ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ/INFORMATICS AND INFORMATION PROCESSES

DOI: https://doi.org/10.60797/IRJ.2025.162.140

РАЗРАБОТКА ОБУЧАЮЩЕЙ 3D-СРЕДЫ ДЛЯ ФОРМИРОВАНИЯ ПРАКТИЧЕСКИХ НАВЫКОВ КОНФИГУРАЦИИ ЛОКАЛЬНЫХ СЕТЕЙ

Научная статья

Белашова Е.С.[1, *], Маршалова И.Н.[2], Гашигуллин Д.А.[3], Кремлева Э.Ш.[4]
[1] ORCID : 0000-0003-4662-185X;
[4] ORCID : 0000-0003-0858-0575;
[1, 3, 4] Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань, Российская Федерация
[2] Основная общеобразовательная школа №17 имени Героя Советского Союза Н.А. Катина, Зеленодольск, Российская Федерация

* Корреспондирующий автор (bel_lena[at]mail.ru)

Аннотация

В статье рассматривается разработка интерактивной трехмерной платформы, направленной на обучение построению и конфигурации локальных сетей с применением игровых технологий. Использование игрового движка Unity позволило создать гибкую модульную среду, сочетающую наглядность визуализации, элементы геймификации и практико-ориентированный подход. Разработанное решение предоставляет возможность пошагового освоения теоретических знаний и практических умений, обеспечивает вовлечение пользователей и формирует устойчивые навыки работы с сетевой инфраструктурой. Практическая значимость платформы заключается в возможности её применения в образовательных учреждениях, корпоративной подготовке специалистов и самостоятельном онлайн-обучении. Представленные технические решения демонстрируют перспективность интеграции игровых технологий в процесс преподавания информационно-технологических дисциплин.

Ключевые слова: интерактивное обучение, 3D-платформа, локальные сети, геймификация, Unity, образовательные технологии, сетевые симуляторы, практическое обучение, компьютерные сети, визуализация, игровые механики, цифровое образование.

DEVELOPMENT OF A 3D TRAINING ENVIRONMENT FOR DEVELOPING PRACTICAL SKILLS IN CONFIGURING LOCAL AREA NETWORKS

Research article

Belashova E.S.[1, *], Marshalova I.N.[2], Gashigullin D.A.[3], Kremleva E.S.[4]
[1] ORCID : 0000-0003-4662-185X;
[4] ORCID : 0000-0003-0858-0575;
[1, 3, 4] Kazan National Research Technical University named after A.N. Tupolev – KAI, Kazan, Russian Federation
[2] Basic Secondary School No. 17 named after Hero of the Soviet Union N.A. Katin, Zelenodolsk, Russian Federation

* Corresponding author (bel_lena[at]mail.ru)

Abstract

The article examines the development of an interactive three-dimensional platform aimed at teaching the construction and configuration of local networks using gaming technologies. The use of the Unity game engine has made it possible to create a flexible modular environment that combines clear visualisation, gamification elements and a practice-oriented approach. The developed solution provides the opportunity for step-by-step mastery of theoretical knowledge and practical skills, ensures user engagement, and forms sustainable skills for working with network infrastructure. The practical significance of the platform lies in its applicability in educational institutions, corporate training of specialists, and independent online learning. The technical solutions presented demonstrate the promise of integrating gaming technologies into the teaching of information technology disciplines.

Keywords: interactive learning, 3D platform, local networks, gamification, Unity, educational technologies, network simulators, practical training, computer networks, visualisation, game mechanics, digital education.

Введение

В современном образовательном пространстве наблюдается устойчивый рост интереса к интерактивным технологиям, способным упростить освоение сложных дисциплин. Одной из таких дисциплин является построение и конфигурирование локальных сетей, играющих важную роль в деятельности современных организаций и предприятий. Однако традиционные средства обучения часто оказываются недостаточно эффективными для студентов, особенно в условиях формирования клипового мышления, когда внимание человека удерживается лишь на короткий промежуток времени и восприятие информации носит поверхностный характер. В связи с этим актуальным становится внедрение геймифицированных подходов, которые позволяют компенсировать указанные ограничения и повышают мотивацию обучающихся [1].

Интерактивная трехмерная платформа представляет собой средство обучения, в котором сложные абстрактные концепции сетевого взаимодействия преобразуются в наглядные визуальные образы. Такой подход способствует реализации принципа наглядности и обеспечивает возможность поэтапного освоения материала с постепенным

усложнением заданий. В результате учащиеся не только лучше понимают теоретические основы сетевых технологий, но и получают практический опыт работы с виртуальными объектами, моделирующими реальные элементы сетевой инфраструктуры. Образовательный эффект усиливается благодаря элементам игры, включающим систему вознаграждений и уровней, что стимулирует интерес и поддерживает высокий уровень вовлеченности.

Для реализации подобной платформы был выбран игровой движок Unity, который предоставляет широкий спектр возможностей для создания трехмерных образовательных сред. Его применение обусловлено кроссплатформенностью, развитой экосистемой и активным сообществом разработчиков, наличием инструментов для работы с трехмерной графикой, поддержкой сетевых режимов, а также оптимизацией рабочего процесса и экономической эффективностью. Использование Unity позволяет создавать модульное программное обеспечение, расширяемое за счет добавления новых заданий и сценариев, что делает проект перспективным как в образовательной, так и в профессиональной сфере [2].

В отличие от существующих решений [3], [4], [5], [6], ориентированных преимущественно на управление сетями, разработанная платформа направлена именно на обучение пользователей с минимальным уровнем предварительных знаний. В ней предусмотрено предоставление базового теоретического материала, необходимого для понимания принципов построения сетей, а также практических заданий различной сложности, выполняемых в интерактивном виртуальном пространстве. Такая организация процесса обеспечивает постепенное погружение в предметную область и способствует формированию у студентов устойчивых навыков конфигурации и диагностики локальных сетей.

Практическая ценность разработанного программного обеспечения заключается в его многофункциональности. Оно может быть интегрировано в лабораторные работы для студентов информационно-технологических специальностей, использоваться в корпоративной подготовке сетевых администраторов, а также применяться в формате самостоятельного онлайн-обучения. Техническим результатом проекта является создание открытого программного продукта с модульной архитектурой, позволяющей преподавателям и специалистам адаптировать платформу под конкретные образовательные цели и расширять ее возможности. Таким образом, интерактивная трехмерная платформа становится эффективным инструментом для подготовки квалифицированных специалистов в области сетевых технологий, объединяя в себе теоретическую основу, практическую направленность и современные игровые подходы к обучению.

Анализ существующих аналогов приложения

В настоящее время для симуляции работы локальных сетей и для обучения другим IT-технологиям уже используются геймифицированные решения. Ниже рассмотрены три наиболее подходящих решений, которые можно взять как отправную точку при разработке интерактивного симулятора освоения локальных сетей.

Cisco Packet Tracer — это профессиональный сетевой симулятор от компании Cisco. Его широко применяют в учебных заведениях [7]. Он позволяет проектировать и конфигурировать виртуальные сети, используя виртуальные маршрутизатору, коммутаторы и другие устройства. Взаимодействовать с ними можно через командную строку. Интерфейс данного приложения можно посмотреть на рисунке 1.

Ключевыми возможностями данного приложения являются следующие:

1) поддержка всех сетевых протоколов TCP/IP;

2) режим работы в реальном времени или в режиме симуляции, что позволяет анализировать обмен данными пошагово;

3) кроссплатформенность;

4) визуализация передачи пакетов и подробных лог взаимодействия протоколов.

Из преимуществ данного приложения можно выделить следующие:

1) оно позволяет обучающимся моделировать сложные сетевые конфигурации без физического оборудования;

2) служит хорошей площадкой для практики командной строки и отладки реальных сценариев.

Недостатки Cisco Packet Tracer:

1) ориентирован на оборудование Cisco;

2) в приложении полностью отсутствуют игровые механики, что может снизить пользовательской вовлеченности и мотивации;

3) отсутствие интуитивно понятного интерфейса.

Cisco Packet Tracer имеет ряд ограничений для новичков. Во-первых, минимальные возможности визуализации сетевых механизмов в трехмерном пространстве затрудняют изучение физической топологии сетей. Во-вторых, отсутствие пошагового обучения с помощью геймификации делает самообучение менее мотивирующим. В-третьих, достаточно сложный интерфейс настройки оборудований затрудняет использование приложения. Инструмент по-прежнему остается полезным при подготовке к сертификации Cisco CCNA, поскольку он точно имитирует поведение реального оборудования. Такие особенности выводят это приложение на идеальный уровень для продвинутых пользователей и создают барьеры для обучения неподготовленных новичков.

