

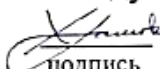
Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Высшей математики, информатики, экономики и естествознания
кафедра

УТВЕРЖДАЮ



Заведующий кафедрой

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

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии
код-наименование направления

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИГРЫ В СТИЛЕ «ГЕРОЙ-
ПРОГРАММИСТ» ДЛЯ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ

Руководитель	 14.06.24 подпись, дата	доцент, канд. пед. наук должность, ученая степень	<u>С.С. Ахтамова</u> инициалы, фамилия
Выпускник	<u>НХВ-14.06.2024</u> подпись, дата		<u>Д.С. Хабибулин</u> инициалы, фамилия
Нормоконтролер	 14.06.2024 подпись, дата		<u>А.В. Фирер</u> инициалы, фамилия

Лесосибирск 2024

РЕФЕРАТ

Выпускная квалификационная работа по теме «Проектирование и разработка игры в стиле «Герой-программист» для обучения программированию» содержит 52 страницы текстового документа, 40 использованных источника, 8 иллюстраций.

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОГРАММИРОВАНИЕ, ПРОЕКТИРОВАНИЕ, БРАУЗЕРНАЯ ИГРА.

Актуальность заключается в привлечении и мотивации учащихся к обучению посредством геймификации навыкам программирования.

Цель исследования: разработать образовательную игру в стиле «Герой-программист» для обучения основам программирования.

Объект исследования: обучение программированию с использованием образовательных игр.

Предмет исследования: методы и технологии разработки образовательных игр.

Основные задачи исследования:

- на основе анализа учебной, научно-технической литературы сформулировать концепцию к разрабатываемой обучающей игре;
- рассмотреть этапы планирования и разработки игры в стиле «Герой-программист»;
- разработать браузерную игру.

В ходе выполнения выпускной квалификационной работы спроектирована и разработана игра в стиле «Герой-программист» для обучения программированию.

СОДЕРЖАНИЕ

Введение.....	4
1 Анализ предметной области	7
1.1 Актуальность разработки обучающих игр	9
1.2 Обучение программированию с использованием игр.....	13
1.3 Методы и технологии разработки обучающих игр	17
1.4 Концепция игры «Герой-программист»	21
2. Разработка игры в стиле «Герой-программист» для обучения программированию	24
2.1 Установка программного обеспечения.....	24
2.2 Разработка графического оформления	26
2.3 Функциональное тестирование.....	30
Заключение	33
Список использованных источников	35
Приложение А Листинг HTML кода отвечающий за отображения лабиринта..	39
Приложение Б Листинг javascript кода для отображения поля ввода команд....	42
Приложение В Листинг HTML кода для связи лабиринта с полем ввода	46
Приложение Г Скрипт javascript для связи уровней лабиринта.....	49

ВВЕДЕНИЕ

Цифровая революция оказала огромное влияние на общество и образование, особенно за счёт развития информационных технологий, включая компьютерные игры. Эти игры стали неотъемлемой частью нашей культуры, проникая в различные сферы жизни и обучения.

«Компьютерные игры – это вид интерактивных развлечений, созданных с помощью компьютерной технологии. Они представляют собой программы, которые выполняются на компьютере или другом устройстве и позволяют пользователю взаимодействовать с виртуальным миром, созданным внутри игры.» [7, с.97].

Сегодня компьютерные игры не только популярны, но и играют важную роль в развитии индивида и общества. Они представляют собой уникальное средство для моделирования различных ситуаций, позволяя нам исследовать альтернативные пути решения проблем. Благодаря этому, игры стимулируют креативное мышление, развивают навыки принятия решений и способствуют формированию личности.

Современные технологии делают компьютерные игры ещё более захватывающими и реалистичными. Улучшенная графика, звуковое сопровождение и возможности виртуальной реальности погружают нас в удивительные миры и помогают пережить разнообразные эмоции.

В современном мире компьютерные игры не просто развлечение, а инструмент активного обучения и развития. Их влияние на формирование личности и культуры общества становится всё более значимым в эпоху цифровизации и технологического прогресса.

Актуальность исследования заключается в привлечении и мотивации учащихся с помощью геймификации, которая способствует эффективному овладению навыками программирования.

Цель исследования: разработать образовательную игру в стиле «Герой-программист» для обучения основам программирования.

Объект исследования: обучение программированию с использованием образовательных игр.

Предмет исследования: методы и технологии разработки образовательных игр.

Основные задачи исследования:

– на основе анализа учебной, научно-технической литературы сформулировать концепцию к разрабатываемой обучающей игре;

– рассмотреть этапы планирования и разработки игры в стиле «Герой-программист»;

– разработать браузерную игру.

В работе использовались теоретические методы исследования, такие как анализ учебной и научно-технической литературы по теме исследования, обобщение, а также методы, относящиеся к разработке программного обеспечения.

Разработанная компьютерная игра в стиле «Герой-программист» имеет практическое значение для обучения основам программирования для учащихся от 12 лет. Выбор такой категории обуславливается тем, что дети в 12 лет уже спокойно умеют читать и анализировать информацию. Она может быть полезна педагогам как инструмент для проведения интересных и познавательных внеклассных мероприятий. Материалы исследования могут служить источником информации для студентов при подготовке различных научно-исследовательских работ, таких как статьи, рефераты, курсовые и выпускные квалификационные работы.

Результаты исследования представлены на следующих научных мероприятиях:

1. VII Всероссийской научно-практической конференции «Актуальные проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирска, ЛПИ – филиал СФУ, 01-02 ноября 2023 г., участие).

2. II Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ 10-15 апреля 2023 г., участие).

Принята к публикации статья:

1. Хабибулин, Д.С. Проектирование и разработка игры в стиле «Герой-программист» для обучения программированию / Д.С. Хабибулин // Всероссийская научно-практическая конференция «Современное педагогическое образование: теоретический и прикладной аспекты» – Лесосибирск, 2024 – 3 стр.

1 Анализ предметной области

В век информационных технологий программирование становится востребованным навыком во многих сферах. Традиционные методы обучения программированию, основанные на лекциях и решении задач, могут показаться скучными и сложными для многих людей. Игры же, с другой стороны, способны заинтересовать и увлечь пользователей, делая процесс обучения более эффективным и мотивирующим.

Актуальность разработки игр в контексте обучения программированию обуславливается их потенциалом в:

Игры, основанные на геймификации, позволяют сделать процесс обучения более увлекательным и интерактивным. Это может привести к повышению интереса к изучению программирования и увеличению времени, которое учащиеся готовы посвятить этому занятию. Геймификация включает элементы, такие как уровни, очки, награды и достижения, которые стимулируют учащихся к продолжению обучения и совершенствованию своих навыков.

Игры погружают игроков в виртуальный мир, где они могут применять свои знания и навыки на практике, решая задачи и преодолевая препятствия. Это повышает вовлеченность учащихся в процесс обучения и делает его более эффективным. Вовлеченность увеличивается за счет интерактивных элементов, таких как сюжетная линия, персонажи и динамические события, которые делают обучение более захватывающим и интересным.

Игры позволяют учащимся в игровой форме практиковать полученные знания, что улучшает их запоминание и применение. Кроме того, игры способствуют развитию таких навыков, как логическое мышление, критическое мышление и решение проблем, которые необходимы для успешного обучения программированию. Практическое применение теоретических знаний в игровой среде помогает учащимся лучше понимать и запоминать материал, а также учит их решать реальные задачи.

Анализ научно-технической литературы по данной теме показывает, что существует множество различных подходов к разработке обучающих игр. Некоторые игры фокусируются на обучении синтаксису конкретного языка программирования. Эти игры позволяют учащимся изучить основы языка программирования и научиться писать простые программы. Примеры таких игр включают платформы, где игроки пишут код для управления персонажами или решения логических задач.

Другие игры фокусируются на развитии навыков решения проблем и логического мышления. Эти игры предлагают учащимся задачи, которые они должны решить, используя свои знания программирования. Решение этих задач позволяет учащимся развить навыки, которые необходимы для успешного программирования в реальном мире. Например, игры, где игроки создают алгоритмы для преодоления препятствий или оптимизации процессов.

Примеры успешных образовательных игр

CodeCombat – это игра, в которой игроки изучают программирование, создавая код для управления своими персонажами. Игра охватывает различные языки программирования, включая Python и JavaScript, и предлагает множество уровней с возрастающей сложностью. Игроки учатся писать код, решать логические задачи и разрабатывать стратегии для прохождения уровней.

Scratch – это платформа, разработанная MIT, которая позволяет детям создавать свои собственные интерактивные истории, игры и анимации с использованием визуального языка программирования. Scratch упрощает процесс обучения программированию, предоставляя блоки кода, которые можно перетаскивать и комбинировать. Это помогает учащимся понять основные концепции программирования, такие как циклы, условия и переменные, в легкой и доступной форме.

