МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по курсовой работе
по дисциплине «Web-технологии»
Тема: Разработка игр на языке JavaScript

Студент гр. 3343 _____ Пименов П.В.

Преподаватель _____ Беляев С.А.

Санкт-Петербург
2025

Цель работы

Основной целью разработки было создание полноценной браузерной игры с использованием технологий веб-разработки и принципов игрового проектирования. В рамках проекта были выполнены следующие задачи: реализация логики и механик платформера, создание системы управления персонажем с проработанной физикой движения и прыжков, разработка искусственного интеллекта для игровых противников, интеграция редактора уровней Tiled для удобного конструирования игровых карт, создание системы подсчёта очков, разработка пользовательского интерфейса и меню, а также реализация таблицы рекордов с сохранением данных в локальном хранилище браузера.

Задание

Необходимо выполнить курсовую работу в соответствии с учебным пособием Беляев С.А. «Разработка игр на языке JavaScript». Учебное пособие можно взять в библиотеке.

Все делают в соответствии с общим заданием в соответствии с учебным пособием на «чистом» JavaScript (ES6). В группах по 1 человеку.

Минимум 2 уровня игры

Реализованы все менеджеры в соответствии с учебным пособием (УП)

Есть таблица рекордов

Есть препятствия

Есть «интеллектуальные» противники и «бонусы»

Используются tiles с редактором Tiled (www.mapeditor.org) в соответствии с УП

Выполнение работы

Архитектура игры

Данный проект представляет собой двухмерный платформер, реализованный с использованием модульной архитектуры. Каждый компонент системы инкапсулирует строго определённую часть логики, что обеспечивает высокую читаемость, простоту поддержки и расширяемость кода. Вся игровая логика разделена на отдельные JavaScript-модули, которые располагаются в директории js. В корне проекта находятся index.html (основная страница), а также вспомогательные файлы конфигурации, ресурсы и скрипты для запуска через Docker и nginx. Графические ресурсы (тайлсеты, спрайты) размещены в папке img, карты уровней — в data, звуковые файлы — в sound, стили — в css. Такая структура позволяет легко добавлять новые ресурсы и уровни без необходимости модифицировать существующий код.

Точка входа main.js отвечает за запуск игры после полной загрузки страницы. В этом файле происходит инициализация центрального управляющего компонента — gameManager.js. Именно gameManager реализует основной игровой цикл, управление состояниями (игра, победа, переход между уровнями, рестарт), обработку событий, статистику и таблицу рекордов. Модуль mapManager.js отвечает за загрузку, парсинг и отрисовку карт, поддерживает работу с несколькими слоями, оптимизирован для отрисовки только видимой части карты (viewport), что критично для производительности при больших уровнях. physicsManager.js реализует простую, но надёжную физическую модель: гравитацию, определение проходимости тайлов, обработку столкновений с платформами и поддержку прыжков с "coyote time". spriteManager.js управляет загрузкой и отрисовкой спрайтов динамических сущностей (игрок, души, враги) по данным из атласа, что позволяет легко добавлять новые анимации и объекты. soundManager.js отвечает за проигрывание фоновой музыки с учётом ограничений браузера, реализует автоматический повтор и обработку событий пользователя для

старта звука. eventManager.js реализует обработку нажатий клавиш, хранит текущее состояние управления и обеспечивает поддержку одновременного нажатия нескольких клавиш. Для каждой игровой сущности (игрок, душа, враг) выделены отдельные классы (player.js, soul.js, enemy.js), что облегчает расширение и поддержку кода, а также позволяет реализовать уникальное поведение для каждого типа объекта.

Описание игровой механики

Игровой процесс построен вокруг управления персонажем, который прыгает по платформам, собирает души и избегает врагов. Управление осуществляется клавишами WASD: A/D — движение влево/вправо, W/S — прокрутка камеры вверх/вниз, пробел — прыжок. Камера реализована как отдельное "окно" (viewport), которое может перемещаться по карте независимо от положения игрока, что позволяет просматривать разные части уровня. Цель каждого уровня — собрать все души, разбросанные по карте, и добраться до специальной финальной зоны. После сбора всех душ и достижения этой зоны запускается таймер победы, по истечении которого происходит переход на следующий уровень или завершение игры.

Враги представлены в виде красных кружков, которые появляются в случайных местах верхней части карты. Каждый враг медленно летит к игроку по кратчайшему пути, игнорируя платформы и другие препятствия. При столкновении с игроком сбоку или снизу враг исчезает, а у игрока вычитается две души. Если же игрок прыгает на врага сверху, враг уничтожается, а игроку начисляется одна душа. Такая механика поощряет активное взаимодействие с врагами и добавляет элемент риска. Количество душ может быть отрицательным, что отражается в статистике и влияет на итоговый результат. После завершения уровня игроку предлагается ввести имя для таблицы

рекордов, которая хранится в localStorage браузера и отображает 10 лучших результатов по количеству собранных душ и времени прохождения.