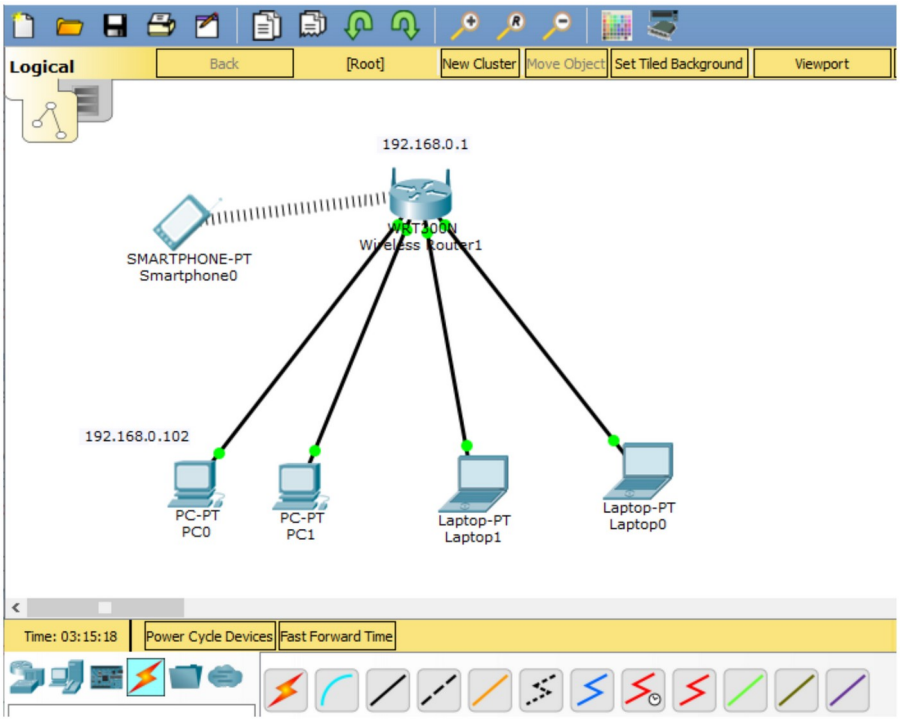

Рисунок 1 - Интерфейс приложения Cisco Packet Tracer
DOI: https://doi.org/10.60797/IRJ.2025.162.140.1

Hacknet — необычный симулятор хакерской деятельности с ярко выраженным нарративом. Пользователи получают задачу по взлому систем и анализу файлов через командную строку, очень напоминающую Unix [8]. Интерфейс игры можно увидеть на рисунке 2.

Из ключевых особенности данной игры можно выделить:

1) реалистичная командная строка;
2) интересная сюжетная линия;
3) поддержка пользовательских сценариев.

Преимущества:

1) учёт логики работы консоли и базовых утилит Unix;
2) сюжет повышает интерес и мотивацию.

Недостатки:

1) требует определённых навыков работы с командной строкой;
2) минимальная визуализация;
3) недостаток ощущения окружения из-за отсутствия 3D визуализации;
4) отсутствует обучение локальным сетям.

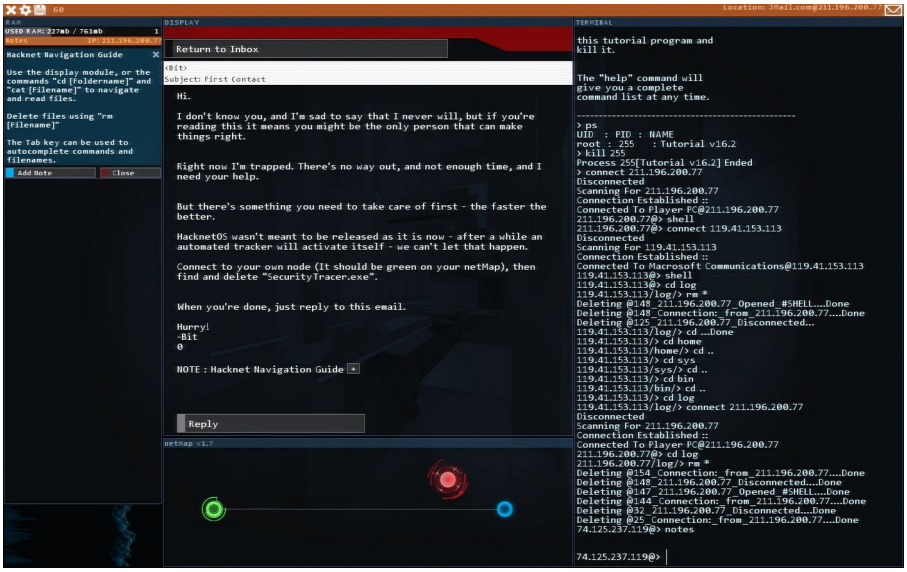

Рисунок 2 - Интерфейс игры Hacknet
DOI: https://doi.org/10.60797/IRJ.2025.162.140.2

while True: learn() — приложение головоломка, посвящённая основам машинного обучения [9]. В данной игре сюжет предлагает рассмотреть жизнь разработчика и его кота. Обучение происходит через визуальные блок-схемы. Интерфейс данной головоломки представлен на рисунке 3.

Ключевые особенности данной игры такие:

1) визуальное программирование с помощью блок-схем;
2) пошаговое введение в методы классификации, нейросети и обработку данных;
3) кастомизация рабочего интерфейса;
4) геймификация с юмором.

Из преимуществ данной видеоигры можно выделить:

1) доступность для пользователей без технических навыков;
2) интуитивная визуализация сложных алгоритмов.

Недостатки заключаются в следующем:

1) отсутствие изучение локальных сетей;
2) поверхностный охват тем.

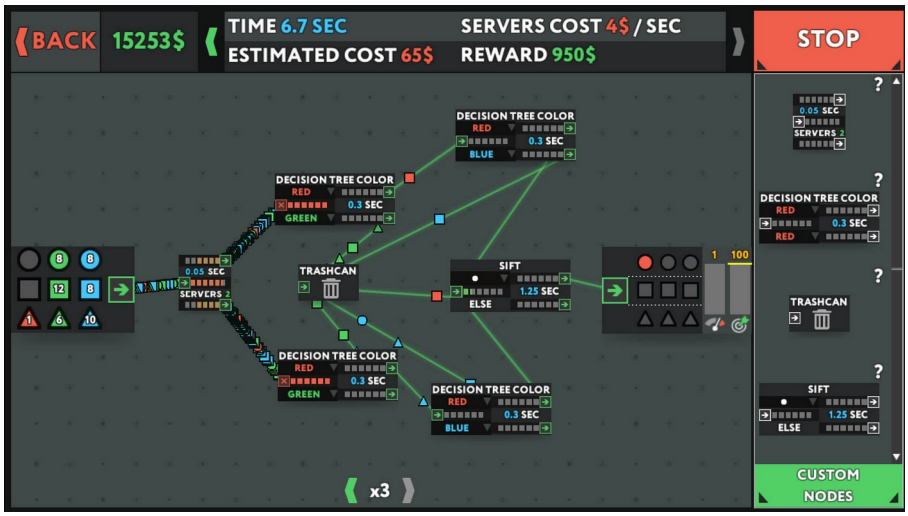

Рисунок 3 - Интерфейс игры while True: learn()
DOI: https://doi.org/10.60797/IRJ.2025.162.140.3

Каждое из рассмотренных приложений предлагает уникальный подход к обучению техническим навыкам. Для разработки интерактивной 3D-игры, направленной на обучение созданию локальных сетей, можно объединить лучшие элементы этих приложений:

1) реалистичную симуляцию сетей из Cisco Packet Tracer;
2) геймифицированный подход из Hacknet;
3) визуальное и интуитивное обучение из while True: learn().

Проектирование платформы

Современный мир игровых технологий имеет обширное множество игровых движков. Некоторые из них пользуются популярностью, а некоторые почти неизвестны. Игровые индустрии либо используют готовые игровые движки для своих игр, либо создают собственные. Так австралийская команда разработчиков для своей очень популярной 2D видеоигры использовали готовый движок Unity. А американская организация Rockstar для своих игр создавали свой отдельный движок — RAGE Engine, что требует больших затрат ресурсов, но позволяет полностью контролировать технологию [10], [11].

В видеоиграх очень большую роль играет выбранный игровой движок. Как отмечает автор множества книг про программирование А.Н. Баланов, игровой движок — это программное обеспечение, которое облегчает процесс разработки видеоигр, содержащий обширный набор готовых инструментов, библиотек и ресурсов для создания игровых приложений [12], [13]. Обычно в набор входят следующие инструменты:

1) графический рендеринг;
2) физический движок;
3) аудиосистема;
4) анимации;
5) средства программирования.

Индустрия разработки видеоигр сегодня предлагает широкий спектр готовых игровых движков. Каждый из них обладает своими уникальными характеристиками. Разработчики выбирают среду создания видеоигр исходя из нескольких критериев своего проекта.

Для разработки своей игры был выбран игровой движок Unity 6. Unity является одной из самых популярных игровых движков в индустрии. Выбор обусловлен следующими факторами:

1) кроссплатформенность;
2) интегрированная среда разработки:
− интуитивный редактор с визуальным построением сцен;
− встроенные системы;
− графические конвейеры.
3) экосистема Asset Store, содержащая готовые модели игровых элементов, плагины для программирования, инструменты UI;
4) язык программирования C#.

Для создания пользовательского интерфейса в проекте используется UI Toolkit — встроенная в Unity технология на основе UI Document и USS (Unity Style Sheets) [14], [15]. Преимущества выбора UI Toolkit заключают в следующих факторах:
1) оптимизация производительности:
− работает быстрее Canvas, так как не требует постоянного перестроения интерфейса;
− использует декларативный подход, аналогичный веб-разработке, что снижает нагрузку на CPU.
2) Гибкость и масштабируемость:
− поддержка стилей USS, что позволяет легко менять внешний вид элементов без правки кода;
− возможность динамической генерации UI через C#;
− адаптация под разные разрешения экрана.

Для реализации плавной и адаптируемой анимации и переходов в проекте используется пакет DOTween. Он обладает следующими особенностями:
1. Производительность:
− оптимизированная Tween-система, работающая быстрее стандартного Unity Lerp;
− поддержка пулинга анимаций, что снижает нагрузку на Garbage Collector.
2. Простота использования:
− читаемый синтаксис (методы записываются цепочкой);
− автоматическая обработка прерывания анимаций.
3. Богатый функционал:
− поддержка сложных кривых;
− анимация UI-элементов;
− возможность последовательных и параллельных анимаций.
4. Работает на всех платформах.