Lightbot – это головоломка, которая учит основам программирования через управление роботом, чтобы он проходил уровни, решая логические задачи. Игроки создают последовательности команд для управления роботом, используя концепции процедурного программирования, такие как циклы и условные

выражения. Это помогает учащимся развить навыки алгоритмического мышления и решения проблем.

Перед началом разработки игры важно определить конкретные образовательные цели. Это могут быть цели, связанные с изучением синтаксиса языка программирования, развитием логического мышления, навыков решения проблем или практическим применением теоретических знаний.

Игра должна быть интерактивной и увлекательной, чтобы удерживать интерес учащихся. Это можно достичь с помощью увлекательных сюжетных линий, интересных персонажей, динамических событий и стимулирующих задач. «Геймификация в образовании – это некая механика или набор инструментов, которые позволяют существенно разнообразить учебный процесс и привнести в него как развлекательную составляющую, так и учебную, социальную и мотивационную.» [20, с. 99].

Игра должна предоставлять возможности для практического применения знаний. Это могут быть задачи, требующие написания кода, решение логических головоломок, управление персонажами или создание собственных проектов. Практическое применение помогает учащимся лучше понять и запомнить материал, а также развивает навыки решения реальных задач.

Использование игр в обучении программированию представляет собой перспективное направление, способное значительно повысить мотивацию, вовлеченность и эффективность усвоения материала. Грамотно разработанные образовательные игры могут стать мощным инструментом для преподавателей, способствующим развитию важных навыков у учащихся и облегчению процесса изучения программирования.

1.1 Актуальность разработки обучающих игр

В современном мире программирование становится не просто востребованным навыком, а необходимым условием для успешной карьеры и самореализации. Однако традиционные методы обучения программированию, основанные на лекциях, чтении учебников и решении задач, могут показаться

скучными и сложными для многих людей. Это приводит к тому, что многие люди не хотят или не могут освоить этот важный навык.

Вопросу об обучающих играх, их структуре и функциях уделяется внимание в работах таких исследователей как А. А. Попов [29], Н. А. Прохоренок [31], Т. В. Рябова [36].

Традиционные методы обучения часто включают в себя лекции, чтение учебников и выполнение абстрактных задач. Хотя эти методы могут быть эффективными для некоторых студентов, они не всегда учитывают различные стили обучения и мотивации студентов. Сложность и монотонность таких методов могут оттолкнуть многих потенциальных учеников, особенно тех, кто предпочитает практическое и интерактивное обучение. Например, обучающиеся могут столкнуться с такими проблемами, как:

- сложность восприятия теории: без практического применения абстрактные концепции программирования могут быть трудными для понимания;

- низкая мотивация: отсутствие интерактивных элементов и мгновенной обратной связи может снижать интерес и мотивацию;

- однообразие заданий: стандартные задачи и упражнения могут не учитывать личные интересы и уровни подготовки студентов.

Обучающие компьютерные игры позволяют решить эту проблему. Игры, основанные на геймификации, делают процесс обучения более увлекательным, интерактивным и мотивирующим. Это позволяет привлечь к изучению программирования широкую аудиторию: включая детей и людей, которые ранее не интересовались программированием. Игры могут быть разработаны так, чтобы быть доступными и привлекательными для различных возрастных групп и уровней подготовки.

Обучающие игры позволяют:

– повысить мотивацию учащихся к изучению программирования: игровые элементы, такие как уровни, награды и достижения, могут стимулировать студентов к продолжению обучения и достижению новых высот;

– сделать процесс обучения более эффективным и улучшить усвоение материала: через практическое применение знаний в игровой форме, студенты могут лучше запоминать и понимать материал.

Обучающие игры имеют ряд преимуществ перед традиционными методами обучения:

1. Погружение в виртуальный мир

Игры позволяют учащимся погрузиться в виртуальный мир, где они могут применять свои знания и навыки на практике. Это делает процесс обучения более реалистичным и увлекательным. Например, студенты могут управлять персонажем-программистом, который решает различные задачи, что позволяет им видеть непосредственные результаты своих действий.

2. Интерактивность

Игры позволяют учащимся взаимодействовать с виртуальным миром и другими игроками. Это делает процесс обучения более динамичным и позволяет учащимся получать мгновенную обратную связь. Например, в многопользовательских играх студенты могут сотрудничать или соревноваться с другими игроками, что способствует развитию командных навыков и повышает мотивацию.

3. Геймификация

Игры используют элементы геймификации, такие как очки, уровни, награды и рейтинги. Это повышает мотивацию учащихся к изучению программирования. Элементы геймификации могут стимулировать студентов к выполнению дополнительных заданий, исследованию новых тем и постоянному совершенствованию своих навыков.

4. Персонализация

Игры позволяют персонализировать процесс обучения в соответствии с индивидуальными потребностями и уровнем подготовки каждого учащегося. Например, система может адаптироваться к уровню знаний студента, предлагая более сложные задачи по мере его прогресса, или предоставлять подсказки и дополнительную помощь тем, кто испытывает трудности.

Конкретные примеры и исследования

Научно-исследовательские работы и практические примеры демонстрируют эффективность обучающих игр в контексте программирования:

- CodeCombat используется в образовательных учреждениях повышает успеваемость и интерес студентов к программированию. Игра позволяет студентам учиться, создавая код для управления персонажами, что способствует развитию практических навыков;

- Scratch помогает детям и подросткам лучше понимать основы программирования и логического мышления через создание собственных интерактивных проектов. Платформа позволяет учащимся видеть результат их работы в реальном времени, что делает процесс обучения более осмысленным и интересным.

Разработка обучающих игр является актуальной и перспективной задачей. Однако создание эффективных образовательных игр требует учета ряда факторов:

- качество контента: игры должны содержать качественный образовательный контент, соответствующий учебным стандартам и целям;

- баланс между игрой и обучением: важно найти баланс между развлекательными элементами и образовательными задачами, чтобы игра оставалась увлекательной, но при этом способствовала обучению;

- технические и дизайнерские аспекты: необходимо учитывать технические возможности и ограничения, а также разрабатывать удобный и интуитивно понятный интерфейс.

Обучающие компьютерные игры представляют собой мощный инструмент для повышения мотивации, вовлеченности и эффективности обучения программированию. Они способны привлечь к изучению программирования широкую аудиторию и сделать процесс обучения более доступным и интересным. В перспективе, развитие таких игр может значительно изменить подход к обучению программированию и способствовать развитию необходимых навыков у большого числа людей.

1.2 Обучение программированию с использованием игр

Обучение программированию – это сложный процесс, который требует усидчивости, логического мышления и аналитических способностей. Традиционные методы обучения, основанные на лекциях, чтении учебников и решении задач, могут показаться скучными и сложными для многих людей. Это приводит к тому, что многие люди не хотят или не могут освоить этот важный навык.

Традиционные методы обучения часто включают в себя лекции, чтение учебников и выполнение абстрактных задач. Хотя эти методы могут быть эффективными для некоторых студентов, они не всегда учитывают различные стили обучения и мотивации студентов. Сложность и монотонность таких методов могут оттолкнуть многих потенциальных учеников, особенно тех, кто предпочитает практическое и интерактивное обучение.

Опираясь на работы таких исследователей как Д. Р. Бартон [5], Ф. Джейсон [10], В.В. Касихин [23], выявим основные проблемы:

– сложность восприятия теории: без практического применения абстрактные концепции программирования могут быть трудными для понимания;

– низкая мотивация: отсутствие интерактивных элементов и мгновенной обратной связи может снижать интерес и мотивацию;

– однообразие заданий: стандартные задачи и упражнения могут не учитывать личные интересы и уровни подготовки студентов.

Использование игр в процессе обучения программированию позволяет сделать его более увлекательным, интерактивным и мотивирующим. Игры помогают погрузить учащихся в виртуальный мир, где они могут применять свои знания и навыки на практике. Это делает процесс обучения более реалистичным и эффективным.

В своих работах Ю. А. Алексеев [2], Д. А. Винокуров [8] указывают, что использование компьютера эффективно помогает в развитии творческих способностей детей.

Теоретические основы использования игр

Существует ряд теоретических основ, на которых основано использование игр в процессе обучения программированию:

1. Теория геймификации

«Геймификация – это использование игровых элементов в неигровом контексте. В контексте обучения программированию геймификация позволяет сделать процесс обучения более увлекательным и мотивирующим. Элементы геймификации включают очки, уровни, награды и рейтинги, которые стимулируют учащихся к достижению образовательных целей.» [22, с.62].

Исследование результатов внедрения геймификации в образовательный процесс позволяет сделать выводы о том, что для этого процесса, по мнению А. Л. Мазелиса [26], игра является родовым понятием, а использование игры (игровых механик, сюжета и сценария игры, методов к неигровым видам деятельности) вызывает мотивацию и вовлеченность, изменение поведения обучающихся.

