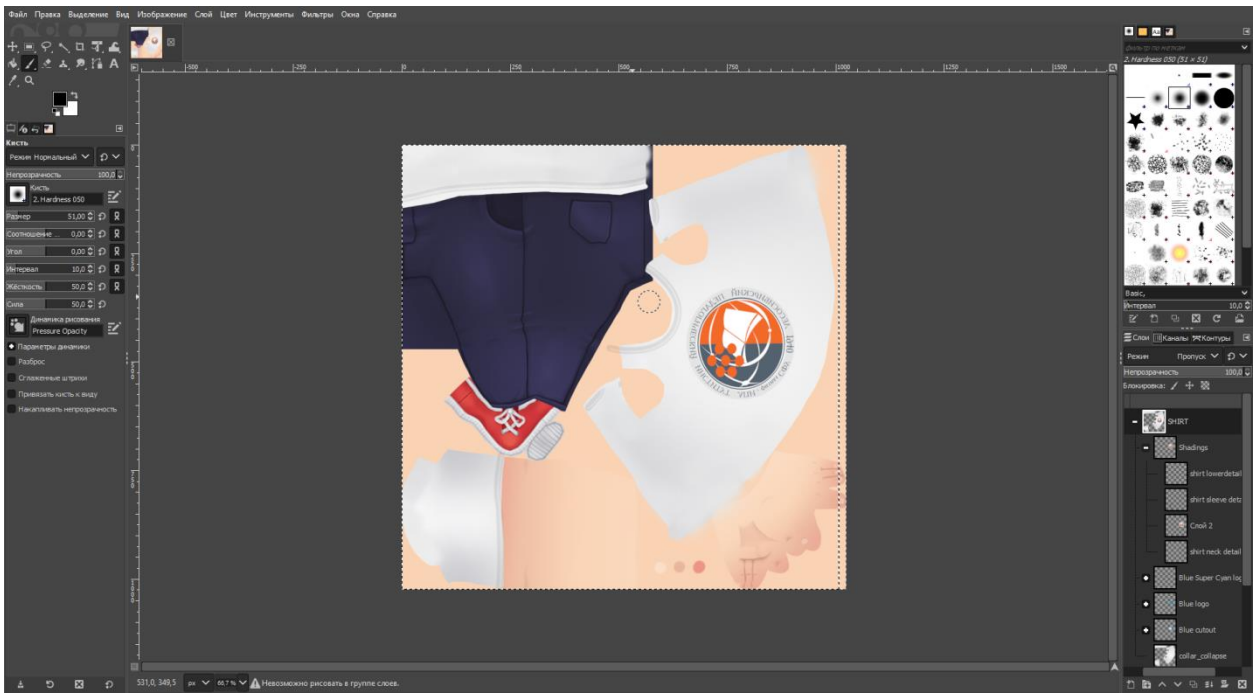

Рисунок 12 – Текстура тела игрового персонажа

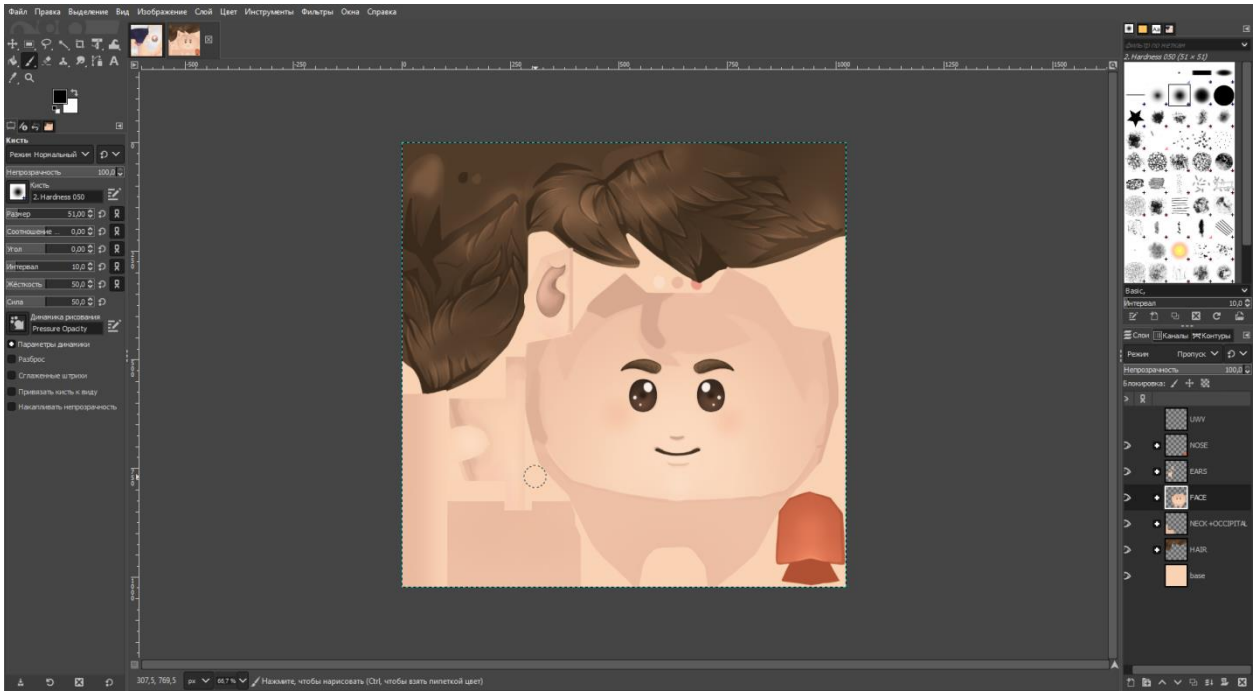

Рисунок 13 – Текстура головы персонажа

Из этой же библиотеки добавляется в проект рюкзак, который будет носить персонаж на спине. Рюкзак без текстуры изображен на рисунке 14.

Рисунок 14 – Форма рюкзака

С помощью Gimp нужно нарисовать для рюкзака текстуру, как представлено на рисунке 15.

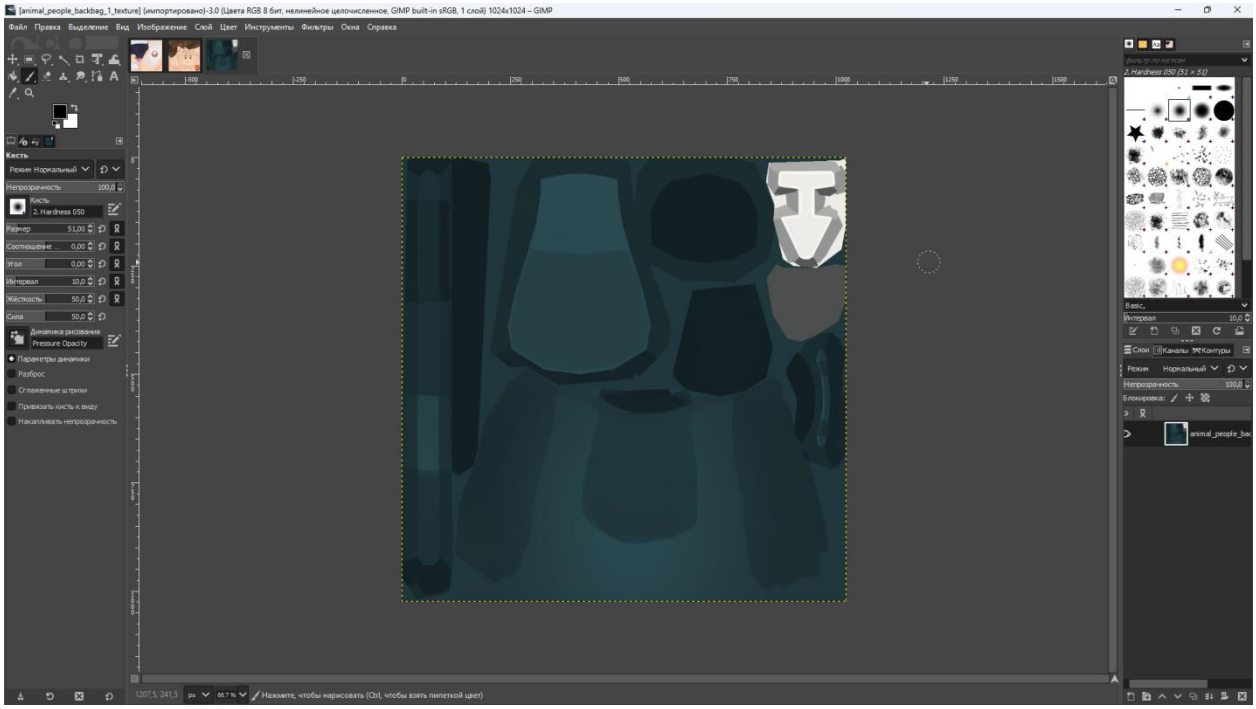

Рисунок 15 – Текстура рюкзака

Далее необходимо создать скрипт для правильного расположения рюкзака на спине персонажа. Для этого создаётся новый скрипт в окне

Project с именем «Accessory Wear Logic». В открывшемся окне MVS 2022 нужно написать код для правильного положения рюкзака на персонаже. После чего необходимо выделить персонажа и перенести в окно Inspector скрипт из окна Project. Затем рюкзак добавляется в группу персонажа во вкладке Hierarchy. Поиск необходимой точки скелета для крепления рюкзака на персонаже в скрипте представлен в Приложение А.