Игрок, заходя в игру, попадает в главное меню (рис. 4). В главном меню отображается название игры и кнопки. В зависимости от нажатой кнопки игрок может перейти к изучению материала либо к экрану выбора задания.

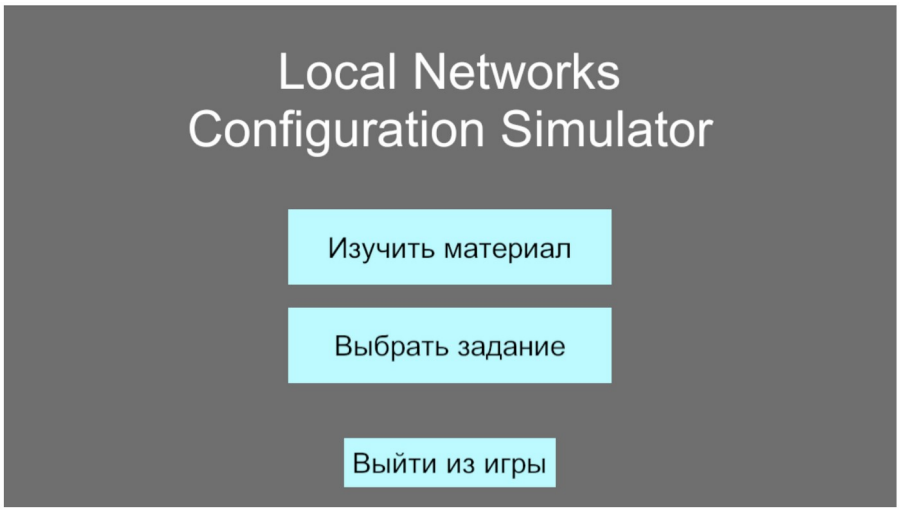

Рисунок 4 - Главное меню
DOI: https://doi.org/10.60797/IRJ.2025.162.140.4

Игрок во время игрока может находиться в следующих игровых состояниях (рис. 5):
1) главное меню;
2) изучение материала;
3) выбор задания (рис. 6);
4) игровой процесс;
5) конец задания.

Начальное игровое состояние — главное меню. Оно начинается с момента входа в игру. В нем игроку предоставлены возможности перейти к экрану изучения материала, перейти к экрану выбора задания. На главном экраны находятся несколько элементов интерфейса, такие как кнопка начала игры, кнопка выбора задания, кнопка выхода из игры, название игры. При взаимодействии с кнопками игрок переходит в следующее состояние.

В состоянии изучении материала на интерфейсе игрока появляется большая табличка с текстом и картинками. Игрок может листать содержимое, изучая теоретический материал, способы взаимодействия с устройствами и командами для их конфигурации. Игрок из данного состояния может вернуться в главное меню, либо перейти в состояние выбора задания.

В состоянии выбора задания у игрока появляется возможность перейти к конкретному заданию. Задания отображены таблицей. Кнопка с заданием перемещает пользователя в игровую зону данного задания. В кнопке с заданием находится текст, в котором написан номер задания.

Игровой процесс состоит из пространства, на котором расставлены игровые элементы. Среди них есть компьютеры и маршрутизаторы. Пользователь узнает о содержании задания из текстовой таблички, находящейся на стене. Затем пользователь настраивает устройства и выполняет задание, после чего игра переходит в следующее состояние.

В состоянии конца задания игрок видит перед собой текст об успешном выполнении задания. На экране также представлена кнопка для возвращения в исходное состояние – главное меню.

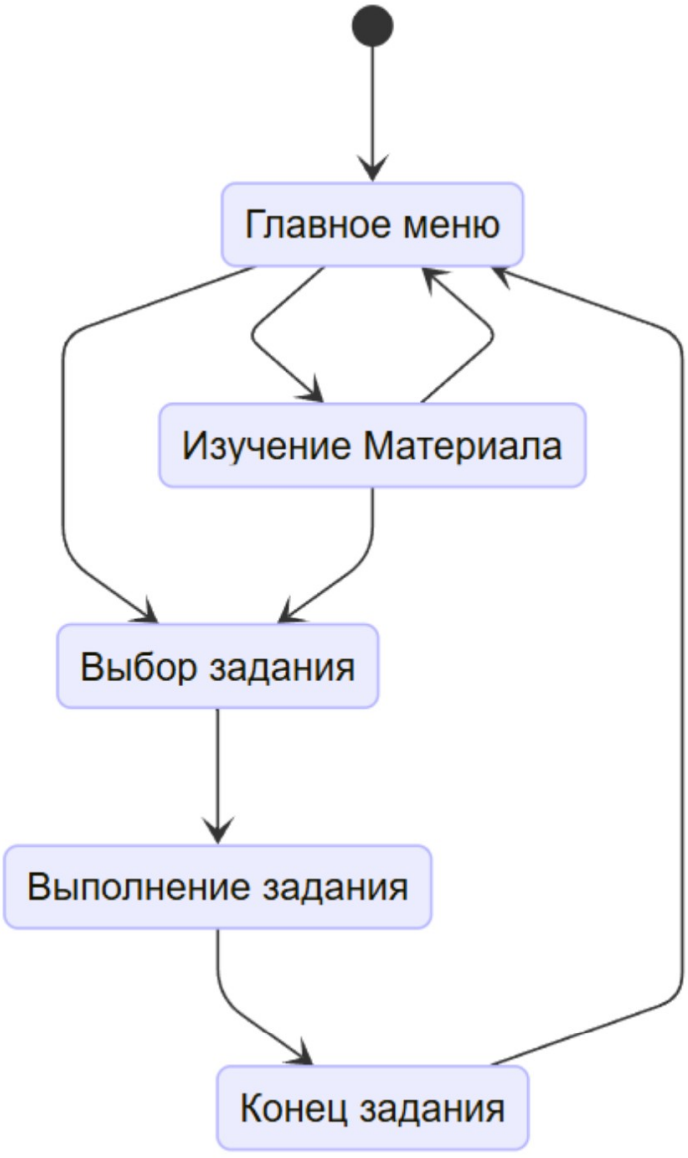

Рисунок 5 - Диаграмма состояний игрока

DOI: https://doi.org/10.60797/IRJ.2025.162.140.5

Рисунок 6 - Экран выбора задания
DOI: https://doi.org/10.60797/IRJ.2025.162.140.6

Благодаря визуализации игрового процесса с помощью игровых состояний можно упростить наглядное понимание взаимодействия пользователя с миром приложения. В дальнейшем систему можно расширить, добавив дополнительные состояния, такие как настройки игры, достижения или многопользовательский режим, что сделает игровой процесс еще более разнообразным и вовлекающим.

Программная реализация

Одним из основных составляющих видеоигр является игровой персонаж (рис. 7). Игровой персонаж представляет пользователя в пространстве игры, управляется пользователем с помощью системы ввода и взаимодействует с окружением через физический движок.

Для реализации игрового персонажа необходимо создать его 3D вид, а затем добавить на него необходимые компоненты. В данном проекте пользователь не видит персонажа, поэтому для него не нужны анимации и особый человеческий полигон. Исходя из этого можно упростить создание визуализации игрового персонажа [16], [17].

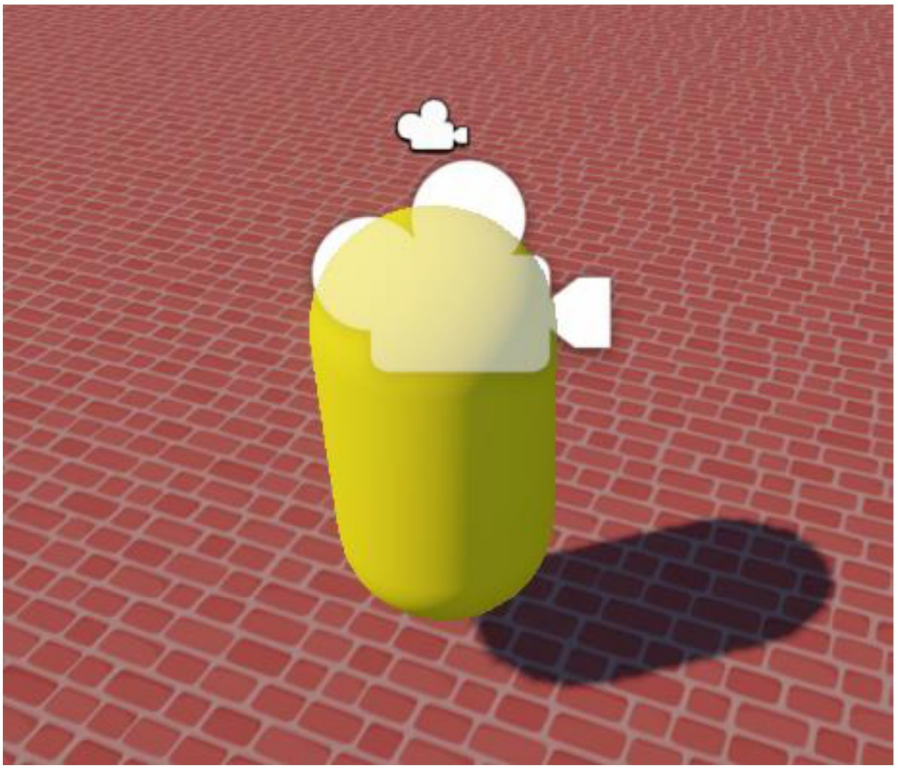

Рисунок 7 - Игровой персонаж
DOI: https://doi.org/10.60797/IRJ.2025.162.140.7

Для того, чтобы дать персонажу любой вид, необходимо подключить к объекту в Unity определенные компоненты. Выбор 3D модели происходит в компоненте Mesh Filter. Для персонажа данной видеоигры была выбрана фигура

Capsule. Таким образом, персонаж приобретает свою собственную 3D фигуру, но он пока не виден на сцене проекта. Для отображения используется другой компонент — Mesh Renderer [18], [19]. В нем есть поле, отвечающее за материал данной 3D фигуры. С помощью него был сделан желтый цвет игрового персонажа. Настройка этих двух компонентов представлена на рисунке 9.