2. Теория активного обучения

«Активное обучение – это метод обучения, который фокусируется на практической деятельности учащихся.» [4, с.75]. Игры являются отличным способом активного обучения программированию, так как они требуют от учащихся взаимодействия с контентом и принятия активного участия в процессе

обучения. Практическая деятельность способствует лучшему пониманию и запоминанию материала.

Игры позволяют учащимся самостоятельно строить знания о программировании, предоставляя им возможность экспериментировать, решать проблемы и находить собственные решения. Это способствует развитию критического мышления и самостоятельности.

Использование игр в процессе обучения программированию имеет ряд преимуществ:

1. Повышение мотивации

Игры повышают мотивацию учащихся к изучению программирования. Элементы геймификации, такие как очки, уровни, награды и рейтинги, стимулируют учащихся к продолжению обучения и достижению новых высот.

2. Развитие навыков

Игры позволяют развить навыки логического мышления, решения проблем, критического мышления и другие навыки, которые необходимы для успешного программирования. Например, решая задачи в игре, учащиеся учатся разрабатывать алгоритмы, оптимизировать решения и анализировать свои действия.

3. Улучшение усвоения материала

Игры позволяют улучшить усвоение материала по программированию. Практическое применение знаний в игровой форме помогает учащимся лучше запоминать и понимать материал. Игры также предоставляют мгновенную обратную связь, что помогает учащимся осознавать и исправлять свои ошибки.

4. Снижение стресса

Игры позволяют снизить стресс и сделать процесс обучения более приятным. Игровая среда создает безопасное пространство для экспериментов и ошибок, что способствует более расслабленному и позитивному отношению к обучению.

Недостатки использования игр в обучении программированию

Использование игр в процессе обучения программированию имеет ряд недостатков:

1. Высокая стоимость разработки

Разработка игр может быть дорогостоящим процессом. Это включает затраты на дизайн, программирование, тестирование и поддержку игр. Высокая стоимость может быть барьером для внедрения игровых технологий в образовательные учреждения.

2. Сложность разработки

Разработка игр может быть сложным процессом, который требует специальных знаний и навыков. Создание качественных образовательных игр требует междисциплинарного подхода, включая педагогов, программистов, дизайнеров и психологов.

3. Ограниченность возможностей

Игры могут быть не так эффективны, как традиционные методы обучения в некоторых случаях. Например, некоторые сложные концепции могут требовать более глубокого теоретического изучения, которое может быть трудно реализовать в игровой форме. Также игры могут быть ограничены в охвате учебного материала.

Рассмотрим примеры успешных образовательных игр, обучающие программированию.

CodeCombat – это игра, в которой игроки изучают программирование, создавая код для управления своими персонажами. Игра охватывает различные языки программирования, включая Python и JavaScript, и предлагает множество уровней с возрастающей сложностью. Игроки учатся писать код, решать логические задачи и разрабатывать стратегии для прохождения уровней.

Scratch – это платформа, разработанная MIT, которая позволяет детям создавать свои собственные интерактивные истории, игры и анимации с использованием визуального языка программирования. Scratch упрощает процесс обучения программированию, предоставляя блоки кода, которые можно перетаскивать и комбинировать. Это помогает учащимся понять основные

концепции программирования, такие как циклы, условия и переменные, в легкой и доступной форме.

Lightbot – это головоломка, которая учит основам программирования через управление роботом, чтобы он проходил уровни, решая логические задачи. Игроки создают последовательности команд для управления роботом, используя концепции процедурного программирования, такие как циклы и условные выражения. Это помогает учащимся развить навыки алгоритмического мышления и решения проблем.

Несмотря на некоторые недостатки, использование игр является эффективным способом обучения программированию. Игры позволяют сделать процесс обучения более увлекательным, интерактивным и мотивирующим. Это позволяет привлечь к изучению программирования широкую аудиторию, в том числе детей и людей, которые не интересовались программированием ранее. Развитие образовательных игр в будущем может значительно изменить подход к обучению программированию, делая его более доступным, эффективным и увлекательным для всех.

1.3 Методы и технологии разработки обучающих игр

Разработка обучающей игры в стиле «Герой-программист» имеет свои особенности, которые необходимо учитывать при выборе методов и технологий. Так как игра будет реализована в виде браузерной игры, рассмотрим некоторые из наиболее популярных методов и технологий, которые подходят для разработки игр.

«Web-приложение – это прикладное программное обеспечение, которое работает на веб-сервере, в отличие от компьютерных программ, которые запускаются локально в операционной системе.» [38, с.78].

Web-приложения предоставляют пользователю интерфейс для взаимодействия и выполнения задач, такие как регистрация и вход в систему,

просмотр, добавление и редактирование данных, обмен сообщениями и другие операции, в зависимости от целей и требований приложения.

Рассмотрим языки программирования с помощью, которых можно разработать обучающую web-игру

1. «HTML используется для создания основной структуры web-страницы, включая заголовки, параграфы, списки, таблицы и другие элементы.» [13, с.56].

Применение HTML позволяет определить различные типы контента, такие как текст, изображения, видео, аудио и другие медиа элементы.

2.CSS используется для задания внешнего вида и стилей web-страниц. Он позволяет определить цвета, шрифты, размеры, отступы, границы, фоны и другие аспекты визуального оформления.

Применение CSS позволяет стилизовать HTML-элементы, задавая внешний вид и расположение. В контексте игры, CSS используется для оформления интерфейсов, анимации и адаптивного дизайна.

3. JavaScript

Разработанные с использованием Java и его фреймворков, таких как Spring или JavaServer Faces (JSF). «Java web-приложения могут быть масштабируемыми и надежными.» [19, с.73].

JavaScript язык программирования, который используется для добавления интерактивности на веб-страницы.

Применение JavaScript используется для добавления динамического контента и создания интерактивности на web-страницах. Он позволяет создавать анимации, обрабатывать события, выполнять валидацию форм, обмениваться данными с сервером без перезагрузки страницы и многое другое.

4. TypeScript язык программирования, надмножество JavaScript, который позволяет писать более структурированный и читаемый код.

Применение TypeScript добавляет статическую типизацию и другие современные возможности в JavaScript, что помогает улучшить качество кода и упростить его поддержку.

«Web-приложения работают на сервере и обычно взаимодействуют с клиентскими устройствами (компьютеры, смартфоны, планшеты) через интернет. Пользователи могут получать доступ к web-приложениям через URL-адрес, запуская их в web-браузере без необходимости установки специального программного обеспечения.» [14, с.43].

Web-приложения доступны через web-браузеры, такие как Yandex Browser, Mozilla Firefox, Safari или Internet Explorer. Это означает, что пользователи могут получить доступ к web-приложению с любого устройства, имеющего доступ к интернету, и в любое удобное время.

«Web-приложения предоставляют пользователю интерфейс для взаимодействия и выполнения задач, такие как регистрация и вход в систему, просмотр, добавление и редактирование данных, обмен сообщениями и другие операции, в зависимости от целей и требований приложения.» [27, с.54].

Библиотеки и фреймворки:

1. Phaser библиотека JavaScript, которая специально разработана для разработки 2D-игр. Phaser предоставляет множество инструментов для создания 2D-игр, таких как физика, анимация, управление ресурсами и звуковыми эффектами.

2. Pixi.js библиотека JavaScript, которая позволяет создавать 2D-игры с высокой производительностью. Pixi.js использует WebGL и Canvas API для рендеринга графики, обеспечивая высокую производительность и гибкость в разработке игр.

3. «Three.js – это библиотека JavaScript, которая позволяет создавать 3D-игры. Three.js предоставляет удобные инструменты для работы с 3D-графикой, включая рендеринг, освещение, текстурированные и физику.» [21, с.89]

4. Babylon.js фреймворк JavaScript, который позволяет создавать 3D-игры и приложения. Babylon.js предлагает полный набор инструментов для разработки 3D-игр, включая поддержку различных форматов моделей, анимации и виртуальной реальности (VR).

Инструменты разработки

1. Visual Studio Code бесплатный редактор кода, поддерживающий множество языков программирования, в том числе HTML, CSS и JavaScript.

Применение Visual Studio Code предоставляет удобную среду для написания кода, отладки и интеграции с системами контроля версий. Он также поддерживает расширения для улучшения функциональности.

Visual Studio не ограничивается только редактированием и отладкой кода, в функционал входят компиляторы, инструменты авто завершения кода, системы управления версиями и множество расширений и дополнительных возможностей, направленных на улучшение этапа процесса разработки программного обеспечения.

2. WebStorm Платная IDE, которая специально разработана для разработки веб-приложений. WebStorm предоставляет множество инструментов для разработки, отладки и тестирования веб-приложений, что делает его отличным выбором для профессиональных разработчиков.