Далее нужно перенести созданные текстуры в окно Inspector персонажа и рюкзака. Результат выполнения представлен на рисунке 16.

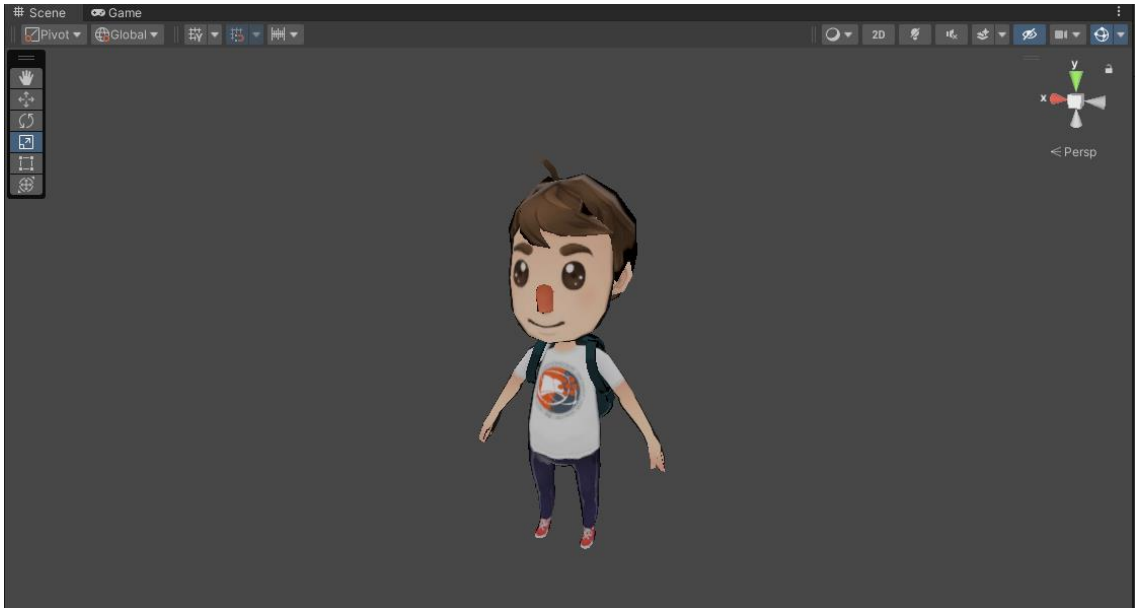

Рисунок 16 – Игровой персонаж

Во вкладку Hierarchy нужно добавить в группу персонажа виртуальную камеру для вида от третьего лица при запуске игрового уровня. Результат представлен на рисунке 17.

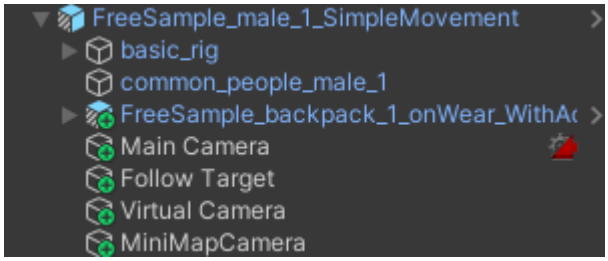

Рисунок 17 – Группа объектов внутри объекта персонажа

2.3.2 Добавление анимации персонажа

В бесплатной библиотеке необходимо найти готовые анимации скелета персонажа. После выбора нужных анимаций, необходимо выбрать персонажа и перейти в окно Animator в UnityHub. Далее создается условие для проигрывания различных анимаций скелета персонажа.

Для создания схемы условия нужно сгруппировать выбранные анимации скелета:

– анимация нахождения в воздухе (midair);

– анимация приземления (Land);

– анимация передвижения (Movement);

– анимация прыжка (jump).

На рисунке 18 представлена схема проигрывания анимаций персонажа в различных положениях.

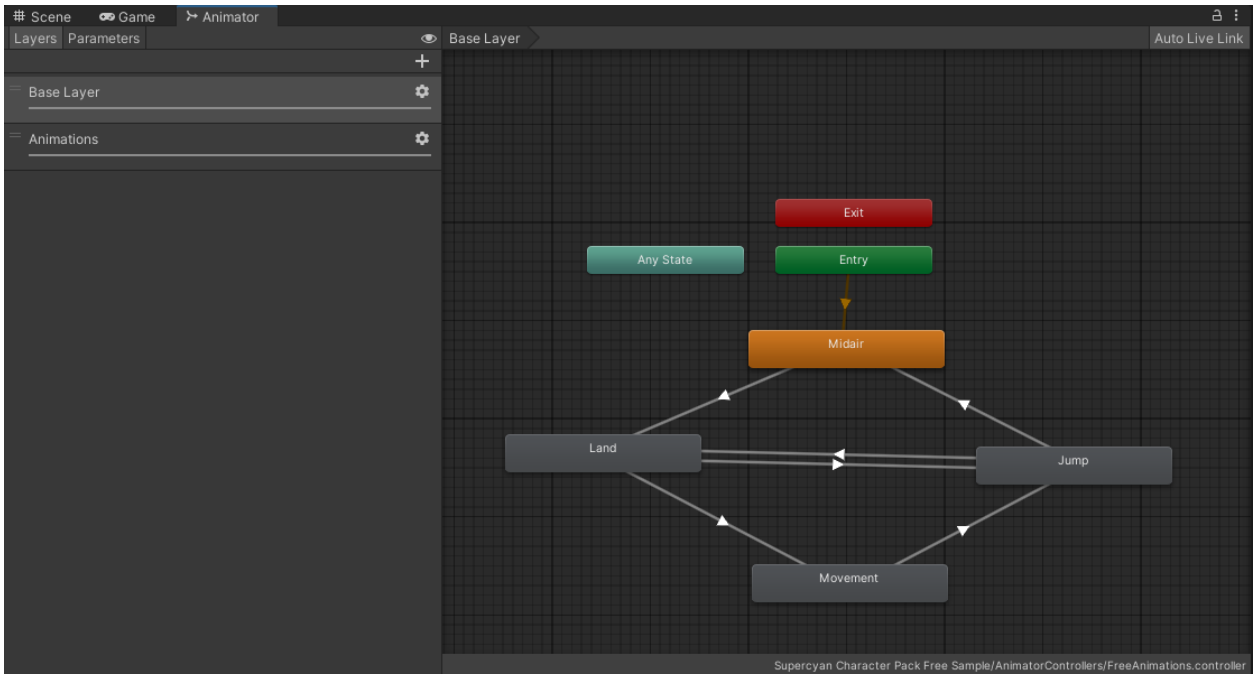

Рисунок 18 – Схема условий поведения анимаций персонажа

После группировки анимаций из бесплатной библиотеки, необходимо создать скрипт для управления персонажем, при движении которого будут

срабатывать анимации. Для этого нужно перейти в MVS 2022 и написать код скрипта, при соблюдении условий которого будут срабатывать прописанные анимации. Код скрипта для управления персонажем представлен в Приложение Б.

В методе TankUpdate передвижение персонажа выполняется через стандартные Unity-входы, такие как клавиши:W, A, S, D; или же стрелки на клавиатуре. При зажатии клавиши L.Shift, персонаж замедлит шаг.

После создания скрипта его нужно перенести в окно Inspector созданного персонажа. Теперь, когда игрок будет нажимать стандартные клавиши Unity-входы, будет проигрываться нужная анимация персонажа.

В данном параграфе был описан алгоритм создания игрового персонажа, которым в дальнейшем будет управлять игрок.

2.4 Создание игрового уровня

Уровни в игре будут представлены в виде лабиринтов, которые необходимо будет пройти за отведенное игроку время. Для усложнения прохождения на уровне будут присутствовать различные физические преграды, которые будут замедлять игрока. Чтобы завершить уровень, игроку необходимо найти среди коридоров дверь для перехода на следующий уровень или его завершения.

Для создания игрового уровня потребуется скачать из бесплатной библиотеки Unity необходимые игровые объекты (ассеты), которые будут использоваться при построении локаций игры.

Скачав нужные ассеты, необходимо их импортировать. Для этого потребуется перейти во вкладку Window → Package Manager → My assets, и, нажав кнопку Import, импортировать необходимые ассеты, после чего они появятся в окне проекта.

Для размещения ассетов в окне Scene необходимо перенести необходимый объект из окна Project в область окна Scene. Используя стандартные инструменты Unity, можно: перемещать, вращать и изменять размер добавленных объектов.

Далее нужно создать напольную плиту, стены, арки и двери, необходимые для проектирования структуры лабиринта. Для замедления прохождения игроком уровня, создаются препятствия, которые игроку необходимо будет перепрыгнуть. Чтобы локация не казалась пустой, расставляются различные предметы декора. Далее добавляется стандартный источник света «Directional Light». Для всех ассетов на уровне добавляется свойство Mesh Collider в окне Inspector для того, чтобы они приняли физическую форму, и персонаж не проходил сквозь данные объекты.

Результат создания игрового уровня 1 представлен на рисунке 19.