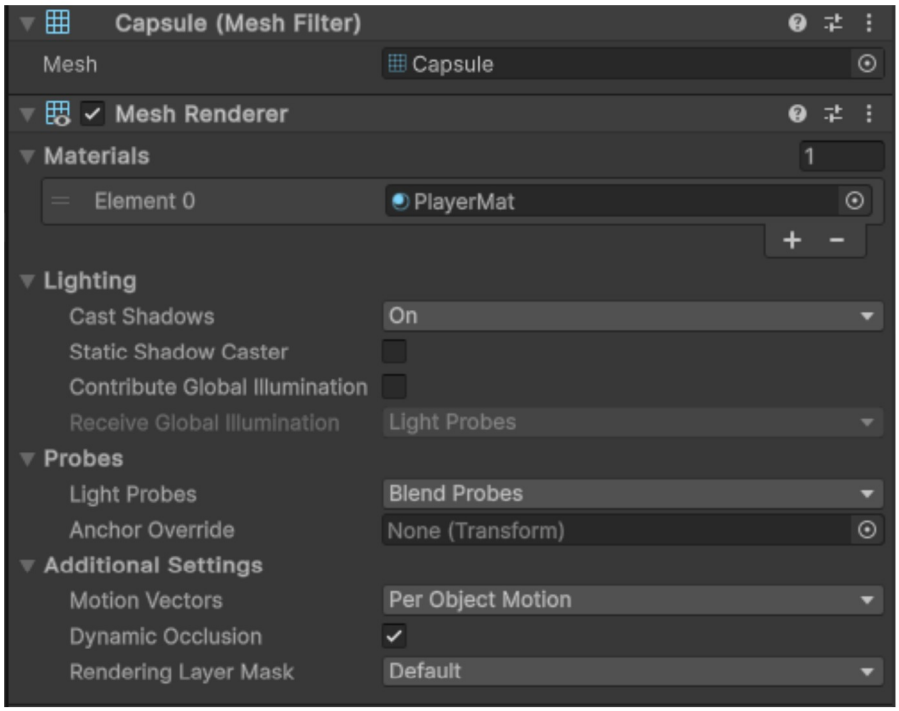

Рисунок 8 - Настройка компонентов Mesh Filter и Mesh Renderer
DOI: https://doi.org/10.60797/IRJ.2025.162.140.8

Таким образом, реализация 3D-модели игрового персонажа была выполнена с соблюдением принципа минимальной достаточности, что полностью соответствует задачам обучающего приложения. Данное решение демонстрирует эффективный подход к разработке функциональных элементов обучающих приложений, где приоритет отдается не визуальной составляющей, а стабильности работы и эффективности учебного процесса.

Добавления визуализации мало для полноценного создания персонажа игрока. Без внедрения физики в игрового персонажа он будет всего лишь статическим игровым объектом. Для того, чтобы игрок мог передвигаться и прыгать в пространстве, в Unity существуют специальные физические компоненты. В данном проекте использованы Rigidbody и CapsuleCollider. Rigidbody отвечает за физические свойства объекта, например гравитация, а CapsuleCollider отвечает за границу соприкосновения игрока с пространством. Перемещение персонажа реализовано в классе PlayerMovement (рис. 10).

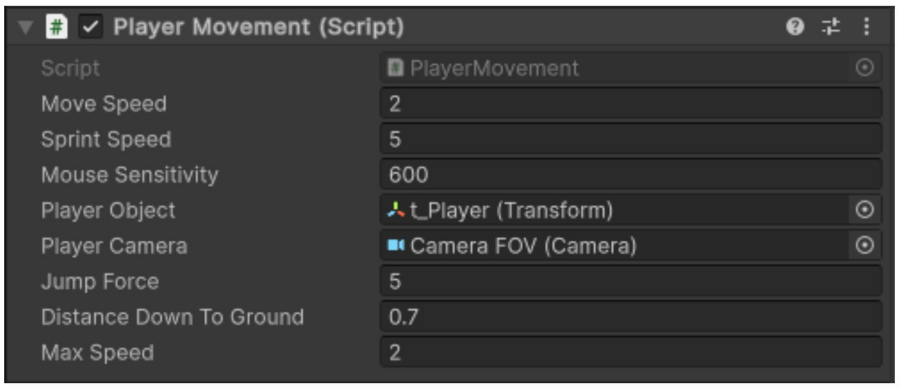

Рисунок 9 - Настройка компонента PlayerMovement
DOI: https://doi.org/10.60797/IRJ.2025.162.140.9

При запуске сцены с самого начала происходит назначение значений переменным. Метод Update вызывается каждый прошедший кадр во время работы приложения. Метод Update проверяет текущее состояние персонажа. Он может находиться в 5 разных состояниях.

Начальное состояние, в котором может находиться игрок — это состояние ходьбы (листинг 1). В этом состоянии игрока передвигается с минимальной скоростью, пока не столкнется с другим объектом. В методе Update вызывается MovePlayer(), в котором считывается нажатие пользователем клавиш WASD. Считывание клавиш происходит благодаря классу Input и методу GetAxis-Raw(string axisName). Этот метод отличается от GetAxis(string axisName) тем, что в нем отсутствует сглаживание движения, из-за чего после прекращения нажатия клавиши игрок немного продолжал своё движение. Чтобы считывать именно клавиши клавиатура необходимо указать в переменной axisName оси Horizontal и Vertical. Оптимизация перемещения персонажа в игровом пространстве достигается с помощью следующих факторов:

1) минимальные вычисления в Update();
2) отсутствие физических расчетов через Rigidbody для простых сцен;
3) подготовка к легкому расширению (бег, усталость и т.д.).

Листинг 1 — Реализация ходьбы персонажа в классе PlayerMovement

```
void MovePlayer()
{
float moveX = Input.GetAxisRaw("Horizontal");
float moveZ = Input.GetAxisRaw("Vertical");
Vector3 direction = new Vector3(moveX, 0, moveZ).normalized;
Vector3 moveDirection = playerObject.transform.TransformDirection(direction);
playerObject.transform.position += moveDirection * _currentSpeed * Time.deltaTime;
}
```

При нажатии клавиши Shift игрок переходил в следующее состояние под названием бег (листинг 2). При активации этого состояния вызывается метод SprintEnable(), который вызывает SetSprint(bool state) со значением true. Когда пользователь перестает нажимать клавишу, вызывается метод SprintDisable(), который вызывает SetSprint(bool state) со значением false. В зависимости от передаваемого bool значения, скорость игрока меняется либо на значение скорости при беге, либо на значение скорости при ходьбе.

Листинг 2 – Реализация бега персонажа в классе PlayerMovement

```
void SetSprint(bool state)
{
_currentSpeed = state ? sprintSpeed : moveSpeed;
}
public void SprintEnable() => SetSprint(true);
public void SprintDisable() => SetSprint(false);
```

Также игрок может находиться в состоянии прыжка. Данное состояние вызывается нажатием клавиши Space либо при ходьбе, либо при беге. Данное состояние реализуется в методе Jump() при помощи компонента Rigidbody и метода AddForce(). Этот метод запускает объект в указанном направлении с указанной силой. Чтобы игрок не имел возможности прыгать бесконечное количество раз, преодолевая верхнюю границу игрового пространства, метод Jump переключает флаг _isJumpingAllowed в значение false, чтобы нажатия на клавишу Space больше на считывались. Чтобы переключить этот флаг обратно в значение true, был реализован метод bool IsGrounded. Он использует физику объекта, запуская луч задаваемой длины от игрока по направлению вниз. Если луч задевает объект, то метод IsGrounded возвращает значение true, возвращая флаг в исходное состояние.

В видеоиграх при создании игрового объекта также добавляют ему возможность осматриваться вокруг себя. Для реализации этого функционала необходимо добавить камеру к объекту персонажа. Движение камеры реализовано в методе LookAround() (листинг 3), который вызывается из метода Update. Этот метод считывает перемещение курсора на экране с помощью метод Input.GetAxisRaw(string axisName). Если для перемещения персонажа необходимо было писать оси Horizontal и Vertical, то в данном случае необходимо указывать оси Mouse X и Mouse Y. Затем после считывания значение перемещения умножается на значение чувствительности мыши, задаваемой в инспекторе Unity, и на значение разности времени Time.deltaTime [20]. Это необходимо сделать, чтобы камера двигалась равномерно и без резкости. После считывания всех значений создается вектор поворота персонажа пользователя, а вместе с пользователем перемещается и камера, что позволяет игроку осматриваться в пространстве игрового мира.

Листинг 3 – Реализация вращения камеры в классе PlayerMovement

```
void LookAround()
{
float mouseX = Input.GetAxisRaw("Mouse X") * mouseSensitivity * Time.deltaTime;
float mouseY = Input.GetAxisRaw("Mouse Y") * mouseSensitivity * Time.deltaTime;
playerObject.Rotate(Vector3.up * mouseX);
xRotation -= mouseY;
xRotation = Mathf.Clamp(xRotation, -90f, 90f);
```

```
playerCamera.transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
}
```

Игровой персонаж, перемещаясь по полю может взаимодействовать с различными устройствами, чтобы выполнять задания. С целью интуитивного понимания игрового процесса, был реализован пользовательский интерфейс в классе PlayerInterfaceManager. Интерфейс реализован с помощью встроенного в Unity редактора UI Toolkit. Этот мощный инструмент схож с HTML и CSS. Пользовательский интерфейс содержит в себе небольшую табличку, в которой пишется текст. Табличка может адаптироваться под текст, чтобы отступ между текстом оставался постоянным, а значит случаев, когда текст выходит за рамки таблички возникать не будет. Данный класс содержит в себе публичные функции для взаимодействия с интерфейсом. С помощью функций можно выключить или включить интерфейс, задать текст в табличке, убрать часть текста или убрать его целиком. Переменные и функции класса PlayerInterfaceManager можно посмотреть на рисунке 11.