3. Chrome DevTools инструменты разработки, встроенные в браузер Google Chrome. Применения Chrome DevTools позволяют отлаживать код, просматривать сетевой трафик, анализировать производительность веб-страниц и выявлять проблемы с доступностью.

Мобильные браузеры

При разработке браузерных игр необходимо учитывать особенности разных мобильных браузеров, так как пользователи могут играть на различных устройствах.

1. Google Chrome предоставляет мощные инструменты и API для работы с веб-приложениями, включая поддержку PWA (прогрессивных веб-приложений).

2. Safari мобильный браузер, предустановленный на устройствах Apple. Применение safari поддерживает современные веб-технологии и предлагает инструменты для оптимизации производительности на устройствах Apple.

3. Firefox бесплатный мобильный браузер, применение firefox предлагает хорошую поддержку веб-стандартов и мощные инструменты для разработки и отладки веб-приложений.

Доступность

При разработке браузерных игр важно учитывать принципы доступности, чтобы игра была доступна для людей с ограниченными возможностями.

1. Использование альтернативных текстов для изображений

Альтернативные тексты помогают пользователям с нарушениями зрения понять содержание изображений с помощью экранных читалок.

2. Обеспечение возможности управления игрой с помощью клавиатуры

Эта возможность позволяет пользователям, которые не могут использовать мышь, управлять игрой с помощью клавиатуры.

3. Использование контрастных цветов для текста и фона

Хороший контраст между текстом и фоном улучшает читаемость для пользователей с нарушениями зрения.

Использование перечисленных методов и технологий позволяет разработать качественную и увлекательную образовательную браузерную игру. Комплексный подход к разработке, включающий выбор правильных языков программирования, библиотек, фреймворков и инструментов, а также внимание к доступности и поддержке мобильных устройств, обеспечивает создание эффективного образовательного продукта.

Браузерные игры, построенные на этих принципах, имеют потенциал для широкого применения в образовательной сфере, способствуя повышению интереса и мотивации к изучению программирования, улучшению усвоения материала и развитию необходимых навыков у учащихся.

1.4 Концепция игры «Герой-программист»

Концепция предлагаемой игры предусматривает создание браузерной игры на языке программирования Pascal, где пользователь будет перемещаться по лабиринту с целью достижения ценного ресурса – алмаза. Игра разрабатывается с учетом педагогических аспектов, направленных на развитие логического мышления и программирования у игрока.

Основные аспекты концепции:

1. Образовательные цели

Целью игры является улучшение навыков программирования на языке Pascal, а также развитие логического мышления. Игра ставит перед игроком задачу добраться до алмаза в лабиринте, что способствует применению базовых конструкций языка программирования.

2. Интерфейс и визуальный дизайн

Визуальное оформление игры будет выполнено с учетом простоты и функциональности для удобства пользователей. Интерфейс будет интуитивно понятным, что позволит пользователям легко взаимодействовать с игрой через браузер.

3. Игровой процесс и механика

Игрок будет управлять персонажем, используя управление через написание кода с помощью клавиш клавиатуры, с целью найти кратчайший путь до алмаза в лабиринте. Лабиринт будет иметь разные уровни сложности, что позволит игрокам постепенно углубляться в изучение и практику языка Pascal.

4. Технические аспекты

Игра будет реализована с использованием веб-технологий, что обеспечит доступность игры через браузер на различных платформах без необходимости установки дополнительного ПО. Для реализации игровой механики и визуального оформления будут использованы возможности языка программирования HTML и возможности веб-разработки.

5. Оценка эффективности

Эффективность игры будет оцениваться по достижению образовательных целей, а также через сбор обратной связи от пользователей. Предусмотрены механизмы для сбора данных о прогрессе игроков и их взаимодействии с игрой для последующего анализа и улучшения концепции.

Таким образом, разработка игры для обучения языка Pascal, где игроку предстоит перемещаться по лабиринту с целью достижения алмаза, сочетает в

себе педагогические принципы и игровые механики для достижения образовательных целей.

2. Разработка игры в стиле «Герой-программист» для обучения программированию

Разработка игры – это комплексный процесс создания игры от идеи до релиза.

Он включает в себя:

- дизайн: разработка игрового мира, персонажей, сюжета, уровней.
- программирование: создание кода для игровой логики, искусственного интеллекта, физики и других систем.
- релиз: выпуск игры на целевых платформах.
- поддержка: исправление ошибок.

Разработка игр – это творческий и увлекательный процесс, который требует различных навыков и знаний.

В этом параграфе рассмотрим разработку игры в стиле «Герой-программист для обучения программирования».

Создание игры – это увлекательное путешествие, где творческие идеи обретают форму, а фантазии оживают на экранах. Но, как и в любом путешествии, важен маршрут.

Планирование разработки игры выступает в роли путеводной звезды, помогая избежать подводных камней и достичь желаемой цели.

На этапе планирования закладывается фундамент проекта, определяются его цели, задачи и основные элементы.

Без него невозможно ни оптимизировать ресурсы, ни создать продукт, который захватит сердца игроков.

2.1 Установка программного обеспечения

Для написания программного кода выбрал Microsoft Visual Studio Code, для его установки перешли на официальный сайт и выбрали подходящий

тарифный план для пользователей, загрузка Microsoft Visual Studio Code представлена на рисунке 1.

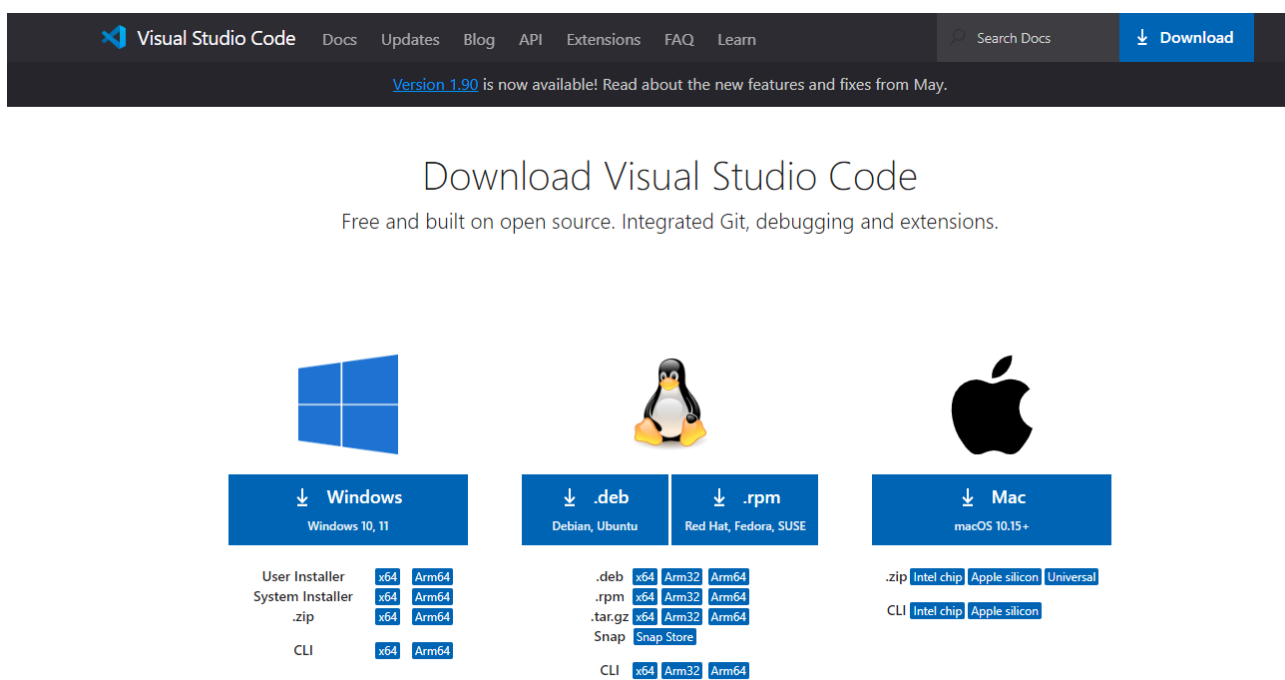


Рисунок 1– Установка MVS 2022

Далее создадим пустой файл, в котором будем разрабатывать игру, для это перейдем в раздел «File» -> «New File» (или используйте горячую клавишу Ctrl+N) для создания нового файла, первоначальный файл представлен на рисунке 2.