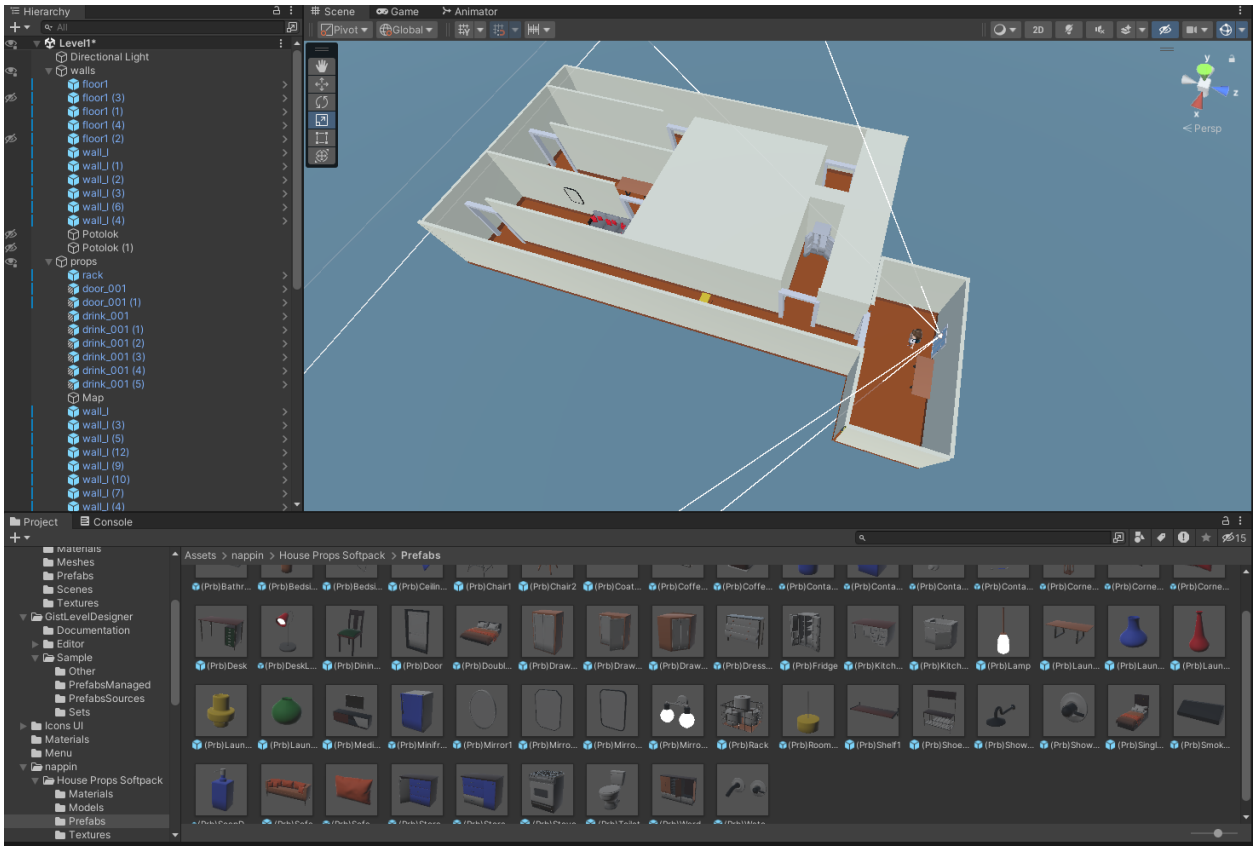

Рисунок 19 – Игровой уровень 1

В данном параграфе был описан алгоритм разработки уровень 1, знакомящий игрока с игровыми механиками и условиями игры, остальные уровни создаются аналогично.

2.5 Создание пользовательского интерфейса

2.5.1 Создание пользовательского интерфейса внутри уровня

Разработку пользовательского интерфейса целесообразно начать с создания таймера, отсчета времени отведенного на прохождение одного уровня. На каждом уровне он будет задавать разные значения, в зависимости от сложности прохождения уровня. Для этого создается UI элемент Text в окне Hierarchy и переименовывается в TimerText. Далее необходимо разместить данный элемент интерфейса в верхней части экрана посередине, после чего добавить его в группу Timer.

Для создания скрипта таймера нужно выделить его группу элементов и поместить в него файл с прописанным кодом. Полный листинг кода представлен в Приложении В.

Время отсчета таймера редактируется в поле скрипта в окне Inspector. Результат работы над таймером представлен на рисунке 20.

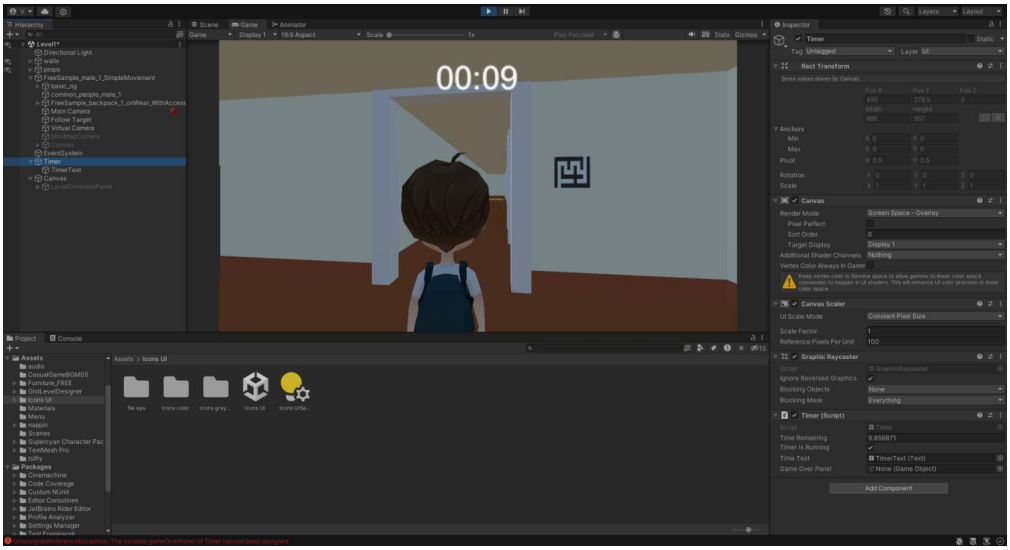

Рисунок 20 – Таймер

Далее нужно создать окно завершения уровня. Для этого нужно игровому объекту «дверь», который расположен в конце уровня, поставить тег Finish из выплывающего меню в свойствах Inspector, Как продемонстрировано на рисунке 21.

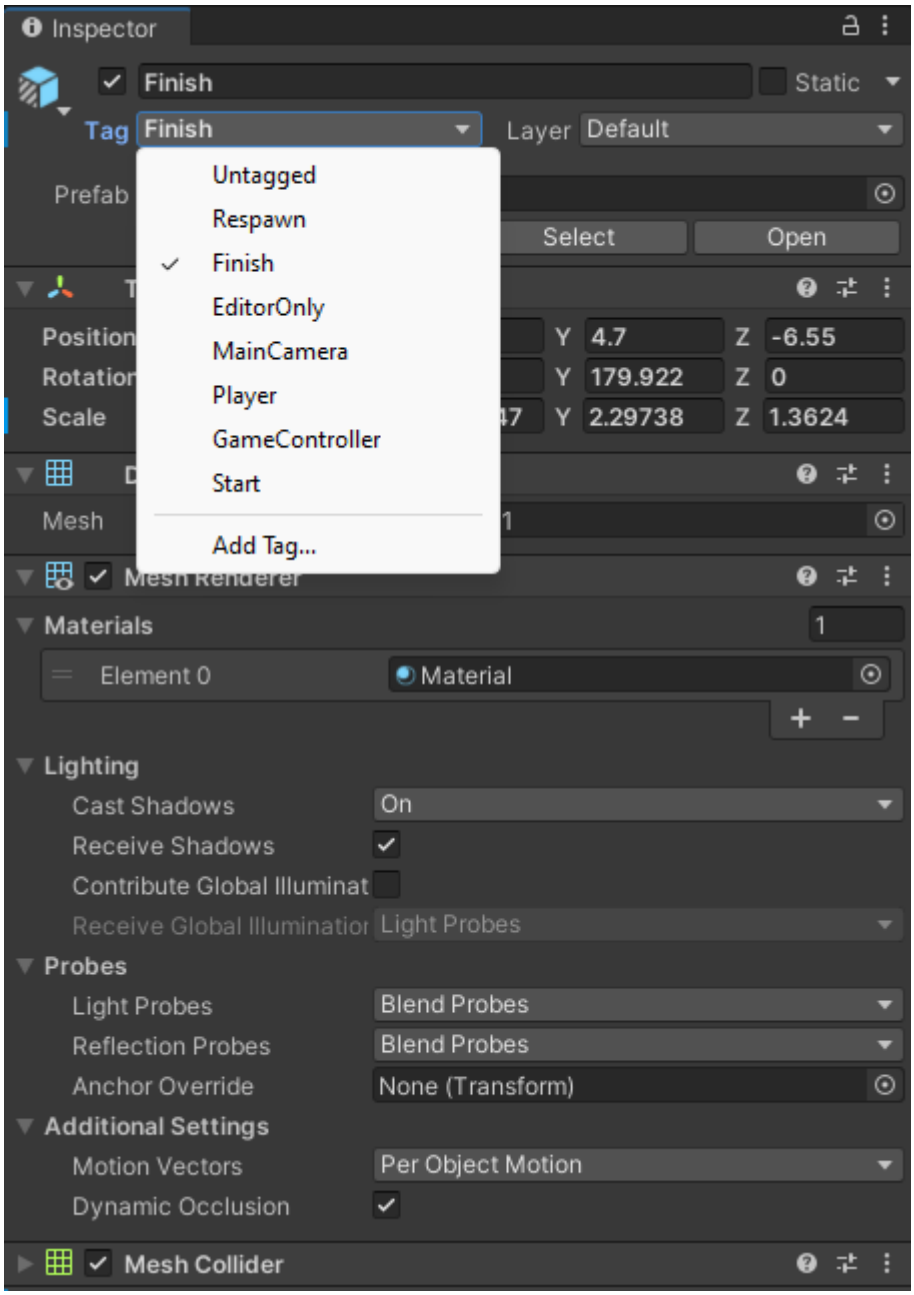

Рисунок 21 – Присвоение тега Finish

В свою очередь, объекту персонажа нужно присвоить тег Player, как продемонстрировано на рисунке 22.