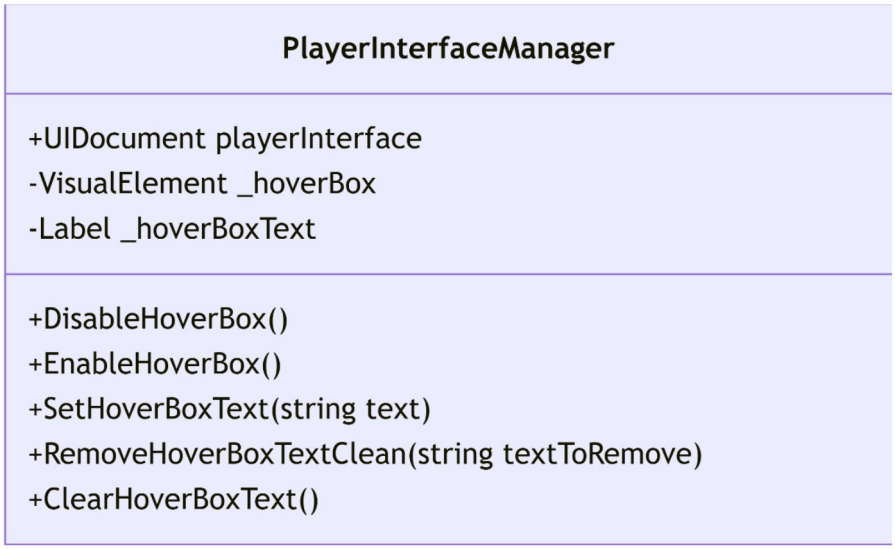

Рисунок 10 - Переменные и функции класса PlayerInterfaceManager
DOI: https://doi.org/10.60797/IRJ.2025.162.140.10

В инспекторе Unity переменной playerInterface задается добавленный в объекте компонент UIDocument, который содержит в себе созданный интерфейс. Присвоение переменным _hoverBox и _hoverBoxText своих элементов происходит в методе OnEnable(), который вызывается в самом начале работы сцены (листинг 4). В методе происходит поиск элементов в документе по названию. Чтобы изначально текст не содержал в себе никаких лишних элементов, вызывается метод для очистки текста. Инициализация пользовательского интерфейса происходит на раннем этапе загрузки сцены через стандартный механизм Unity. Это гарантирует готовность UI-элементов до их первого использования.

Листинг 4 – Присвоение значений визуальным элементам интерфейса

```
void OnEnable()
{
_hoverBox = playerInterface.rootVisualElement.Q("Hover-Box");
_hoverBoxText = _hoverBox.Q<Label>("Hover-Box-Text");
ClearHoverBoxText();
}
```

Для добавления текста в табличку другие классы обращаются к методу SetHoverBoxText(string text), передавая значение, которое необходимо вставить в нее. Перед добавлением переданного текста метод проверяет, не дублируется ли он в уже существующем тексте. В случае нахождения дубликатов метод завершает работу, не добавив переданной переменной text. Полный код данного метода можно посмотреть в листинге 5.

Листинг 5 – Метод SetHoverBoxText(string text)

```
public void SetHoverBoxText(string text){
string currentText;
currentText = _hoverBoxText.text;
if (currentText.Contains(text)) return;
```

```
_hoverBoxText.text = [__rj__content__placeholder__]quot;{currentText}\n{text}";
}
```

Помимо добавления текста, другие классы могут и убрать текст, используя метод RemoveHoverBoxTextClean(string textToRemove). Этот метод использует регулярные выражения для точного удаления. Благодаря регулярным выражениям в тексте не остается лишних переносов строки и символов. Реализация метода удаления текста показана в листинге 6.

Листинг 6 – Метод RemoveHoverBoxTextClean(string textToRemove)

```
public void RemoveHoverBoxTextClean(string textToRemove)
{
_hoverBoxText.text = System.Text.RegularExpressions.Regex.Replace(_hoverBoxText.text, $@"(\n)?
{System.Text.RegularExpressions.Regex.Escape(textToRemove)}","");
_hoverBoxText.text = _hoverBoxText.text.Trim('\n');
}
```

Реализованные классы управления игровым персонажем и пользовательским интерфейсом управляются главным классом (рис. 12). Класс PlayerController представляет собой главный мозг и переключатель состояний персонажа. В нем содержится большое количество переменных, слушателей и методов (рис. 13).

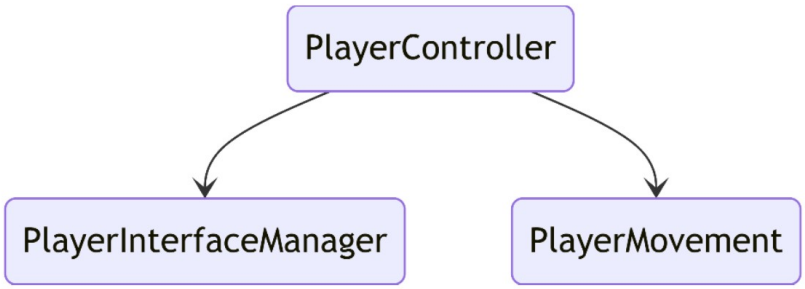

Рисунок 11 - Зависимость классов интерфейса и движения от PlayerController

DOI: https://doi.org/10.60797/IRJ.2025.162.140.11

Рисунок 12 - Переменные и методы класса PlayerController
DOI: https://doi.org/10.60797/IRJ.2025.162.140.12

Класс PlayerController отвечает за взаимодействие пользователя с игровым пространством с помощью переключения состояний персонажа. В методе Update вызывается stateListener(). Этот метод отвечает за переключение видимости текстов интерфейса, за переключение видимости и блокировки позиции курсора, за видимость интерфейсов маршрутизатора и компьютера.

Начальным состоянием является AbleToMove. Исходя из названия можно понять, что это состояние, в котором игрок может свободно перемещаться, не находясь в процессе взаимодействия с окружающими устройствами. Находясь в этом состоянии в методе stateListener() вызываются следующие функции:

1) разрешение движения персонажа;
2) выключение интерфейса маршрутизатора;
3) выключение интерфейса компьютера;
4) блокировка позиции курсора;

5) установка спрятанного режима видимости курсора;

6) удаление из интерфейса пользователя строк, которые могли остаться после взаимодействия с устройствами.

В состоянии AbleToMove пользователь может перейти в 3 различных состояния. Первым из них является подключение и отключение устройств к маршрутизатору — AssigningRouter (рис. 14). При активации этого метода маршрутизатор, с которым взаимодействовал игрок загорается зеленым цветом, а подключенные устройства загораются синим. В этом режиме ставится блокировка на переход в режим конфигурации устройств. В нем разрешено только подключение и отключение устройств к выбранному маршрутизатору.

Рисунок 13 - Демонстрация пользовательского интерфейса и игрового пространства в режиме подключения устройств к маршрутизатору

DOI: https://doi.org/10.60797/IRJ.2025.162.140.13

На рисунке видно, что маршрутизатор, с которым взаимодействовал пользователь загорелся зеленым цветом, а компьютер, находящийся слева от него, подсветился синим. Это значит, что к текущему маршрутизатору компьютер слева подключен, а компьютер справа — не подключен. Благодаря подсветке пользователь наглядно будет видеть текущее состояние игрового пространства, а значит меньше шанс, что он не запутается в том, какое устройство подключено, а какое — нет.

Для считывания нажатий на компьютерную мышь используется метод checkClicks, вызываемый в Update. Этот метод создает луч, исходящий от игрока по направлению центра экрана. При столкновении данного луча с объектом сначала проверяется флаг _routerAssigningModeComputerClicksOnly. Этот флаг принимает значение true в рассматриваемом состоянии AssigningRouter. С помощью данного флага, при нажатии на левую кнопку мыши, проверяются только объекты, тэг которых либо ComputerAccessZone, либо RouterAssignZone. Все остальные объект пропускаются. Это было сделано для того, чтобы назначать текущему маршрутизатору компьютеры и другие маршрутизаторы. Из состояния AssigningRouter можно перейти только в состояние AbleToMove при помощи нажатия на клавишу Escape.

Пользователь для настройки устройств в локальной сети взаимодействует с ними. Одно из устройств, с которым взаимодействует игрок – это компьютер. Чтобы с начать его настройку необходимо нажать левую кнопку мыши. Сам игрок перейдет в состояние настройки компьютера под названием UsingComputer. В момент перехода в данное состояние метод stateListener() выполняет следующие команды:

1) остановка движения игрока;

2) перевод курсора в видимый режим

3) отключение блокировки позиции курсора;

4) отключение режима задания устройств маршрутизатору

5) установка таблички подсказок в видимый режим;

6) удаление из таблички подсказок лишних текстов;

7) добавление в табличку подсказок инструкцию выхода из данного состояния.

Как и было сказано ранее в рассматриваемом состоянии пользовал находится в режиме настройки компьютера. В нем игрок не может перемещаться, прыгать и взаимодействовать с другими объектами и устройствами данного игрового пространства. Пользователь может только взаимодействовать с интерфейсом компьютера.