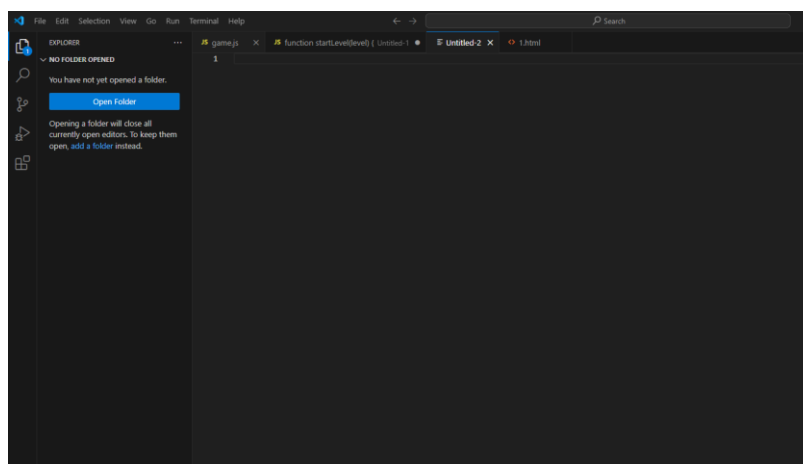


Рисунок 2– Первоначальный файл

Далее приступим к разработке графического оформления игры.

2.2 Разработка графического оформления

Графическое оформление игры: искусство создания виртуальных миров

Графическое оформление игры – это искусство создания визуальной составляющей игры, которая захватывает внимание, погружает в атмосферу и делает игровой процесс незабываемым.

Оно включает в себя:

- дизайн персонажей: создание уникальных образов героев, злодеев, второстепенных персонажей, которые запоминаются и отражают суть игры;
- создание игрового мира: разработка детально проработанных локаций, неповторимых пейзажей, атмосферных объектов, которые оживляют виртуальную вселенную;
- дизайн интерфейса: обеспечение удобного и понятного взаимодействия игрока с игрой, используя элементы меню, иконки, шрифты и другие элементы;
- визуальные эффекты: создание спецэффектов, анимации и освещения, которые добавляют реалистичности и динамизма игровому процессу;

Графическое оформление должно соответствовать жанру игры, целевой аудитории и техническим возможностям.

Графическое оформление – это не просто картинка, а важная часть игры, которая влияет на ее восприятие и успех.

Продуманная и качественная графика способна сделать даже самую простую игру увлекательной и захватывающей.

Лабиринт представляется в виде двумерного массива фиксированного размера, где каждый элемент массива представляет собой одну клетку лабиринта. Структура лабиринта включает стены, пустые пространства, начальную позицию игрока и позицию алмаза. На рисунке 3 представлен листинг кода с параметрами лабиринта.

```
const WIDTH = 10;
const HEIGHT = 10;
const PLAYER = 'P';
const DIAMOND = 'D';
const WALL = '#';
const EMPTY = ' ';
```

Рисунок 3 – Листинг кода лабиринта

WIDTH и HEIGHT задают размеры лабиринта (10x10).

Константы PLAYER, DIAMOND, WALL и EMPTY обозначают символы для игрока, алмаза, стены и пустого пространства соответственно.

Инициализация лабиринта

На рисунке 4 изображен двумерный массив, инициализированный пустыми клетками.

```
function createMaze(width, height) {
  const maze = Array.from({ length: height }, () => Array(width).fill(EMPTY));

  // Create walls
  for (let i = 0; i < width; i++) {
    maze[0][i] = WALL;
    maze[height - 1][i] = WALL;
  }
  for (let i = 0; i < height; i++) {
    maze[i][0] = WALL;
    maze[i][width - 1] = WALL;
  }

  // Random walls
  for (let i = 0; i < 20; i++) {
    const x = Math.floor(Math.random() * (width - 2)) + 1;
    const y = Math.floor(Math.random() * (height - 2)) + 1;
    maze[y][x] = WALL;
  }

  return maze;
}
```

Рисунок 4 – Инициализация лабиринта

Функция `createMaze` создает двумерный массив и инициализирует его пустыми клетками. Затем по периметру массива добавляются стены. Также случайным образом добавляются дополнительные стены внутри лабиринта для увеличения сложности. Внешние циклы `for` добавляют стены по периметру лабиринта. Внутренний цикл `for` добавляет случайные стены внутри лабиринта.

Визуализация лабиринта

Функция `renderMaze` отвечает за отображение лабиринта на веб-странице. Она создает HTML-элементы для каждой клетки лабиринта и добавляет соответствующие CSS-классы для визуального представления стен, игрока и алмаза. На рисунке 5 представлена визуализация лабиринта игры.

`gameContainer` – это HTML-элемент, содержащий лабиринт. Вложенные циклы `for` проходят по каждой клетке лабиринта. Внешний цикл «`for`» итерируется по строкам, а внутренний цикл `for` – по колонкам внутри каждой строки.

Для каждой клетки создается элемент `div`, которому назначаются CSS-классы в зависимости от содержимого клетки. Если клетка представляет стену, ей присваивается класс `wall`. Если клетка содержит игрока, используется класс `player`. Для клетки с алмазом назначается класс `diamond`. Если клетка является пустым пространством, применяется класс `empty`.

Эти классы определяют визуальное представление клетки с помощью CSS, позволяя легко изменять стили и добавлять анимации или другие эффекты. Использование семантических CSS-классов также упрощает поддержку и модификацию кода в будущем. Кроме того, такая структура позволяет легко расширять функциональность игры, добавляя новые типы клеток или изменяя существующие, без необходимости значительных изменений в коде.

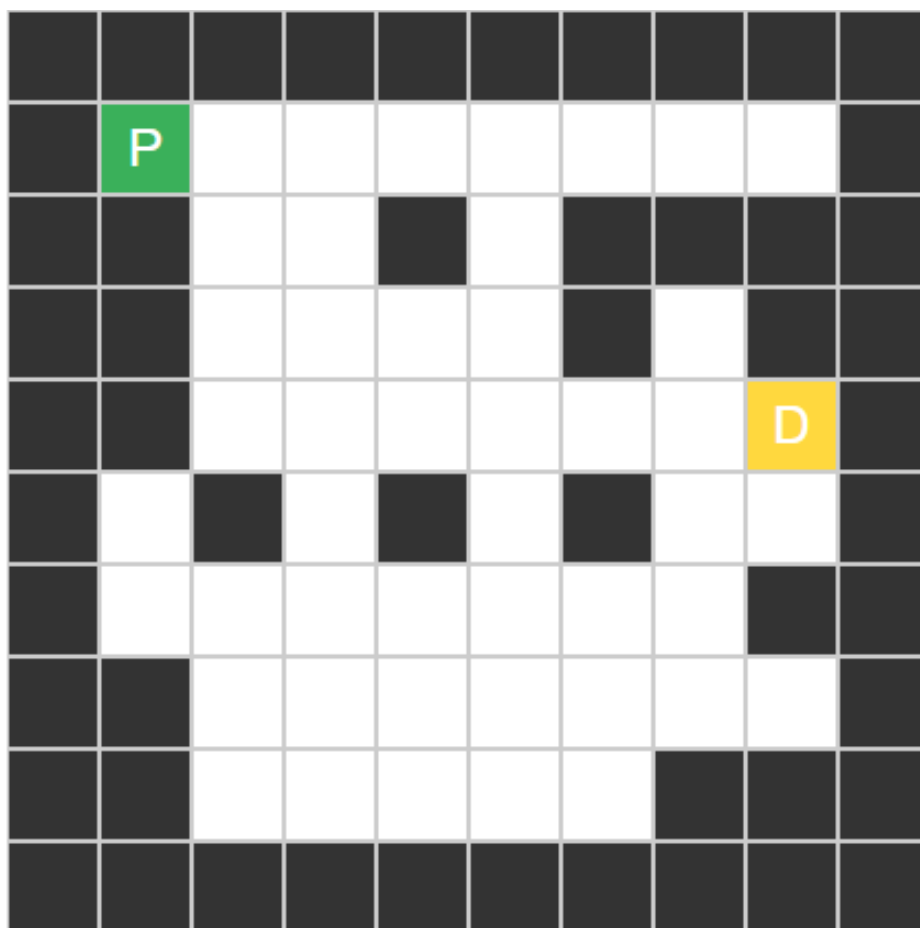


Рисунок 5 – Визуализация лабиринта

Создание анимации персонажа

Функция `movePlayer` изменяет позицию игрока в лабиринте на основе команды перемещения (вверх, вниз, влево, вправо). Функция проверяет допустимость перемещения (чтобы не выйти за границы лабиринта и не столкнуться со стеной) и обновляет позицию игрока. На рисунке 6 представлен код программы, отвечающий за перемещение персонажа. `direction` – строка, определяющая направление перемещения.

Проверки `if` гарантируют, что игрок не выйдет за пределы лабиринта и не переместится на клетку со стеной.