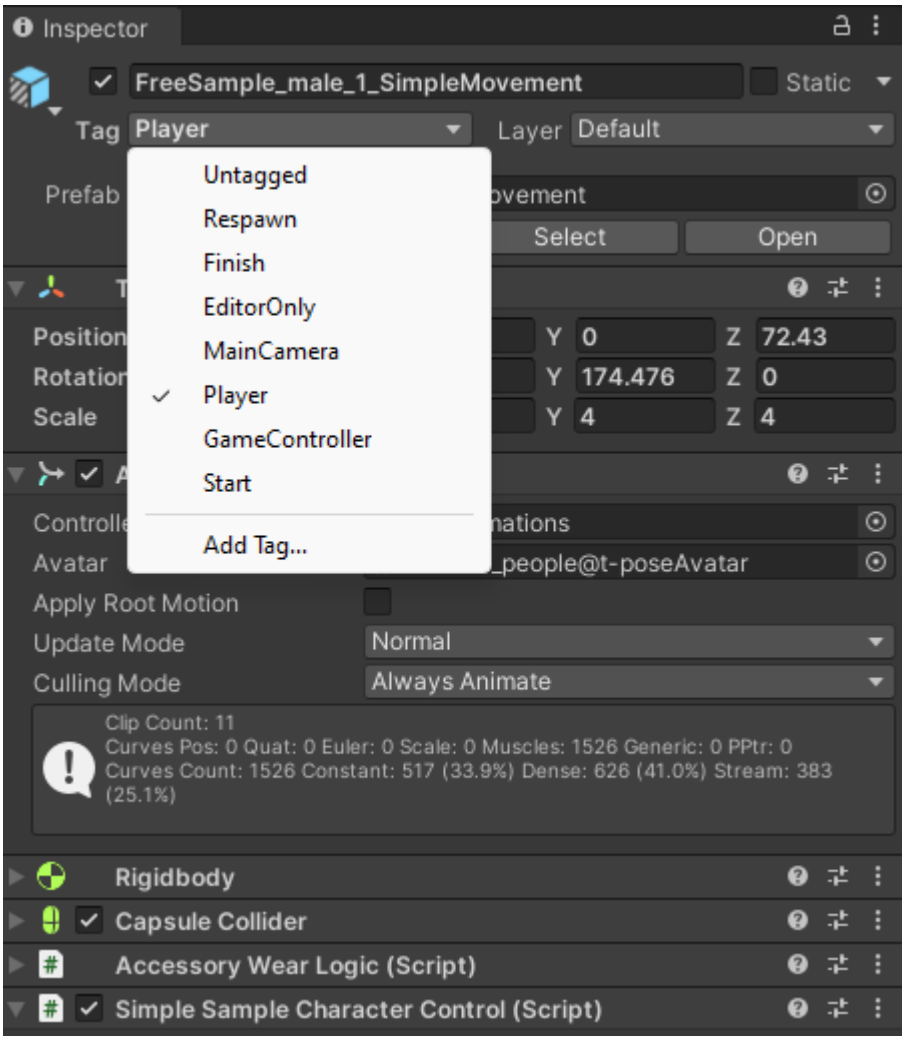

Рисунок 22 – Присвоение тега Player персонажу

Далее нужно создать UI объект Canvas и внутрь него поместить объект Panel, переименовав его в LevelCompletePanel. Внутри панели необходимо создать две кнопки, отвечающих за переход на следующий уровень и возвращение в главное меню, а также текстовую линию с надписью «Вы прошли 1 уровень!».

Для отображения панели LevelCompletePanel нужно создать новый скрипт, который будет срабатывать при завершении игроком уровня. Скрипт представлен в Приложении Г.

Теперь при соприкосновении объекта с тегом «Player» объекта «Finish», будет выводиться панель завершения с предложением перейти на следующий уровень или вернуться в главное меню. При нажатии кнопки «Следующий уровень» с присоединенным скриптом, игрока перенесет на следующий уровень, а при нажатии кнопки «В главное меню», откроется сцена MainMenu. Результат выполненной работы представлен на рисунке 23.

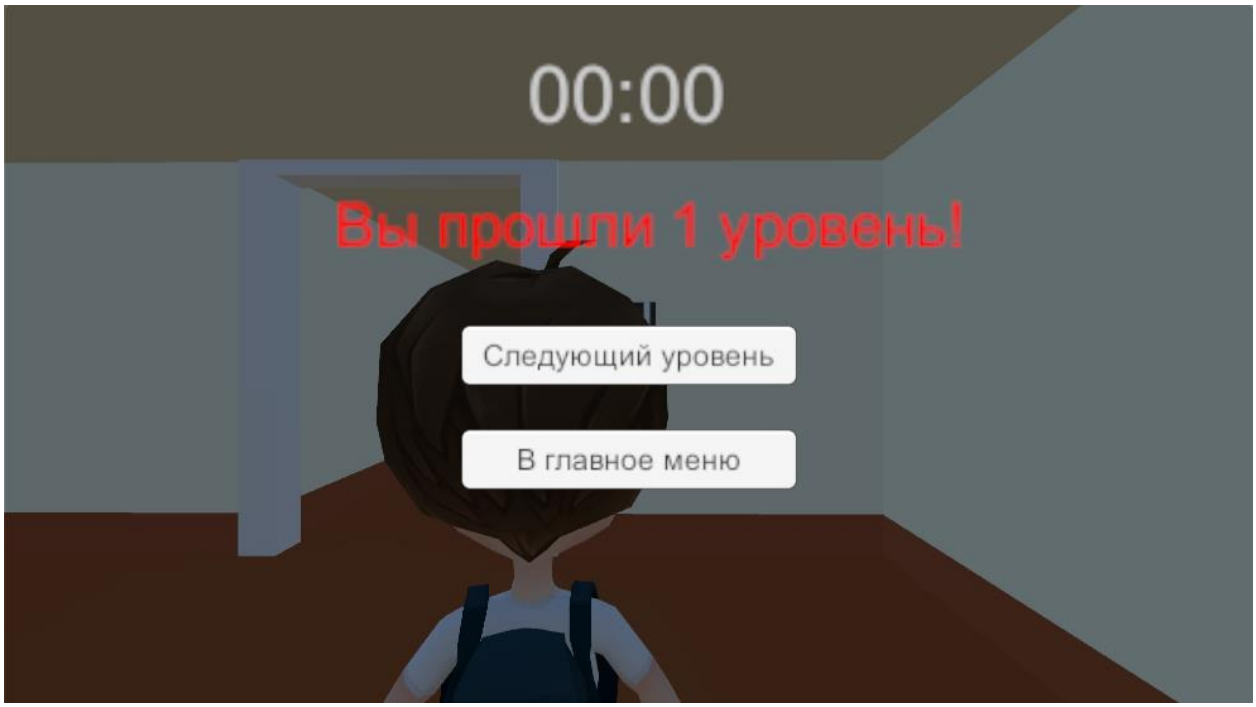

Рисунок 23 – Панель завершения уровня

Также нужно создать панель экрана проигрыша. Для этого необходимо создать элемент Panel с названием gameOverPanel и внутри него разместить две кнопки, отвечающих за перезапуск уровня и возвращение в главное меню, а также создать текстовый объект с надписью «Время вышло».

Для корректной работы кнопки «Заново» нужно создать новый скрипт, отвечающий за перезапуск уровня. Скрипт представлен в Приложении Д.

Для корректной работы кнопки «В главное меню» нужно создать новый скрипт, отвечающий за возвращение игрока в главное меню игры. Скрипт представлен в Приложении Е.

Теперь по истечению времени таймера высветится окно, сообщающее о проигрыше игрока на данном уровне, которое предложит перезапустить уровень, либо вернуться в главное меню игры. Результат представлен на рисунке 24.