Похожим состоянием на UsingComputer является UsingRouter. В нем метод stateListener() выполняет те же функции, но вместо интерфейса компьютера взаимодействует с интерфейсом маршрутизатора. Пользователь из данных двух состояний может перейти обратно в состояние AbleToMove нажав кнопку, указанную в табличке подсказок в левом нижнем угле экрана. Таким образом, система состояний UsingComputer и UsingRouter обеспечивает последовательное взаимодействие пользователя с сетевым оборудованием, исключая возможность случайных действий вне текущего интерфейса настройки.

Одним из ключевых устройств данного проекта, с которым может взаимодействовать игрок — компьютер. Основной класс, который отвечает за установленные параметры для сети, за интерфейс и остальное, что связано

именно с конкретным компьютером, называется ComputerController. Данный класс добавляется в качестве компонента к объекту компьютера на сцене. Пример настройки данного компонента представлен на рисунке 15.

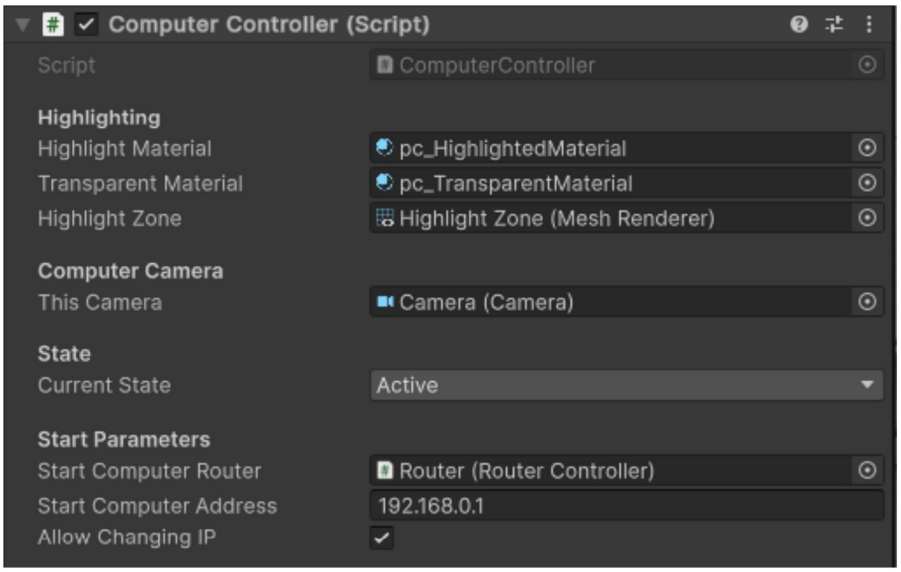

Рисунок 14 - Пример настройки компонента ComputerController
DOI: https://doi.org/10.60797/IRJ.2025.162.140.14

На данном рисунке видно, что параметры распределены по секциям. В первой секции находятся параметры, отвечающие за подсветку компьютера. Подсветка компьютера необходима в режиме подключения и отключения устройств к маршрутизатору. Изначально в методе Start() подсвечивание компьютера отключается. За включение и отключение выделения отвечают методы HighlightComputer() и UnhighlightComputer() соответственно.

Следующая секция отвечает за камеру, которая включается при переходе в состояния UsingComputer. Чтобы включить камеру, пользователь должен нажать на компьютер. После нажатия в ComputerController вызывается метод, который считывает изначальную позицию камеры игрока, перемещает камеру компьютера в данную позицию, а затем благодаря пакету DOTween плавно пе-ремещает активную камеру к компьютеру, создавая эффект приближения.

Секции State и StartParameters отвечают за установку начальных значений. Переменная currentState отвечает за состояния компьютера и может быть равна значению «включен» либо «выключен». За включение и выключение компьютера отвечает кнопка, находящаяся на интерфейсе компьютера в правой нижней части его экрана.

У каждого компьютера есть своя зона, при нажатии на которую пользователь подключается к его интерфейсу. Данная зона отмечена тэгом ComputerAccessZone. После нажатия на данную зону игрок переходит в состояние UsingComputer, камера перемещается ближе к компьютеру, а когда действие закончится, включается интерфейс. Пример интерфейса компьютера представлен на рисунке 16.

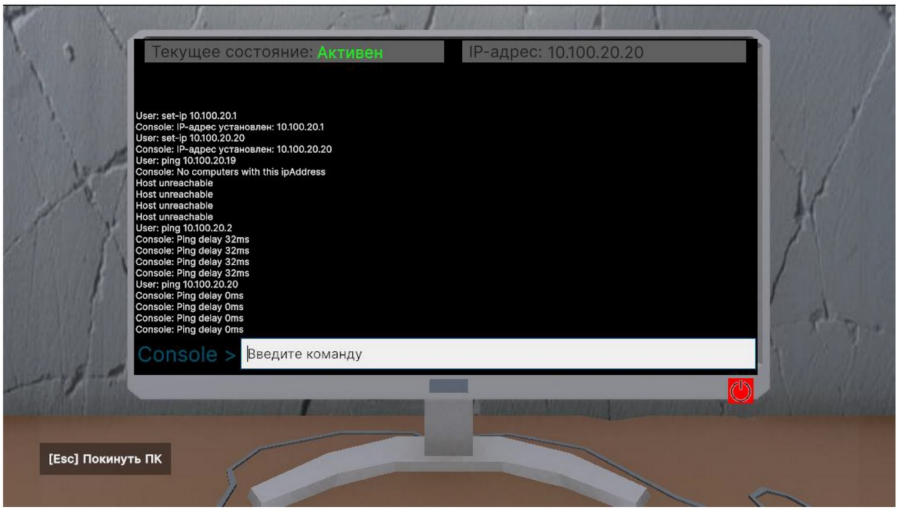

Рисунок 15 - Пример интерфейса компьютера
DOI: https://doi.org/10.60797/IRJ.2025.162.140.15

На данном интерфейсе отображаются следующие элементы:

1) кнопка включения или выключения компьютера: первый элемент, с которым может взаимодействовать пользователь, изменяющий текущее состояние на обратное, то есть с состояния «включен» компьютер переходит в «выключен» и наоборот;

2) текущее состояния, указывающее включен или выключен в данный момент компьютер;

3) IP-адрес: данное поле указывает установленный в текущий момент адрес в сети данного компьютера;

4) консоль вывода: в этой консоли выводится ответ обработчика команд;

5) консоль ввода: второй элемент интерфейса компьютера, с которым может взаимодействовать пользователь (в данном поле игрок вводит команды).

Класс ComputerController также содержит в себе методы, которые необходимы для выполнения условий задания. Одним из главных факторов выполнения задания — команда ping. В данном классе за это действие отвечает метод TryPing(string ip). Данный метод имеет тип значения bool. Если компьютер не подключен ни к одному маршрутизатору или компьютер не имеет своего собственного IP-адреса в сети, то метод возвращает значение false. При подключенном маршрутизаторе и наличии своего адреса, метод начинает проверку наличия переданного IP-адреса в подключенном роутере. После проверки возвращается значение true или false в зависимости от полученного значения проверки доступности.

За интерфейс компьютера отвечает класс PCInterfaceManager. Данный класс содержит в себе большое количество методов и проверок, которые необходимы для обеспечения правильности выполнения работы команд, действий и других взаимодействий с ним. Запуск интерфейса происходит из метода SetupInterface(), находящемся в классе ComputerController. Этот метод вызывает SetAndActivateInterface(ComputerController cc), передавая себя в качестве аргумента. Суть данного метода заключается в следующем:

1) переменным, отвечающим за визуальные элементы, задаются соответствующие элементы интерфейса;

2) переменной, отвечающей за хранение текущего используемого компьютера, задается текущий компьютер;

3) текущей консоли вывода задается хранящаяся в классе ComputerController история консоли компьютера;

4) кнопке включения и выключения задается функция, которая выполняет соответствующую функцию;

5) на событие нажатия клавиши Enter при использовании консоли ввода задается функция, отвечающая за обработку введенных пользователем команд.

Главным обработчиком команд, вводимых в компьютерный интерфейс, является класс PCInterfaceManager. Всего есть две команды, которые можно ввести в консоль:

1) set-ip;

2) ping.

После подтверждения команды, она разделяется на части, используя в качестве разделителя пробел. В текущую консоль вывода выводится запись User, а рядом прописывается введенная команда. Дальнейшая проверка не начинается, если введенная строка пустая.

Первая команда отвечает за установку IP-адреса используемому компьютеру. Чтобы задать этот адрес, необходимо написать данную команду, а рядом прописать желаемый адрес для компьютера. Перед установкой проверяется разрешение изменения адреса компьютера. На компьютере можно включить флаг, который заблокирует доступ изменению его адреса. Это необходимо для заданий, где адрес либо задан изначально и не должен изменяться, либо где адрес задается другим способом. Если проверка разрешения вернула успешный результат, проверяется было ли введено две части. Если проверка не прошла, в консоль выводится сообщение, которое информирует о неправильном вводе, показывая формат корректного ввода команды. При успешном прохождения каждой проверки вызывается метод SetIpAddress(string ip) в PCInterfaceManager. Адрес обязан быть в корректном формате виде *.*.*.*, где звезда означает любое число от 0 до 255 включительно. Если всё введено верно, то текущему компьютеру задается адрес, соответствующее сообщение выводится в консоль.

Вторая команда проверяет доступность передаваемого адреса компьютера. Команда ping проверяется аналогично команде set-ip, но в ней нет проверки разрешения замены IP-адреса компьютера в виду отсутствия необходимости в этом. При прохождении проверок корректности ввода команды вызывается метод Ping(string ip). Код данного метода представлен в листинге 7.