Позиция игрока обновляется, и лабиринт перерисовывается с помощью функции `renderMaze`.

```

1 function movePlayer(direction) {
2   let newX = playerPos.x;
3   let newY = playerPos.y;
4   if (direction === 'up' && playerPos.y > 0 && maze[playerPos.y - 1][playerPos.x] !== WALL) {
5     newY -= 1;
6   } else if (direction === 'down' && playerPos.y < HEIGHT - 1 && maze[playerPos.y + 1][playerPos.x] !== WALL) {
7     newY += 1;
8   } else if (direction === 'left' && playerPos.x > 0 && maze[playerPos.y][playerPos.x - 1] !== WALL) {
9     newX -= 1;
10  } else if (direction === 'right' && playerPos.x < WIDTH - 1 && maze[playerPos.y][playerPos.x + 1] !== WALL) {
11    newX += 1;
12  }
13  playerPos = { x: newX, y: newY };
14  renderMaze();
15  checkWin();

```

Рисунок 6 – Перемещение персонажа

2.3 Функциональное тестирование

Функция checkWin проверяет, достиг ли игрок позиции алмаза. В случае успеха игра переходит к следующему уровню или выводит сообщение о завершении всех уровней. На рисунке 7 представлен код программы, отвечающий за проверку что персонаж дошел до алмаза, что означает успешное прохождение уровня.

```

function checkWin() {
  if (playerPos.x === diamondPos.x && playerPos.y === diamondPos.y) {
    setTimeout(() => {
      alert("Congratulations! You found the diamond!");
      currentLevel++;
      if (currentLevel < levels.length) {
        loadLevel(currentLevel);
      } else {
        alert("You've completed all levels!");
        showMainMenu();
      }
    }, 100);
  }
}

```

Рисунок 7 – Проверка победы

Если позиция игрока совпадает с позицией алмаза, выводится сообщение о победе.

Игра переходит к следующему уровню или возвращает игрока в главное меню, если все уровни завершены.

Запуск уровней и управление прогрессом

Функции `startLevel`, `loadLevel`, `showMainMenu`, `showInstructions` и `resetProgress` управляют загрузкой уровней, отображением меню и инструкций, а также сбросом игрового прогресса.

`startLevel` запускает выбранный уровень.

`loadLevel` загружает данные уровня и отображает соответствующий лабиринт.

`showMainMenu` отображает главное меню игры.

`showInstructions` выводит инструкции для игрока.

`resetProgress` сбрасывает прогресс игрока.

На рисунке 8 код отвечающий за запуск уровней.

```
function startLevel(level) {
  currentLevel = level;
  loadLevel(level);
}

function loadLevel(level) {
  const levelData = _levels[level];
  playerPos = { ...levelData.start };
  diamondPos = { ...levelData.diamond };
  levelTitle.textContent = levelData.title;
  commandInput.value = levelData.codeTemplate;
  levelTitle.style.display = 'block';
  gameContainer.style.display = 'grid';
  commandInput.style.display = 'block';
  executeButton.style.display = 'block';
  mainMenu.style.display = 'none';
  renderMaze();
}

function showMainMenu() {
  levelTitle.style.display = 'none';
  gameContainer.style.display = 'none';
  commandInput.style.display = 'none';
  executeButton.style.display = 'none';
  mainMenu.style.display = 'flex';
}

function showInstructions() {
  alert(`Instructions:
1. Choose a level to start the game.
2. Write Pascal code in the text area to move the player to the diamond.
3. Click the Execute button to run your code.
4. Complete all levels to finish the game.`);
}

function resetProgress() {
```

Рисунок 8 – Код запуска уровней

Работа над функциональным тестированием завершена программа отображает запускает выбранный уровень, загружает данные уровня, отображает соответствующий лабиринт и главное меню игры, выводит инструкции для игрока, сбрасывает прогресс игрока.

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе была спроектирована и разработана обучающая web-игра в стиле «Герой программист» для детей 12 лет.

Для достижения поставленной цели были решены следующие задачи:

- на основе анализа учебной, научно-технической литературы и сформирована концепция к разрабатываемой обучающей игре;
- рассмотрены этапы планирования и разработки компьютерной игры в стиле «Герой-программист»;
- разработана образовательная браузерная игра в стиле «Герой-программист» для обучения программированию.

В заключение можно отметить, что разработка обучающих игр в контексте обучения программированию представляет собой актуальную и перспективную задачу. В условиях стремительного развития информационных технологий программирование становится необходимым навыком, который открывает широкие возможности для самореализации и профессионального роста. Традиционные методы обучения часто оказываются недостаточно эффективными для привлечения и удержания интереса учащихся. В этом контексте обучающие игры предоставляют уникальную возможность сделать процесс обучения более увлекательным, интерактивным и мотивирующим.

Использование таких языков программирования, как HTML, CSS и JavaScript, позволяет создать высококачественные браузерные игры. Эти инструменты обеспечивают гибкость и мощь, необходимые для реализации разнообразных игровых механик и интерактивных элементов. Выбор подходящих инструментов разработки, таких как Visual Studio Code и WebStorm, а также использование возможностей Chrome DevTools, способствует повышению производительности и эффективности процесса разработки.

Результаты исследования представлены на следующих научных мероприятиях:

– VII Всероссийской научно-практической конференции «Актуальные проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 01-02 ноября 2023 г., участие).

– II Всероссийский молодежный научный форум ««Современное педагогическое образование: теоретический и прикладной аспекты»» (г. Лесосибирск, ЛПИ – филиал СФУ 10-15 апреля 2023 г., участие).

Принята к публикации статья: Хабибулин, Д.С. Проектирование и разработка игры в стиле «Герой-программист» для обучения программированию / Д.С. Хабибулин // Всероссийская научно-практическая конференция «Современное педагогическое образование: теоретический и прикладной аспекты» – Лесосибирск, 2024 – 3 стр.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Андерсон, С. Приманка для пользователей. Создаем привлекательный сайт / С. Андерсон. – Москва : Питер, 2013. – 793 с. – ISBN: 978-5-496-00214-1
2. 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)
3. Архангельская, М.Д. Бизнес этикет, или игра по правилам / М. Д. Архангельская. – Москва : Эксмо, 2007. – 160 с. – ISBN: 5-699-05011-6
4. Арстанова, Л. Г. Занятия и развлечения со старшими дошкольниками. Разработки занятий, бесед, игр и развлечений на нравственные темы / Л. Г. Арстанова. – Москва : Учитель, 2020. – 324 с. – ISBN: 978-5-7057-1620-3
5. Бартон, Д. Р. Биржевые стратегии. Игры без риска / Д. Р. Бартон, Сьюггеруд. – Москва : Санкт-Петербург: 2007. – 400 с. – ISBN: 5-469-00646-8
6. Браун, Э. Изучаем JavaScript. Руководство по созданию современных веб-сайтов / Э. Браун. – Москва : Альфа-книга, 2020. – 368 с. – ISBN: 978-5-9908463-9-5
7. Вакуленко, Ю. А. Веселая грамматика. Разработки занятий, задания, игры / Ю. А. Вакуленко. – Москва : Учитель, 2017. – 780 с. – ISBN: 978-5-7057-2975-3
8. Винокуров, Д. А. Разработка развивающей мобильной игры на платформе Unity / Д. А. Винокуров // Наука молодых - наука будущего. – 2023. – URL: <https://www.elibrary.ru/item.asp?id=53950874> (дата обращения: 09.11.2023)
9. Губин, М. С. PHP 8. Новинки языка и программы для работы с ним / М. С. Губин, Д. Котеров. – Екатеринбург : Издательские Решения, 2020. – 19 с. – ISBN 9785005138330.

10. Джейсон Ф. Flash-реклама. Разработка микросайтов, рекламных игр и фирменных приложений с помощью Adobe Flash / Ф. Джейсон. – Москва : Рид Групп, 2012. – 945 с. – ISBN: 978-5-4252-0139-3
11. Дакетт Д. Основы веб-программирования с использованием HTML / Джон Дакетт. – Москва : Эксмо, 2020. – 239 с.
12. Дакетт Д. Разработка и дизайн веб-сайтов / Д. Дакетт. – Москва : Эксмо, 2018. – 250 с. – ISBN: 978-5-04-101286-1
13. Дронов, В.А. HTML5, CSS3 и web2.0. Разработка современных Web-сайтов / В.А. Дронов. – Санкт-Петербург : БХВ-Петербург, 2011. – 416 с. – ISBN: 978-5-9775-0596-3
14. Ешану, А. А. Отличия веб-приложения от веб-сайта / А. А. Ешану // Современное программирование. – Нижневартовск : НВГУ, 2020. – С. 41 – 43.
15. Журавлев, А. Е. Корпоративные информационные системы. Администрирование сетевого домена. Учебное пособие для СПО / А. Е. Журавлев, А. В. Макшанов, Л. Н. Тындыкарь. – Санкт-Петербург : Лань, 2021. – 172 с. – ISBN: 978-5-8114-5517-1
16. Ильин, И. В. Базы данных : учебное пособие / И. В. Ильин, О. Ю. Ильяшенко. – Санкт-Петербург : СПбГПУ, 2020. – 96 с. – ISBN: 978-5-7244-7101-7
17. Карпов, Александр. Создание и продвижение сайтов. Непрофессионал для Непрофессионала / Александр Карпов, Джи Ким. – Москва : , 2022. – 280 с. – ISBN: 978-5-6047080-7-1
18. Кириченко, А. В. Web на практике. CSS, HTML, JavaScript, MySQL, PHP для fullstack-разработчиков / А. В. Кириченко, А. П. Никольский, Е. В. Дубовик. – Санкт-Петербург : Наука и техника, 2021. – 432 с. – ISBN 978-5-94387-271-6.
19. Климов, А. JavaScript на примерах / А. Климов. – Санкт-Петербург: БХВ-Петербург, 2018. – 336 с. – ISBN: 978-5-9775-0361-7
20. Котеров, Дмитрий. PHP 8 / Дмитрий Котеров, Игорь Симдянов. – Москва : БХВ, 2023. – 992 с. – ISBN 978-5-9775-1692-1.