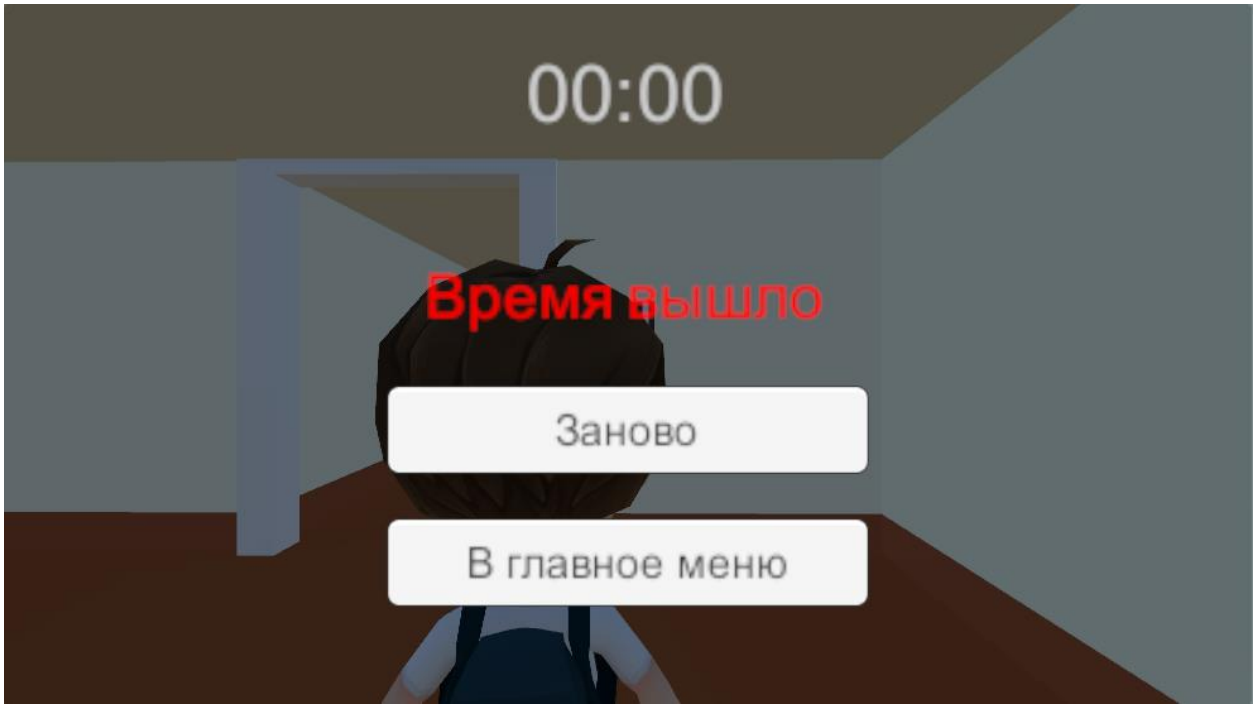

Рисунок 24 – Окно, сообщающее о проигрыше игрока

Далее необходимо создать мини-карту для ориентации игрока по уровню. Для начала нужно создать новый Canvas внутри группы персонажа. Внутри нового полотна создать объект «Camera» и переименовать её в «MiniMapCamera», далее через Inspector поменять её угол обзора так, чтобы она смотрела сверху под прямым углом на нашего персонажа.

Также нужно создать новый компонент «RenderTexture» для проекта, он будет отвечать за отрисовку игрового уровня с положения MiniMapCamera.

Далее нужно создать круглые границы для мини-карты и задать им черный цвет. Также поставить маркер на место нашего персонажа в мини

карте для удобного отслеживания положения, и таким же образом создать метку над объектом с тегом Finish. Результат представлен на рисунке 25.

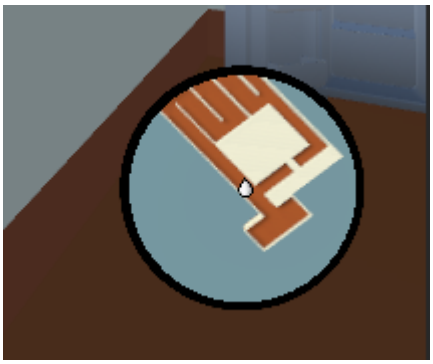

Рисунок 25 – Мини-карта

Мини-карта в играх для развития пространственного мышления (особенно это касается лабиринтов) является важнейшим элементом пользовательского интерфейса по ряду причин:

– лабиринты часто сложны в навигации, и мини-карта помогает игрокам понять, в какой части лабиринта они находятся;

– мини-карта делает игровой процесс более удобным и менее стрессовым, что позволяет игрокам сосредоточиться на решении задач проложения верного маршрута к финишу;

– благодаря мини-карте игрок учится соотносить 2D и 3D проекции.

В данном параграфе был разработан пользовательский интерфейс (UI) с входящими в него элементами, такими как: таймер, мини-карта, окна завершения и проигрыша уровня.

2.5.2 Создание главного меню игры

Главное меню игры – это интерфейсный экран, который содержит основные функции взаимодействия с игрой. В разрабатываемой игре главное меню будет содержать 2 кнопки:

– «выбрать уровень», с помощью которой игрок сможет перемещаться и возвращаться между уровнями;

– «выход из игры», которая способствует закрытию игры.

Для начала нужно создать задний фон, используя бесплатные ассеты из библиотеки Unity. Далее расположить две кнопки посередине экрана и переименовать их в окне Hierarchy на «StartButton» и «ExitButton». Внутри кнопок расположить соответствующий текст («Выбрать уровень» или «Выйти из игры»). Результат представлен на рисунке 26.

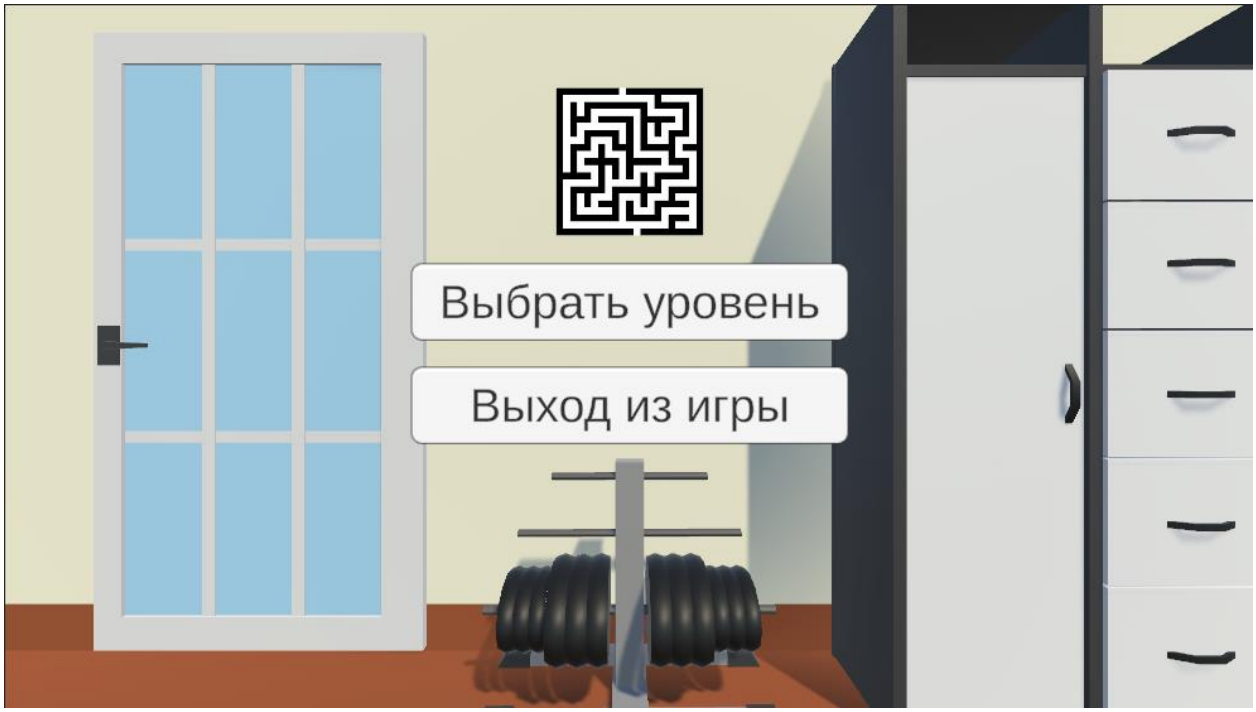

Рисунок 26 – Главное меню игры

Далее нужно создать новую панель для перехода по кнопке «Выбрать уровень» и назвать её «LevelsPanel». Нужно сделать её не статичной, отключив данный параметр в окне Inspector. Внутри панели расположить 9 кнопок, каждая из которых будет перемещать игрока на определенный уровень. Для того, чтобы кнопка «Выбрать уровень» работала корректно, нужно написать новый скрипт и расположить его внутри нового объекта «Controller», который будет отвечать за сохранение прогресса игрока и

загрузки определенного доступного уровня. Скрипт представлен в Приложении И.

Для того, чтобы игрок открыл недоступный уровень, ему потребуется пройти предыдущий за отведенное время.

Окно выбора уровня представлено на рисунке 27.