Листинг 7 – Работа метода проверки доступности компьютера

```
void Ping(string ipAddress)
{
bool pingResult = _activePC.TryPing(ipAddress);
StartCoroutine(PingDelayed(pingResult, ipAddress));
if (!pingResult)
{
_currentConsoleOutput.Add("Console: No computers with this ipAd-dress");
}
}
IEnumerator PingDelayed(bool success, string ip)
{
string delay = ip == _activePC.GetIpAddress() ? "0ms" : "32ms";
string outputString = success ? [__rj__content__placeholder__]quot;Console: Ping delay {delay}" : "Host
unreachable";
for (int i = 0; i < 4; i++)
{
yield return new WaitForSeconds(0.32f);
_currentConsoleOutput.Add(outputString);
SetConsoleOutput(_currentConsoleOutput);
```

```
}
}
```

Данный метод изначально проверяет доступность компьютера, сохраняя конечный результат. В зависимости от результата выводится либо задержка соединения, либо ответ о недоступности адресата. Также есть зависимость от IP-адреса. Если адрес адресата совпадает с адресом отправителя, то задержка равно нулю, иначе 32 мс.

Следующим ключевым устройством в локальной сети данной видеоигры является маршрутизатор. Маршрутизаторы используются для подключения компьютеров к одной сети. Также маршрутизаторы могут объединять несколько подсетей, образуя одну большую.

В данной игре маршрутизатор реализован как сложная система, состоящая из трёх ключевых взаимосвязанных классов. Два из этих классов добавляются в объект маршрутизатора в качестве компонентов. Основным классом, отвечающий за хранение ключевых параметров для маршрутизатора является класс RouterController (рис. 17).

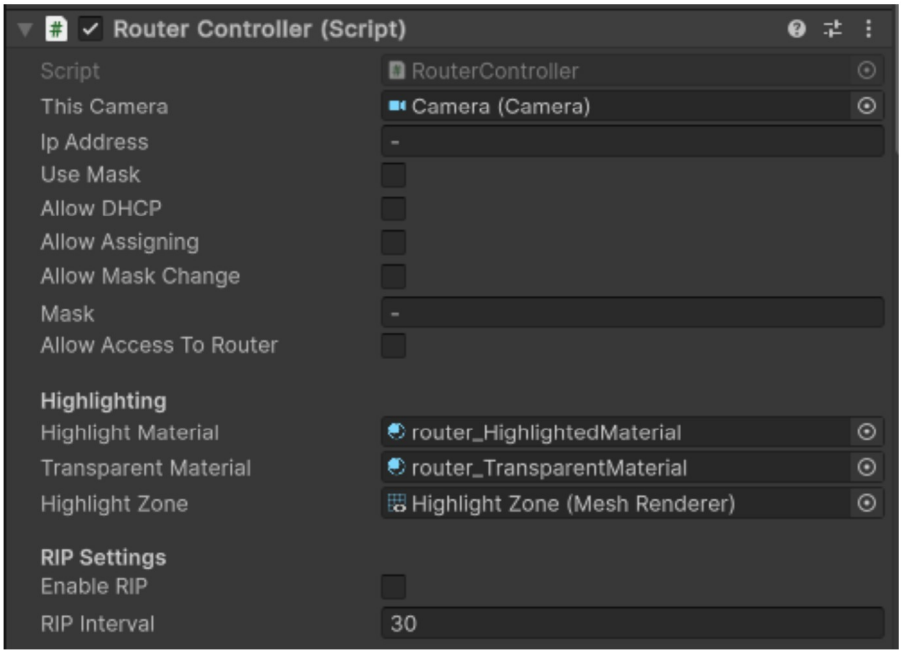

Рисунок 16 - Пример настройки класса RouterController
DOI: https://doi.org/10.60797/IRJ.2025.162.140.16

Так же, как и в главном управляющем классе компьютера, тут реализована подсветка устройства при взаимодействии с ним. Переменные, находящиеся в сегменте Highlighting, отвечают за подсветку маршрутизатора. Это необходимо для того, чтобы наглядно показать пользователю принадлежность данного устройства другой подсети.

Для маршрутизатора реализованы следующие настройки:

1) IP-адрес: данная переменная отвечает за установленный адрес данного маршрутизатора в подсети;

2) Mask: эта переменная устанавливает маску, с помощью которой проверяется принадлежность введенного IP-адреса к подсети;

3) Enable RIP: флаг включение RIP отвечает за автоматическую настройку и обновление таблицы маршрутизаций;

4) RIP Interval: значение данной переменной устанавливает временной промежуток, прохождение которого вызывает метод для обновления таблицы маршрутизаций.

Для того чтобы пользователь взаимодействовал с маршрутизатором помимо режима, когда он может подключать или отключать от него устройства, был реализован интерфейс, с которым игрок может взаимодействовать. Управлением этого интерфейса занимается класс RouterInterfaceManager. Его задача заключается в обработке команд пользователя и выводе информации на экран. Метод, включающий данный интерфейс, вызывается из RouterController.

Заключение

Разработанная интерактивная 3D-платформа демонстрирует высокий потенциал в области обучения построению и конфигурации локальных сетей, объединяя в себе современные игровые механики, визуализацию и практико-ориентированный подход. Использование геймификации и трехмерной среды позволяет преодолеть ограниченность традиционных методик, делая процесс обучения наглядным, увлекательным и мотивирующим. Платформа обеспечивает не только постепенное освоение теоретического материала, но и формирование устойчивых практических навыков, необходимых для будущей профессиональной деятельности.

Практическая значимость решения заключается в его универсальности: продукт может применяться как в образовательных учреждениях при подготовке студентов технических специальностей, так и в системе корпоративного обучения, а также в формате самостоятельного онлайн-образования. Модульная архитектура и

гибкость реализации открывают возможности для дальнейшего расширения функционала, интеграции новых сценариев и адаптации под конкретные цели.

Перспективы развития проекта связаны с внедрением многопользовательского режима, который позволит реализовать сетевые взаимодействия между несколькими участниками в реальном времени, расширением набора учебных заданий и уровней сложности, а также использованием технологий виртуальной и дополненной реальности для усиления эффекта погружения. Существенным направлением совершенствования может стать интеграция интеллектуальных подсказок и автоматизированной оценки действий пользователей на основе методов искусственного интеллекта, что повысит персонализацию и эффективность образовательного процесса. В долгосрочной перспективе подобные системы способны трансформировать практику преподавания сетевых технологий, обеспечив баланс между теорией, практикой и игровыми элементами, что будет способствовать подготовке специалистов нового поколения, ориентированных на работу в цифровой среде.

Конфликт интересов

Не указан.

Рецензия

Пикулев А.Н., Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань Российская Федерация
DOI: https://doi.org/10.60797/IRJ.2025.162.140.17

Conflict of Interest

None declared.

Review

Pikulev A.N., Kazan National Research Technical University named after A.N. Tupolev – KAI, Kazan Russian Federation
DOI: https://doi.org/10.60797/IRJ.2025.162.140.17

Список литературы / References

1. Джанхуватова З.С. Повышение эффективности обучения в рамках проблемы «клипового мышления» и познавательно-творческой деятельности студентов / З.С. Джанхуватова, Г.Н. Арстанова // Известия Балтийской государственной академии рыбопромыслового флота: психолого-педагогические науки. — 2018. — № 4 (46). — С. 85–89.

2. Ларченко Ю.Г. Игровые технологии в дополнительном профессиональном образовании / Ю.Г. Ларченко // Современные исследования социальных проблем. — 2015. — № 1 (21). — С. 195–200.

3. Армеев Д.В. Виртуальный доступ к обучающим средам / Д.В. Армеев, В.И. Гужов // Открытое и дистанционное образование. — 2008. — № 4 (32). — С. 34–36.

4. Свиридов А.П. Методика обучения и интерпретации нейронной сети для компьютерного контроля знаний / А.П. Свиридов, А.М. Титов // Нейрокомпьютеры: разработка, применение. — 2009. — № 8. — С. 63–68.

5. Бойко А.В. Возможности локальной компьютерной сети в организации индивидуализированного обучения / А.В. Бойко, А.В. Боронин // Педагогический университетский вестник Алтая. — 2001. — № 1. — С. 124–127.

6. Гамула Д.С. Цифровой двойник лабораторного стенда по изучению автоматических регуляторов и типовых законов регулирования / Д.С. Гамула, М.Ю. Перухин, Р.Ф. Гибадуллин // Международный научно-исследовательский журнал. — 2024. — № 10 (148). — DOI: 10.60797/IRJ.2024.148.151.

7. Ренев Р.О. Виртуальные локальные сети и маршрутизация между ними / Р.О. Ренев // Спецтехника и связь. — 2012. — № 3. — С. 41–44.

8. Яремчук С. Файловые системы пространства пользователя / С. Яремчук // Системный администратор. — 2005. — № 6 (31). — С. 40–43.

9. Боровиков А.В. Компьютеризация образования / А.В. Боровиков, А.Э. Назиров // Современное образование: содержание, технологии, качество. — 2009. — Т. 1. — С. 77–79.

10. Лазарева О.Ю. Обзор современных игровых движков / О.Ю. Лазарева, А.В. Санина // Вестник Воронежского института высоких технологий. — 2018. — № 4 (27). — С. 29–32.

11. Кушнир Н.В. Базовое программное обеспечение разработки компьютерных видеоигр. Игровые движки / Н.В. Кушнир, К.А. Будников, С.Э. Ирхин [и др.] // Электронный сетевой политематический журнал «Научные труды КубГТУ». — 2020. — № 2. — С. 11–24.