21. Коэн, Исси. Полный справочник по HTML, CSS и JavaScript / Исси Коэн. – Паблшерз : Эксмо, 2007. – 246 с. – ISBN: 978-5-9790-0009-1
22. Казмерчук, С. В. Геймификация как эффективное маркетинговое средство привлечения и удержания клиентов / С. В. Казмерчук // Лидерство и менеджмент. – 2018. – С. 99–104.
23. Касихин, В.В. Как стать создателем компьютерных игр. Краткое руководство / В.В. Касихин. – Москва : Вильямс, 2006. – 208 с.
24. Лоре, А. Проектирование веб-API / А. Лоре. – Москва : ДМК-Пресс, 2020. – 440 с. – ISBN 978-5-97060-861-6.
25. Любанова, Т. П. Бизнес-план: опыт, проблемы. Содержание бизнес-плана, пример разработки / Т. П. Любанова, Л. В. Мясоедова Т. А. Грамотенко, и др.. – Москва : Приор, 2018. – 204 с. – ISBN: 978-5-241-00861-9
26. Мазелис А. Л. Геймификация в электронном обучении // Территория новых возможностей. Вестник Владивостокского государственного университета экономики и сервиса. 2013. № 3 (21). С. 139–142
27. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript / Р. Никсон. – Санкт–Петербург : Питер, 2010. – 768 с. – ISBN: 978-5-4461-1970-7
28. Паласиос, Х. Unity 5.x. Программирование искусственного интеллекта в играх: пер. с англ. Р. Н. Рагимова. – Москва : ДМК Пресс, 2017. – 272 с. – ISBN: 978-5-97060-436-6
29. 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:02.11.2023).
30. Паласиос, Хорхе Unity 5.x. Программирование искусственного интеллекта в играх / Хорхе Паласиос. – Москва : ДМК Пресс, 2017. – 849 с. – ISBN: 978-5-97060-436-6

31. Прохоренок, Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера, 5 изд / Н.А. Прохоренок. – Санкт – Петербург.: БХВ-Петербург, 2019. – 912 с. – ISBN: 978-5-9775-3986-9
32. Пьюривал, Сэмми. Основы разработки веб-приложений / Сэмми Пьюривал. – Москва : Питер, 2015. – 272 с. – ISBN: 978-5-496-01226-3
33. Платов, В. Я. Деловые игры. Разработка, организация, проведение. Учебник / В. Я. Платов. – Москва : Профиздат, 2018. – 192 с. – ISBN 5-255-00129-5
34. Роббинс, Д. Н. Веб-дизайн для начинающих. HTML, CSS, JavaScript и веб-графика / Д. Н. Роббинс. – Санкт-Петербург : BHV, 2021. – 956 с. – ISBN: 978-5-9775-4050-6
35. Робсон, Элизабет. Изучаем HTML, XHTML и CSS / Элизабет Робсон, Эрик Фримен. – Питер, 2019. – 720 с. – ISBN: 978-5-4461-1247-0
36. Рябова, Т. В. Воспитание доброжелательности у детей 6–7 лет в процессе социо-игры / Т. В. Рябова // Молодежная наука как фактор и ресурс инновационного развития : сборник статей II Международной научно-практической конференции. – 2023. – С. 99–110. – URL: <https://www.elibrary.ru/item.asp?id=50106522%20> (дата обращения: 12.12.2023).
37. Сафронов, М. Разработка веб-приложений с помощью Yii 2 и PHP / М. Сафронов. – Москва : ДМК Пресс, 2019. – 377 с. – ISBN: 978-5-97060-252-2
38. Таран, В. А. Играть на бирже просто?! / В. А. Таран. – Москва : Санкт-Петербург: 2007. – 272 с. – ISBN: 5-469-00898-3
39. Торн, А. Основы анимации в Unity: учебное пособие/ А. Торн – ред.: Д. Мовчан, переводчик: Р. Рагимов. – Москва: ДМК, 2016 – 176с. – ISBN: 978-5-97060-377-2
40. Фримен, Э. Изучаем программирование на JavaScript / Э. Фримен, Э. Робсон. – Москва : Эксмо, 2007. – 2018 с. – ISBN: 978-5-4461-0893-0

ПРИЛОЖЕНИЕ А

Листинг HTML кода отвечающий за отображения лабиринта

```
import random

# Определение размера лабиринта
WIDTH, HEIGHT = 10, 10

# Символы для отображения
PLAYER = 'P'
DIAMOND = 'D'
WALL = '#'
EMPTY = ''

# Функция создания лабиринта с некоторыми стенами
def create_maze(width, height):
    maze = [[EMPTY for _ in range(width)] for _ in range(height)]
    # Добавим некоторые стены для примера
    for i in range(width):
        maze[0][i] = WALL
        maze[height-1][i] = WALL
    for i in range(height):
        maze[i][0] = WALL
        maze[i][width-1] = WALL
    # Несколько случайных стен внутри лабиринта
    for _ in range(20):
        x = random.randint(1, width-2)
        y = random.randint(1, height-2)
        maze[y][x] = WALL
    return maze

# Функция печати лабиринта
def print_maze(maze, player_pos, diamond_pos):
    for y in range(len(maze)):
```

```

for x in range(len(maze[y])):
    if (x, y) == player_pos:
        print(PLAYER, end="")
    elif (x, y) == diamond_pos:
        print(DIAMOND, end="")
    else:
        print(maze[y][x], end="")
    print()

# Функция перемещения персонажа
def move_player(player_pos, direction, maze):
    x, y = player_pos
    if direction == 'up' and y > 0 and maze[y-1][x] != WALL:
        y -= 1
    elif direction == 'down' and y < HEIGHT - 1 and maze[y+1][x] != WALL:
        y += 1
    elif direction == 'left' and x > 0 and maze[y][x-1] != WALL:
        x -= 1
    elif direction == 'right' and x < WIDTH - 1 and maze[y][x+1] != WALL:
        x += 1
    return (x, y)

# Инициализация лабиринта и позиций
maze = create_maze(WIDTH, HEIGHT)
player_pos = (1, 1)
diamond_pos = (random.randint(1, WIDTH-2), random.randint(1, HEIGHT-2))

# Проверка, чтобы алмаз не появился на стене
while maze[diamond_pos[1]][diamond_pos[0]] == WALL:
    diamond_pos = (random.randint(1, WIDTH-2), random.randint(1, HEIGHT-2))

# Основной цикл игры
while player_pos != diamond_pos:
    print_maze(maze, player_pos, diamond_pos)

```



```
command = input("Введите команду (up, down, left, right): ")  
player_pos = move_player(player_pos, command, maze)  
print("Поздравляем! Вы нашли алмаз!")
```