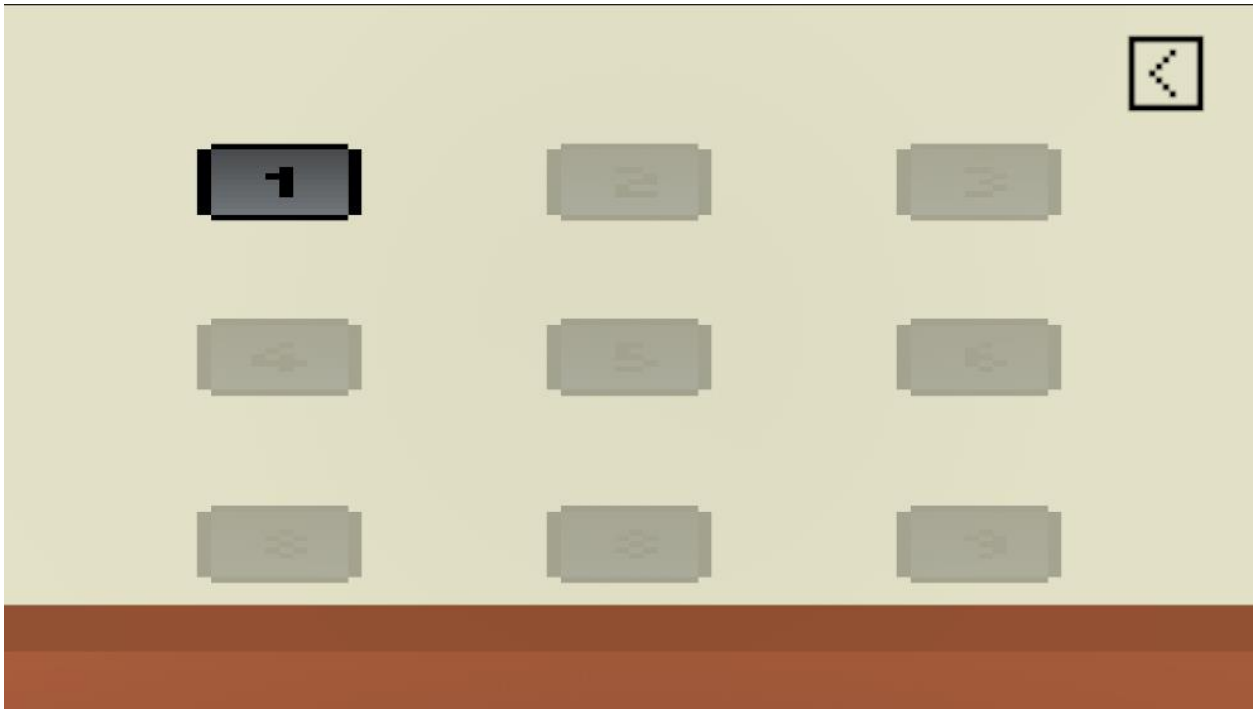

Рисунок 27 – Окно выбора уровня

Далее нужно написать скрипт для кнопки «Выйти из игры». Скрипт представлен в Приложении К.

Теперь после нажатия кнопки «Выйти из игры» в главном меню игры, она закроется.

Для музыкального сопровождения главного меню был написан саундтрек в программе FL Studio и успешно добавлен в сцену проекта. Для этого необходимо создать пустой объект в окне Hierarchy, в свойства которого поместить саундтрек. Результат представлен на рисунке 28.

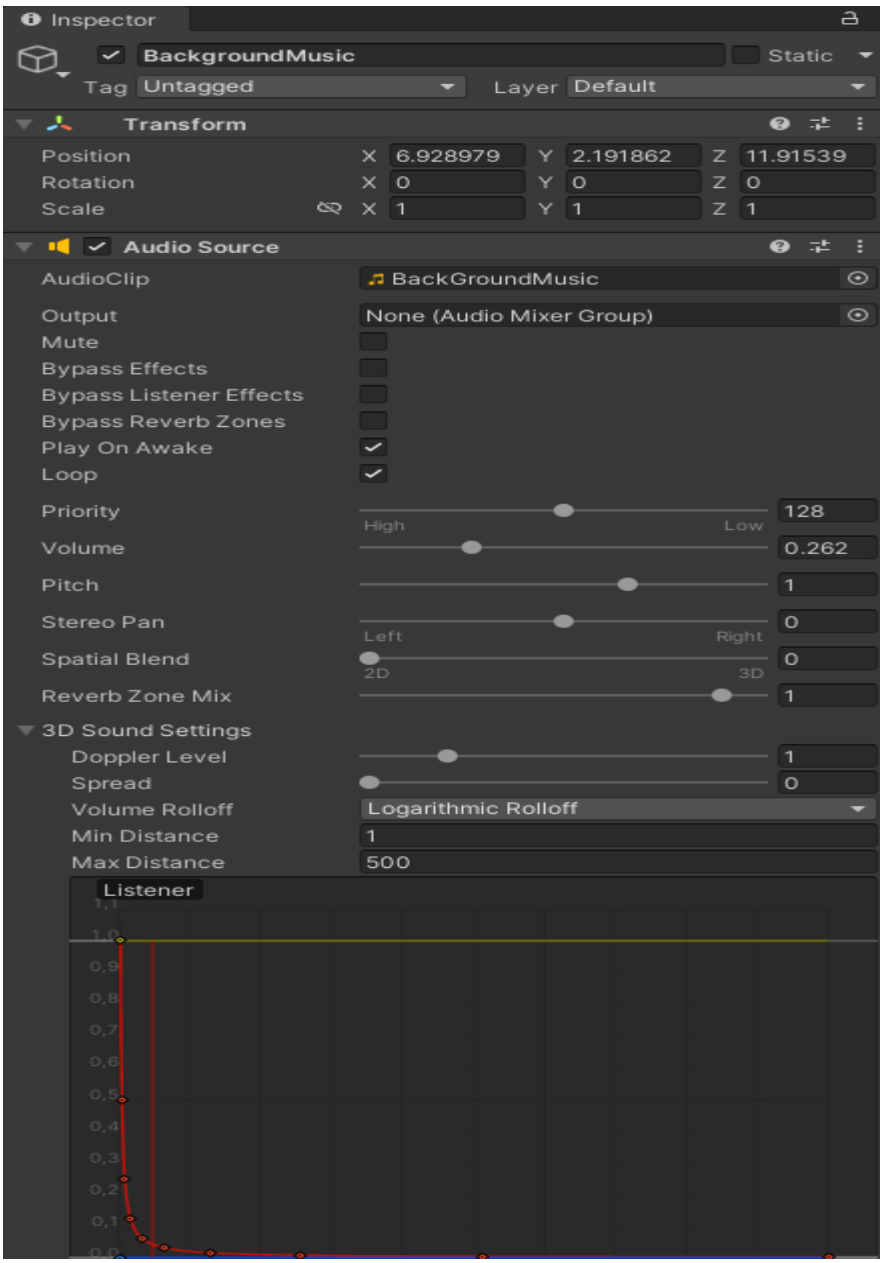

Рисунок 28 – Свойства фоновой музыки

В данном параграфе было разработано функциональное главное меню, содержащее в себе панель главного меню и панель выбора уровней, также была добавлена кнопка выхода из игры.

2.6 Функциональное тестирование

Функциональное тестирование – это процесс проверки функциональности разрабатываемого продукта или системы с целью

убеждения в работоспособности согласно заданным требованиям и ожиданиям. Оно является частью обеспечения качества выпускаемого продукта.

Результаты функционального тестирования компьютерной 3D игры для развития пространственного мышления описаны в таблице 2.

Таблица 2 – Функциональное тестирование

№	Тест	Действие	Ожидаемый результат	Полученный результат	Итог
1	Запуск игры	Запустить игру	Игра запустится	Игра запустилась	Пройдено
2	Проверка кнопки «Выбрать уровень»	Нажать на кнопку «Выбрать уровень»	Смена сцены на выбор уровня	Смена сцены на выбор уровня	Пройдено
3	Проверка запуска доступного уровня	Запустить доступный уровень	Запуск доступного уровня	Запуск доступного уровня	Пройдено
4	Проверка запуска недоступного уровня	Запустить заблокированный уровень	Ничего не произойдет	Ничего не произошло	Пройдено
5	Проверка кнопки «Выход из игры»	Нажать на кнопку «Выход из игры»	Игра закроется	Игра закрылась	Пройдено
6	Проверка работы триггеров завершения уровня	Завершить уровень	Открывается окно завершения уровня	Окно завершения уровня открылось	Пройдено
7	Проверка скрипта проигрыша	Дождаться истечения времени на прохождение уровня	Уровень завершится с окном «Время вышло»	Уровень завершился с окном «Время вышло»	Пройдено
8	Сохранение прогресса	При выходе из игры, игра сохраняет прогресс, пройденный игроком	Разблокированные уровни сохраняются после перезапуска игры	Разблокированные уровни доступны после перезапуска игры	Пройдено

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы была создана компьютерная 3D игра для развития пространственного мышления, в жанре головоломка с элементами лабиринта. Основная цель работы — разработка компьютерной 3D игры с использованием игрового движка Unity — была успешно достигнута. Так же были выполнены поставленные задачи, такие как:

– на основе анализа учебной и научно-технической литературы по теме исследования был определен понятийный аппарат и сущность данного жанра;

– рассмотрены этапы разработки и планирования компьютерной игры в жанре головоломка-лабиринт;

– разработана и протестирована компьютерная игра.

В ходе разработки компьютерной 3D игры для развития пространственного мышления были изучены следующие возможности движка Unity:

– создание проекта;

– создание сцены;

– создание анимации;

– настройка свойств объектов;

– создание UI элементов;

– возможность настройки звукового сопровождения сцены.

Разработанная в рамках выпускной квалификационной работы игра представляет собой эффективное средство для развития навыков пространственного мышления у пользователей старше 6 лет. Она может быть использована как в рамках образовательных учреждений, так и для самостоятельного обучения. Данная игра сочетает в себе развлекательные

элементы и образовательные задачи, что делает процесс обучения интересным и увлекательным.

В будущем планируется продолжить работу над обновлением игровых механик, добавление более сложных уровней и адаптации задач под индивидуальные особенности пользователей. Также рассматривается вариант создания мобильной версии для расширения аудитории пользователей и большего охвата поддерживаемых устройств.

Данная выпускная квалификационная работа позволила получить ценные знания в области разработки образовательных компьютерных 3D игр для развития пространственного мышления.

Результаты исследования были представлены на следующих научных мероприятиях:

1. VII Всероссийская научно-практической конференция «Актуальные проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 1-2 ноября 2023 г.).