12. Данилюк М.Д. Разработка видеоигр: проблемы современных исследований / М.Д. Данилюк, Ю.Ф. Шпаковский // Труды БГТУ. Серия 4: Принт- и медиатехнологии. — 2017. — № 1 (195). — С. 118–122.

13. Саносян Р.А. Комплексный подход к разработке видеоигр: от концепции до реализации / Р.А. Саносян, А.Л. Титова, А.П. Куцый // Молодая наука Сибири. — 2024. — № 3 (25). — С. 79–82.

14. Андреева Е.Г. Обзор методов проектирования пользовательского интерфейса / Е.Г. Андреева // Молодежный научно-технический вестник. — 2016. — № 10. — С. 28.

15. Иванов А.В. Компьютерная графика и процесс проектирования / А.В. Иванов, Л.В. Ремонтова, Е.Ю. Юдина [et al.] // Современные информационные технологии. — 2004. — № S2. — С. 260–261.

16. Шавтикова Л.М. Применение компьютерного моделирования при создании персонажей / Л.М. Шавтикова, А.В. Левченко // Тенденции развития науки и образования. — 2021. — № 80-3. — С. 6–8. — DOI: 10.18411/trnio-12-2021-108.

17. Корнейчук А.В. Разработка и анимация трехмерной модели персонажа для компьютерной игры / А.В. Корнейчук, Е.А. Глибко, М.А. Максимова // Теория и практика дизайна. — 2016. — № 10. — С. 70–77.

18. Батчаева З.Б. 3D моделирование / З.Б. Батчаева, М.Х. Чомаева // Тенденции развития науки и образования. — 2022. — № 90-3. — С. 78–80. — DOI: 10.18411/trnio-10-2022-122.

19. Шорохов В.Д. 3D графика в проектировании / В.Д. Шорохов, Д.В. Гулякин // Тенденции развития науки и образования. — 2023. — № 102-6. — С. 175–179. — DOI: 10.18411/trnio-10-2023-358.

20. Русанова Я.М. Некоторые проблемы визуализации трехмерных сцен в реальном времени / Я.М. Русанова, М.И. Чердынцева // Прикладная информатика. — 2008. — № 6 (18). — С. 78–86.

Список литературы на английском языке / References in English

1. Dzhanhuvatova Z.S. Povyshenie jeffektivnosti obuchenija v ramkah problemy «klipovogo myshlenija» i poznavatel'no-tvorcheskoj dejatel'nosti studentov [Improving the effectiveness of teaching in the context of ‘clip thinking’ and students' cognitive and creative activities] / Z.S. Dzhanhuvatova, G.N. Arstanova // Izvestija Baltijskoj gosudarstvennoj akademii rybopromyslovogo flota: psihologo-pedagogicheskie nauki [Proceedings of the Baltic State Academy of Fishing Fleet: Psychological and Pedagogical Sciences]. — 2018. — № 4 (46). — P. 85–89. [in Russian]

2. Larchenko Ju.G. Igrovye tehnologii v dopolnitel'nom professional'nom obrazovanii [Gaming technologies in continuing professional education] / Ju.G. Larchenko // Sovremennye issledovanija social'nyh problem [Contemporary Research on Social Issues]. — 2015. — № 1 (21). — P. 195–200. [in Russian]

3. Armeev D.V. Virtual'nyj dostup k obuchajushhim sredam [Virtual access to learning environments] / D.V. Armeev, V.I. Guzhov // Otkrytoe i distancionnoe obrazovanie [Open and distance education]. — 2008. — № 4 (32). — P. 34–36. [in Russian]

4. Sviridov A.P. Metodika obuchenija i interpretacii nejronnoj seti dlja komp'juternogo kontrolja znanij [Methodology for training and interpreting neural networks for computer-based knowledge assessment] / A.P. Sviridov, A.M. Titov // Nejrokomp'jutery: razrabotka, primenenie [Neural computers: development, application]. — 2009. — № 8. — P. 63–68. [in Russian]

5. Bojko A.V. Vozmozhnosti lokal'noj komp'juternoj seti v organizacii individualizirovannogo obuchenija [The potential of local computer networks in organising individualised learning] / A.V. Bojko, A.V. Boronin // Pedagogicheskij universitetskij vestnik Altaja [Pedagogical University Bulletin of Altai]. — 2001. — № 1. — P. 124–127. [in Russian]

6. Gamula D.S. Cifrovoj dvojnik laboratornogo stenda po izucheniju avtomaticheskih reguljatorov i tipovyh zakonov regulirovanija [Digital twin of a laboratory stand for studying automatic controllers and typical control laws] / D.S. Gamula, M.Ju. Peruhin, R.F. Gibadullin // Mezhdunarodnyj nauchno-issledovatel'skij zhurnal [International Research Journal]. — 2024. — № 10 (148). — DOI: 10.60797/IRJ.2024.148.151. [in Russian]

7. Renev R.O. Virtual'nye lokal'nye seti i marshrutizacija mezhdu nimi [Virtual local area networks and routing between them] / R.O. Renev // Spectehnika i svjaz' [Specialised equipment and communications]. — 2012. — № 3. — P. 41–44. [in Russian]

8. Jaremchuk S. Fajlovye sistemy prostranstva pol'zovatelja [User space file systems] / S. Jaremchuk // Sistemnyj administrator [System Administrator]. — 2005. — № 6 (31). — P. 40–43. [in Russian]

9. Borovikov A.V. Komp'juterizacija obrazovanija [Computerisation of education] / A.V. Borovikov, A.Je. Nazirov // Sovremennoe obrazovanie: soderzhanie, tehnologii, kachestvo [Modern education: content, technologies, quality]. — 2009. — Vol. 1. — P. 77–79. [in Russian]

10. Lazareva O.Ju. Obzor sovremennyh igrovyh dvizhkov [Overview of modern game engines] / O.Ju. Lazareva, A.V. Sanina // Vestnik Voronezhskogo instituta vysokih tehnologij [Bulletin of the Voronezh Institute of High Technologies]. — 2018. — № 4 (27). — P. 29–32. [in Russian]

11. Kushnir N.V. Bazovoe programmnoe obespechenie razrabotki komp'juternyh videoigr. Igrovye dvizhki [Basic software for developing computer video games. Game engines] / N.V. Kushnir, K.A. Budnikov, S.Je. Irhin [et al.] // Jelektronnyj setevoj politematicheskij zhurnal «Nauchnye trudy KubGTU» [Electronic multi-thematic online journal ‘Scientific Works of Kuban State Technical University’]. — 2020. — № 2. — P. 11–24. [in Russian]

12. Daniljuk M.D. Razrabotka videoigr: problemy sovremennyh issledovanij [Video game development: problems in contemporary research] / M.D. Daniljuk, Ju.F. Shpakovskij // Trudy BGTU. Serija 4: Print- i mediatehnologii [Proceedings of BSTU. Series 4: Print and Media Technologies]. — 2017. — № 1 (195). — P. 118–122. [in Russian]

13. Sanosjan R.A. Kompleksnyj podhod k razrabotke videoigr: ot koncepcii do realizacii [A comprehensive approach to video game development: from concept to implementation] / R.A. Sanosjan, A.L. Titova, A.P. Kucyj // Molodaja nauka Sibiri [Young Science of Siberia]. — 2024. — № 3 (25). — P. 79–82. [in Russian]

14. Andreeva E.G. Obzor metodov proektirovanija pol'zovatel'skogo interfejsa [Overview of user interface design methods] / E.G. Andreeva // Molodezhnyj nauchno-tehnicheskij vestnik [Youth Scientific and Technical Bulletin]. — 2016. — № 10. — P. 28. [in Russian]

15. Ivanov A.V. Komp'juternaja grafika i process proektirovanija [Computer graphics and the design process] / A.V. Ivanov, L.V. Remontova, E.Ju. Judina [et al.] // Sovremennye informacionnye tehnologii [Modern Information Technologies]. — 2004. — № S2. — P. 260–261. [in Russian]

16. Shavtikova L.M. Primenenie komp'juternogo modelirovanija pri sozdanii personazhej [The use of computer modelling in character creation] / L.M. Shavtikova, A.V. Levchenko // Tendencii razvitija nauki i obrazovanija [Tendencies in the development of science and education]. — 2021. — № 80-3. — P. 6–8. — DOI: 10.18411/trnio-12-2021-108. [in Russian]

17. Kornejchuk A.V. Razrabotka i animacija trehmernoj modeli personazha dlja komp'juternoj igry [Development and animation of a three-dimensional character model for a computer game] / A.V. Kornejchuk, E.A. Glibko, M.A. Maksimova // Teorija i praktika dizajna [Theory and Practice of Design]. — 2016. — № 10. — P. 70–77. [in Russian]

18. Batchaeva Z.B. 3D modelirovanie [3D modelling] / Z.B. Batchaeva, M.H. Chomaeva // Tendencii razvitija nauki i obrazovanija [Tendencies in the development of science and education]. — 2022. — № 90-3. — P. 78–80. — DOI: 10.18411/trnio-10-2022-122. [in Russian]

19. Shorohov V.D. 3D grafika v proektirovanii [3D graphics in design] / V.D. Shorohov, D.V. Guljakin // Tendencii razvitija nauki i obrazovanija [Tendencies in the development of science and education]. — 2023. — № 102-6. — P. 175–179. — DOI: 10.18411/trnio-10-2023-358. [in Russian]

20. Rusanova Ja.M. Nekotorye problemy vizualizacii trehmernyh scen v real'nom vremeni [Some problems of real-time visualization of three-dimensional scenes] / Ja.M. Rusanova, M.I. Cherdynceva // Prikladnaja informatika [Applied Informatics]. — 2008. — № 6 (18). — P. 78–86. [in Russian]