ПРИЛОЖЕНИЕ Б

Листинг javascript кода для отображения поля ввода команд

```
const WIDTH = 10;
const HEIGHT = 10;
const PLAYER = 'P';
const DIAMOND = 'D';
const WALL = '#';
const EMPTY = ' ';
let playerPos = { x: 1, y: 1 };
let diamondPos = { x: Math.floor(Math.random() * (WIDTH - 2)) + 1, y:
Math.floor(Math.random() * (HEIGHT - 2)) + 1 };
while (diamondPos.x === 1 && diamondPos.y === 1) {
    diamondPos = { x: Math.floor(Math.random() * (WIDTH - 2)) + 1, y:
Math.floor(Math.random() * (HEIGHT - 2)) + 1 };
}
const maze = createMaze(WIDTH, HEIGHT);
const gameContainer = document.getElementById('game-container');

function createMaze(width, height) {
    const maze = Array.from({ length: height }, () => Array(width).fill(EMPTY));
    for (let i = 0; i < width; i++) {
        maze[0][i] = WALL;
        maze[height - 1][i] = WALL;
    }
    for (let i = 0; i < height; i++) {
        maze[i][0] = WALL;
        maze[i][width - 1] = WALL;
    }
}
```

```

for (let i = 0; i < 20; i++) {
  const x = Math.floor(Math.random() * (width - 2)) + 1;
  const y = Math.floor(Math.random() * (height - 2)) + 1;
  maze[y][x] = WALL;
}
return maze;
}

function renderMaze() {
  gameContainer.innerHTML = "";
  for (let y = 0; y < HEIGHT; y++) {
    for (let x = 0; x < WIDTH; x++) {
      const cell = document.createElement('div');
      cell.classList.add('cell');
      if (x === playerPos.x && y === playerPos.y) {
        cell.classList.add('player');
        cell.textContent = PLAYER;
      } else if (x === diamondPos.x && y === diamondPos.y) {
        cell.classList.add('diamond');
        cell.textContent = DIAMOND;
      } else if (maze[y][x] === WALL) {
        cell.classList.add('wall');
      }
      gameContainer.appendChild(cell);
    }
  }
}

function movePlayer(direction) {
  let newX = playerPos.x;
  let newY = playerPos.y;

```

```

    if (direction === 'up' && playerPos.y > 0 && maze[playerPos.y - 1][playerPos.x]
    !== WALL) {
        newY -= 1;
    } else if (direction === 'down' && playerPos.y < HEIGHT - 1 && maze[playerPos.y
    + 1][playerPos.x] !== WALL) {
        newY += 1;
    } else if (direction === 'left' && playerPos.x > 0 && maze[playerPos.y][playerPos.x
    - 1] !== WALL) {
        newX -= 1;
    } else if (direction === 'right' && playerPos.x < WIDTH - 1 &&
    maze[playerPos.y][playerPos.x + 1] !== WALL) {
        newX += 1;
    }
    playerPos = { x: newX, y: newY };
    renderMaze();
    checkWin();
}
function checkWin() {
    if (playerPos.x === diamondPos.x && playerPos.y === diamondPos.y) {
        setTimeout(() => alert("Congratulations! You found the diamond!"), 100);
    }
}
document.addEventListener('keydown', (event) => {
    if (event.key === 'ArrowUp') {
        movePlayer('up');
    } else if (event.key === 'ArrowDown') {
        movePlayer('down');
    } else if (event.key === 'ArrowLeft') {
        movePlayer('left');
    } else if (event.key === 'ArrowRight') {

```

```
        movePlayer('right');  
    }  
});  
renderMaze();
```

ПРИЛОЖЕНИЕ В

Листинг HTML кода для связи лабиринта с полем ввода

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Labyrinth Game</title>
<style>
  body {
    font-family: Arial, sans-serif;
    display: flex;
    flex-direction: column;
    align-items: center;
    height: 100vh;
    margin: 0;
  }
  #game-container {
    display: grid;
    grid-template-rows: repeat(10, 40px);
    grid-template-columns: repeat(10, 40px);
    gap: 2px;
    margin-bottom: 20px;
  }
  .cell {
    width: 40px;
    height: 40px;
    display: flex;
    justify-content: center;
```

```
    align-items: center;
    border: 1px solid #ccc;
    font-size: 24px;
}
.wall {
    background-color: #333;
}
.player {
    background-color: #4CAF50;
    color: white;
}
.diamond {
    background-color: #FFD700;
    color: white;
}
#command-input {
    width: 300px;
    height: 40px;
    font-size: 18px;
    padding: 5px;
    margin-bottom: 10px;
}
#execute-button {
    width: 150px;
    height: 40px;
    font-size: 18px;
}
</style>
</head>
<body>
```

```
<div id="game-container"></div>
  <input type="text" id="command-input" placeholder="Enter commands (e.g., up,
down)">
  <button id="execute-button">Execute</button>
  <script src="game.js"></script>
</body>
</html>
```


ПРИЛОЖЕНИЕ Г

Скрипт javascript для связи уровней лабиринта

```
const WIDTH = 10;
const HEIGHT = 10;
const PLAYER = 'P';
const DIAMOND = 'D';
const WALL = '#';
const EMPTY = ' ';
let playerPos = { x: 1, y: 1 };
let diamondPos = { x: Math.floor(Math.random() * (WIDTH - 2)) + 1, y:
Math.floor(Math.random() * (HEIGHT - 2)) + 1 };
while (diamondPos.x === 1 && diamondPos.y === 1) {
    diamondPos = { x: Math.floor(Math.random() * (WIDTH - 2)) + 1, y:
Math.floor(Math.random() * (HEIGHT - 2)) + 1 };
}
const maze = createMaze(WIDTH, HEIGHT);
const gameContainer = document.getElementById('game-container');
function createMaze(width, height) {
    const maze = Array.from({ length: height }, () => Array(width).fill(EMPTY));
    for (let i = 0; i < width; i++) {
        maze[0][i] = WALL;
        maze[height - 1][i] = WALL;
    }
}
function renderMaze() {
    gameContainer.innerHTML = "";
    for (let y = 0; y < HEIGHT; y++) {
        for (let x = 0; x < WIDTH; x++) {
            const cell = document.createElement('div');
            cell.classList.add('cell');
            if (x === playerPos.x && y === playerPos.y) {
```

```

        cell.classList.add('player');
        cell.textContent = PLAYER;
    } else if (x === diamondPos.x && y === diamondPos.y) {
        cell.classList.add('diamond');
        cell.textContent = DIAMOND;
    } else if (maze[y][x] === WALL) {
        cell.classList.add('wall');
    }
}
for (let i = 0; i < height; i++) {
    maze[i][0] = WALL;
    maze[i][width - 1] = WALL;
}
for (let i = 0; i < 20; i++) {
    const x = Math.floor(Math.random() * (width - 2)) + 1;
    const y = Math.floor(Math.random() * (height - 2)) + 1;
    maze[y][x] = WALL;
}
return maze;
}
function renderMaze() {
    gameContainer.innerHTML = "";
    for (let y = 0; y < HEIGHT; y++) {
        for (let x = 0; x < WIDTH; x++) {
            const cell = document.createElement('div');
            cell.classList.add('cell');
            if (x === playerPos.x && y === playerPos.y) {
                cell.classList.add('player');
                cell.textContent = PLAYER;
            } else if (x === diamondPos.x && y === diamondPos.y) {

```

```

        cell.classList.add('diamond');
        cell.textContent = DIAMOND;
    } else if (maze[y][x] === WALL) {
        cell.classList.add('wall');
    }
    gameContainer.appendChild(cell);
}
}
}
function movePlayer(direction) {
    let newX = playerPos.x;
    let newY = playerPos.y;
    if (direction === 'up' && playerPos.y > 0 && maze[playerPos.y - 1][playerPos.x]
    !== WALL) {
        newY -= 1;
    } else if (direction === 'down' && playerPos.y < HEIGHT - 1 && maze[playerPos.y
    + 1][playerPos.x] !== WALL) {
        newY += 1;
    } else if (direction === 'left' && playerPos.x > 0 && maze[playerPos.y][playerPos.x
    - 1] !== WALL) {
        newX -= 1;
    } else if (direction === 'right' && playerPos.x < WIDTH - 1 &&
    maze[playerPos.y][playerPos.x + 1] !== WALL) {
        newX += 1;
    }
    playerPos = { x: newX, y: newY };
    renderMaze();
    checkWin();
}

```

```

function checkWin() {
  if (playerPos.x === diamondPos.x && playerPos.y === diamondPos.y) {
    setTimeout(() => alert("Congratulations! You found the diamond!"), 100);
  }
}

function executeCommands(commands) {
  const commandArray = commands.split(',').map(cmd => cmd.trim());
  for (const command of commandArray) {
    movePlayer(command);
  }
}

document.getElementById('execute-button').addEventListener('click', () => {
  const commands = document.getElementById('command-input').value;
  executeCommands(commands);
});

renderMaze();

for i in range(width):
  maze[0][i] = WALL
  maze[height-1][i] = WALL
for i in range(height):
  maze[i][0] = WALL
  maze[i][width-1] = WALL

function renderMaze() {
  gameContainer.innerHTML = "";
  for (let y = 0; y < HEIGHT; y++) {
    for (let x = 0; x < WIDTH; x++) {
      const cell = document.createElement('div');
      cell.classList.add('cell');
      if (x === playerPos.x && y === playerPos.y) {
        cell.classList.add('player');

```

```
    cell.textContent = PLAYER;
} else if (x === diamondPos.x && y === diamondPos.y) {
    cell.classList.add('diamond');
    cell.textContent = DIAMOND;
} else if (maze[y][x] === WALL) {
    cell.classList.add('wall');
```