2. III Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 9 апреля 2024 г.).

По результатам исследования опубликована следующая статья: Русов В. Д. / Игровые технологии для развития пространственного мышления: как компьютерные игры могут помочь в обучении / В. Д. Русов / В. Д. Русов // Информационные технологии в образовании, науке и производстве: материалы III Всероссийского молодежного научного форума «Современное педагогическое образование: теоретический и прикладной аспекты» / ЛПИ – филиал СФУ. – Лесосибирск, 2024.

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: 29.10.2023)

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

3. Богачева, Н. В. Компьютерные игры и психологическая специфика когнитивной сферы геймеров / Н. В. Богачева // Вестник Московского университета. Серия 14: Психология. – 2014. – № 4. – С. 120–130 ; 2015. – № 1. – С. 94–103. – URL: https://clck.ru/3BMk8C ; https://clck.ru/3BMkBJ (дата обращения: 19.11.2023).

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

5. Бонд, Д. Г. Unity и C# Геймдев от идеи до реализации / Д. Г. Бонд. – Санкт-Петербург: Питер, 2019. — 928 с.

6. Виды компьютерных игр / URL: https://wobla.ru/infomat/games.aspx# (дата обращения: 13.11.2023).

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

8. Головоломка (жанр компьютерных игр) / URL: https://ru.wikipedia.org /wiki/Головоломка_(жанр_компьютерных_игр) (дата обращения: 10.10.2023).

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. 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: 10.11.2023).

11. Кудряшов, И. С. Игровой движок // Большая российская энциклопедия: научно-образовательный портал. – 2023. – 8 сент. – URL: https://bigenc.ru/c/igrovoi-dvizhok-6b6ec7 (дата обращения 22.12.2023).

12. Кутлалиев, Т. Х. Использование реквизита в компьютерных играх: к вопросу об эстетических свойствах виртуальности // Гуманитарные и социальные науки. 2013. №4. URL: https://cyberleninka.ru/article/n/ispolzovanie-rekvizita-v-kompyuternyh-igrah-k-voprosu-ob-esteticheskih-svoystvah-virtualnosti (дата обращения: 19.06.2024).

13. 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.

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

15. ЛогикЛайк : официальный сайт. – 2024. – URL: https://logiclike.com/ (дата обращения 19.02.2024).

16. Майерс, Г. Искусство тестирования программ / М. Майерс: ЛитРес: Самиздат, 2018.

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

18. О Godot Engine / URL: https://docs.godotengine.org/ru/stable/about/introduction.html# (дата обращения: 10.04.2024).

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

20. Перельман, Я. И. Дом занимательной науки: Лабиринты / Я. И. Перельман. – Проспект, 2022. — 32 с.

21. Платов, В. Я. Деловые игры: разработка, организация, проведение /

В. Я. Платов. – Москва: Профиздат, 1991. – 192 с. – ISBN 5-255-00129-5.

22. Подвальный, М. А. Видеоигра // Большая российская энциклопедия: научно-образовательный портал. – 2023. – 28 марта – URL: https://bigenc.ru/c/videoigra-915272/?v=6558368 (дата обращения 19.02.2024).

23. 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: 09.11.2023)

24. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование разработка/ Дж. Рамбо. — СПб.: Питер, 2007. – 544 с.

25. 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.

26. Скоморох М. М. Компьютерные игры и утопия интерактивности: на что способны геймеры? // Международный журнал исследований культуры. 2014. №2 (15). URL: https://cyberleninka.ru/article/n/kompyuternye-igry-i-utopiya-interaktivnosti-na-chto-sposobny-geymery (дата обращения: 19.02.2024).

27. Соснина, Н. Развитие пространственного мышления у дошкольников: задания, игры и упражнения / Н. Соснина // Узелки. Учиться интересно! : [сайт]. – 2022. – 8 сент. – URL: https://uzelki.com/razvitie-prostranstvennogo-myshlenia/ (дата обращения 09.11.2023).

28. Торн, А. Основы анимации в Unity: учебное пособие/ А. Торн – ред.: Д. Мовчан, переводчик: Р. Рагимов. – Москва: ДМК, 2016 – 176с.

29. Хокинг, Д. Unity в действии. Мультиплатформенная разработка на C#: учебное пособие/ Джозеф Хокинг – Санкт-Петербург: Питер, 2016 – 336с.

30. Что такое Visual Studio? / URL: https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022 (дата обращения: 10.02.2024).

31. Что такое игровой движок? / URL: https://club.dns-shop.ru/blog/t-64-videoigryi/34701-chto-takoe-igrovoi-dvijok/ (дата обращения: 20.11.2023).

32. C# Programming Guide. URL: https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx (дата обращения: 15.08.2023).

33. FLStudio: официальный сайт / Image-Line, 2023 – . – URL: https://www.image-line.com/ (дата обращения: 20.12.2023).

34. GIMP: официальный сайт / Creative Commons, 2023 – . – / URL: https://www.gimp.org/downloads/ (дата обращения: 15.09.2023).

35. Godot: официальный сайт / Software Freedom Conservancy, 2023 – . – / URL: https://godotengine.org/download/windows/ (дата обращения: 15.09.2023).

36. Unity : официальный сайт / Unity Technologies, 2023 – . – URL: https://unity.com/download (дата обращения: 15.09.2023).

37. 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).

38. 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.

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

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

41. Visual Studio: официальный сайт / Microsoft, 2023 – . – URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 15.09.2023).

Приложение А

Листинг кода C#, отвечающего за положение аксессуаров на персонаже

```
using UnityEngine;
namespace Supercyan.FreeSample
{
   public class AccessoryWearLogic : MonoBehaviour
   {
      [SerializeField] private SkinnedMeshRenderer m_characterRenderer;

      [SerializeField] private AccessoryLogic[] m_accessoriesToAttach = null;

      private Transform[] m_characterBones;

      private bool m_initialized = false;

      private void Initialize(GameObject character)
      {
         if (m_characterRenderer == null)
         {
            m_characterRenderer =
GetComponentInChildren<SkinnedMeshRenderer>();

            if (m_characterRenderer == null)
            {
               Debug.LogWarning("Missing character components.");
               return;
            }
         }
         if (m_characterRenderer.rootBone == null)
```

```
            {
                Debug.LogWarning("Missing character root bone.");
                return;
            }
            m_characterBones = m_characterRenderer.bones;
            m_initialized = true;
        }
        private void Awake()
        {
            Initialize(gameObject);
            foreach (AccessoryLogic a in m_accessoriesToAttach) { Attach(a); }
        }
        public void Attach(AccessoryLogic accessory)
        {
            if (!m_initialized)
            {
                Initialize(gameObject);
                if (!m_initialized)
                {
                    Debug.LogWarning("AccessoryWearLogic not initialized correctly,
can't attach accessories.");
                    return;
                }
            }
            else if (accessory == null)
            {
                Debug.LogWarning("Trying to attach null accessory.");
                return;
            }
            else if (accessory.Renderer == null)
```

```
            {
                Debug.LogWarning("Trying to attach accessory with missing
renderer.");
                return;
            }
            else if (accessory.Renderer.rootBone == null)
            {
                Debug.LogWarning("Trying to attach accessory with missing root
bone.");
                return;
            }
            Transform[] newBones = GetBones(accessory.Renderer.bones,
m_characterBones);
            if (newBones == null)
            {
                Debug.LogWarning("Trying to attach accessory with incompatible
rig.");
                return;
            }
            accessory.Renderer.bones = newBones;
            accessory.Renderer.rootBone = m_characterRenderer.rootBone;
        }
        private Transform[] GetBones(Transform[] accessoryBones, Transform[]
characterBones)
        {
            Transform[] newBones = new Transform[accessoryBones.Length];
            for (int i = 0; i < accessoryBones.Length; i++)
            {
                Transform bone = FindBone(m_characterRenderer.rootBone,
accessoryBones[i].name);
```

```
            if (bone == null) { return null; }
            newBones[i] = bone;
        }

        return newBones;
    }
    private Transform FindBone(Transform rootBone, string name)
    {
        if (rootBone.name == name) { return rootBone; }
        else
        {
            Transform found = null;
            for (int i = 0; i < rootBone.childCount; i++)
            {
                found = FindBone(rootBone.GetChild(i), name);
                if (found != null) { return found; }
            }
            return null;
        }
    }
  }
}
```

Приложение Б

Листинг кода C#, отвечающего за работу персонажа

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

namespace Supercyan.FreeSample
{
   public class SimpleSampleCharacterControl : MonoBehaviour
   {
      private enum ControlMode
      {
         Tank,
         Direct
      }

      [SerializeField] private float m_moveSpeed = 2;
      [SerializeField] private float m_turnSpeed = 200;
      [SerializeField] private float m_jumpForce = 4;

      [SerializeField] private Animator m_animator = null;
      [SerializeField] private Rigidbody m_rigidBody = null;

      [SerializeField] private ControlMode m_controlMode =
ControlMode.Direct;

      private float m_currentV = 0;
      private float m_currentH = 0;

      private readonly float m_interpolation = 10;
```

```
private readonly float m_walkScale = 0.33f;
private readonly float m_backwardsWalkScale = 0.16f;
private readonly float m_backwardRunScale = 0.66f;

private bool m_wasGrounded;
private Vector3 m_currentDirection = Vector3.zero;

private float m_jumpTimeStamp = 0;
private float m_minJumpInterval = 0.25f;
private bool m_jumpInput = false;

private bool m_isGrounded;

private List<Collider> m_collisions = new List<Collider>();

private void Awake()
{
   if (!m_animator) { gameObject.GetComponent<Animator>(); }
   if (!m_rigidBody) { gameObject.GetComponent<Animator>(); }
}

private void OnCollisionEnter(Collision collision)
{
   ContactPoint[] contactPoints = collision.contacts;
   for (int I = 0; I < contactPoints.Length; i++)
   {
      if (Vector3.Dot(contactPoints[i].normal, Vector3.up) > 0.5f)
      {
         if (!m_collisions.Contains(collision.collider))
         {
```

```
                m_collisions.Add(collision.collider);
            }
            m_isGrounded = true;
        }
    }
}

private void OnCollisionStay(Collision collision)
{
    ContactPoint[] contactPoints = collision.contacts;
    bool validSurfaceNormal = false;
    for (int I = 0; I < contactPoints.Length; i++)
    {
        if (Vector3.Dot(contactPoints[i].normal, Vector3.up) > 0.5f)
        {
            validSurfaceNormal = true; break;
        }
    }

    if (validSurfaceNormal)
    {
        m_isGrounded = true;
        if (!m_collisions.Contains(collision.collider))
        {
            m_collisions.Add(collision.collider);
        }
    }
    else
    {
        if (m_collisions.Contains(collision.collider))
```

```
        {
            m_collisions.Remove(collision.collider);
        }
        if (m_collisions.Count == 0) { m_isGrounded = false; }
    }
}

private void OnCollisionExit(Collision collision)
{
    if (m_collisions.Contains(collision.collider))
    {
        m_collisions.Remove(collision.collider);
    }
    if (m_collisions.Count == 0) { m_isGrounded = false; }
}

private void Update()
{
    if (!m_jumpInput && Input.GetKey(KeyCode.Space))
    {
        m_jumpInput = true;
    }
}

private void FixedUpdate()
{
    m_animator.SetBool("Grounded", m_isGrounded);

    switch (m_controlMode)
    {
```

```
        case ControlMode.Direct:
            DirectUpdate();
            break;

        case ControlMode.Tank:
            TankUpdate();
            break;

        default:
            Debug.LogError("Unsupported state");
            break;
    }

    m_wasGrounded = m_isGrounded;
    m_jumpInput = false;
}

private void TankUpdate()
{
    float v = Input.GetAxis("Vertical");
    float h = Input.GetAxis("Horizontal");

    bool walk = Input.GetKey(KeyCode.LeftShift);

    if (v < 0)
    {
        if (walk) { v *= m_backwardsWalkScale; }
        else { v *= m_backwardRunScale; }
    }
    else if (walk)
```

```
            {
                v *= m_walkScale;
            }

            m_currentV = Mathf.Lerp(m_currentV, v, Time.deltaTime *
m_interpolation);
            m_currentH = Mathf.Lerp(m_currentH, h, Time.deltaTime *
m_interpolation);

            transform.position += transform.forward * m_currentV *
m_moveSpeed * Time.deltaTime;
            transform.Rotate(0, m_currentH * m_turnSpeed * Time.deltaTime,
0);

            m_animator.SetFloat("MoveSpeed", m_currentV);

            JumpingAndLanding();
        }

        private void DirectUpdate()
        {
            float v = Input.GetAxis("Vertical");
            float h = Input.GetAxis("Horizontal");

            Transform camera = Camera.main.transform;

            if (Input.GetKey(KeyCode.LeftShift))
            {
                v *= m_walkScale;
                h *= m_walkScale;
```

```
            }

            m_currentV = Mathf.Lerp(m_currentV, v, Time.deltaTime *
m_interpolation);
            m_currentH = Mathf.Lerp(m_currentH, h, Time.deltaTime *
m_interpolation);

            Vector3 direction = camera.forward * m_currentV + camera.right *
m_currentH;

            float directionLength = direction.magnitude;
            direction.y = 0;
            direction = direction.normalized * directionLength;

            if (direction != Vector3.zero)
            {
                m_currentDirection = Vector3.Slerp(m_currentDirection,
direction, Time.deltaTime * m_interpolation);

                transform.rotation =
Quaternion.LookRotation(m_currentDirection);
                transform.position += m_currentDirection * m_moveSpeed *
Time.deltaTime;

                m_animator.SetFloat("MoveSpeed", direction.magnitude);
            }

            JumpingAndLanding();
        }
```

```
        private void JumpingAndLanding()
        {
            bool jumpCooldownOver = (Time.time – m_jumpTimeStamp) >=
m_minJumpInterval;

            if (jumpCooldownOver && m_isGrounded && m_jumpInput)
            {
                m_jumpTimeStamp = Time.time;
                m_rigidBody.AddForce(Vector3.up * m_jumpForce,
ForceMode.Impulse);
            }
        }
    }
```

Приложение В

Листинг кода C#, отвечающего за работу таймера

```
using UnityEngine;
using UnityEngine.UI;
public class Timer : MonoBehaviour
{
   public float timeRemaining = 60f; // Начальное время в секундах
   public bool timerIsRunning = false; // Флаг для запуска и остановки таймера
   public Text timeText; // UI текст для отображения времени
   public GameObject gameOverPanel; // Панель для отображения сообщения о проигрыше
   void Start()
   {
      // Начать таймер
      timerIsRunning = true;
      gameOverPanel.SetActive(false); // Скрыть панель в начале игры
      Time.timeScale = 1; // Убедиться, что время идет нормально в начале игры
   }
   void Update()
   {
      if (timerIsRunning)
      {
         if (timeRemaining > 0)
         {
            timeRemaining -= Time.deltaTime; // Уменьшить оставшееся время
            DisplayTime(timeRemaining); // Обновить отображение времени
         }
         else
```

```
            {
                Debug.Log("Время вышло!");
                timeRemaining = 0;
                timerIsRunning = false;
                GameOver(); // Показать панель проигрыша и остановить игру
            }
        }
    }
    // Функция для отображения времени на UI
    void DisplayTime(float timeToDisplay)
    {
        if (timeToDisplay < 0)
        {
            timeToDisplay = 0;
        }
        int minutes = Mathf.FloorToInt(timeToDisplay / 60); // Подсчет минут
        int seconds = Mathf.FloorToInt(timeToDisplay % 60); // Подсчет секунд
        // Обновить текстовое поле времени
        timeText.text = string.Format("{0:00}:{1:00}", minutes, seconds);
    }
    // Функция для обработки окончания времени
    void GameOver()
    {
        gameOverPanel.SetActive(true); // Отобразить панель с сообщением о
проигрыше
        Time.timeScale = 0; // Остановить время в игре
    }
}
```

Приложение Г

Листинг кода C#, отвечающего за работу панели LevelCompletePanel

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class LevelComplete : MonoBehaviour
{
    public GameObject levelCompletePanel; // Панель для отображения при завершении уровня

    void Start()
    {
        levelCompletePanel.SetActive(false);
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            levelCompletePanel.SetActive(true);
            Time.timeScale = 0;
        }
    }

    public void LoadNextLevel()
    {
        string nextLevelName = "Level" + (SceneManager.GetActiveScene().buildIndex + 1);
        Time.timeScale = 1;
```

```
        SceneManager.LoadScene(nextLevelName);
    }

    // Функция для возврата в главное меню
    public void LoadMainMenu()
    {
        Time.timeScale = 1; // Сбросить скорость времени перед загрузкой главного меню
        SceneManager.LoadScene("MainMenu");
    }
}
```

Приложение Д

Листинг кода C#, отвечающего за работу кнопки «Заново»

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class RetryButton : MonoBehaviour
{
    public void Retry()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);             //
Перезагрузка текущей сцены
    }
}
```

Приложение Е

Листинг кода C#, отвечающего за работу кнопки «В главное меню»

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ExitToMenu : MonoBehaviour
{
    public void PlayGame()
    {
        SceneManager.LoadScene("MainMenu");
    }
    public void QuitGame()
    {
        Debug.Log("Quit Game");
        Application.Quit();
    }
}
```

Приложение И

Листинг кода C#, отвечающего за загрузку уровней

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour
{
    public Button level1;
    public Button level2;
    public Button level3;
    public Button level4;
    public Button level5;
    public Button level6;
    public Button level7;
    public Button level8;
    public Button level9;
    int levelComplete;

    void Start()
    {
        levelComplete = PlayerPrefs.GetInt("LevelComplete");
        level2.interactable = false;
        level3.interactable = false;
        level4.interactable = false;
        level5.interactable = false;
        level6.interactable = false;
        level7.interactable = false;
        level8.interactable = false;
```

```
level9.interactable = false;

switch (levelComplete)
{
    case 1:
        level2.interactable = true;
        break;
    case 2:
        level2.interactable = true;
        level3.interactable = true;
        break;
    case 3:
        level3.interactable = true;
        level4.interactable = true;
        break;
    case 4:
        level4.interactable = true;
        level5.interactable = true;
        break;
    case 5:
        level5.interactable = true;
        level6.interactable = true;
        break;
    case 6:
        level6.interactable = true;
        level7.interactable = true;
        break;
    case 7:
        level7.interactable = true;
        level8.interactable = true;
```

```
                break;
            case 8:
                level8.interactable = true;
                level9.interactable = true;
                break;
        }
    }

    public void LoadTo(int level)
    {
        SceneManager.LoadScene(level);
    }
    void Update()
    {

    }
}
```

Приложение К

Листинг кода C#, отвечающего за работу кнопки «Выход из игры»

```
using UnityEngine;

public class ExitGame : MonoBehaviour
{
    public void QuitGame()
    {
        Application.Quit();
    }
}
```