Система управления игрой

Основной игровой цикл реализован в gameManager.js с помощью setInterval (60 FPS), что обеспечивает плавную анимацию и своевременное обновление состояния всех объектов. gameManager управляет всеми аспектами игры: инициализацией уровней, спавном игрока, душ и врагов, обработкой победы, переходами между уровнями, рестартом, а также ведением статистики и таблицы результатов. Для каждого уровня происходит отдельная загрузка карты, что позволяет легко добавлять новые уровни без изменения основной логики. После победы на уровне происходит плавный переход на следующий, а после финальной победы — отображение формы для ввода имени и таблицы рекордов. Вся статистика (количество душ, время прохождения) сохраняется между уровнями и сбрасывается только при полном рестарте игры, что позволяет отслеживать прогресс игрока на протяжении всей сессии.

Система управления вводом реализована через eventManager.js. В этом модуле реализован биндинг клавиш на действия, хранение состояния нажатий, поддержка одновременного нажатия нескольких клавиш. Такой подход обеспечивает отзывчивое и плавное управление персонажем и камерой, а также позволяет легко расширять систему управления новыми действиями (например, добавление новых способностей или типов взаимодействия).

Система физики

physicsManager.js реализует простую, но надёжную физическую модель, которая включает гравитацию, определение проходимости тайлов, проверку

"на земле" для прыжков, а также ограничение по краям карты. Для платформ используются специальные id тайлов, которые считаются непроходимыми. Проверка коллизий осуществляется по всем слоям карты, что позволяет реализовать сложные многоуровневые платформы и препятствия. Прыжки реализованы с поддержкой "coyote time" — небольшой задержки, позволяющей прыгнуть даже после схода с платформы, что делает управление более прощающим и удобным для игрока. Скорость падения и бега ограничены, чтобы избежать прохождения сквозь платформы и обеспечить предсказуемое поведение персонажа. Вся физика реализована максимально просто, но с учётом всех необходимых для платформера нюансов.

Система работы с картами

mapManager.js обеспечивает загрузку карт в формате JSON (экспорт из редактора Tiled), загрузку tileset, парсинг слоёв и объектов. Карта может содержать несколько слоёв: каждый слой отрисовывается отдельно, прозрачные тайлы не затирают нижние, что позволяет реализовать сложные многоуровневые сцены. Для оптимизации отрисовывается только видимая часть карты (viewport), что особенно важно для больших уровней. Объектный слой используется для размещения душ, врагов и других сущностей, что позволяет легко изменять расстановку объектов без изменения кода. Все координаты объектов задаются в пикселях, что обеспечивает точное позиционирование независимо от сетки тайлов. Такой подход позволяет быстро создавать и тестировать новые уровни, а также легко интегрировать новые типы объектов.

Игровые сущности

Каждая сущность реализована отдельным классом с чётко определённой ответственностью. Игрок (player.js) поддерживает анимацию бега и стояния,

прыжки с кулдауном, отражение спрайта при движении влево, ограничение по краям карты и плавную гравитацию. Вся логика управления, анимации и взаимодействия с физикой инкапсулирована внутри класса, что облегчает расширение и модификацию поведения. Душа (soul.js) реализует анимацию исчезновения при сборе, плавное затухание и проверку коллизии с игроком. Враг (enemy.js) реализует движение к игроку, обработку столкновений (различие между атакой сверху и сбоку), а также корректный спавн на уровне с учётом уникальности позиций. Все сущности наследуют базовые свойства Entity (позиция, размер, состояние), что обеспечивает единый интерфейс для работы с объектами на карте. Такой подход облегчает добавление новых типов сущностей и расширение функциональности игры.

Пользовательский интерфейс

Интерфейс реализован через HTML-элементы, которые располагаются поверх canvas. Слева сверху отображается статистика (количество собранных душ, время прохождения), справа — таблица рекордов, которая обновляется в реальном времени. После победы появляется форма для ввода имени и кнопка добавления результата, что обеспечивает удобную интеграцию с таблицей рекордов. Вся статистика и таблица результатов оформлены в едином стиле: используется белый шрифт, адаптивное позиционирование, отсутствие лишних декоративных элементов, что делает интерфейс лаконичным и удобным для восприятия. Управление отображается в виде справки под статистикой, что помогает новым игрокам быстро освоиться с управлением.

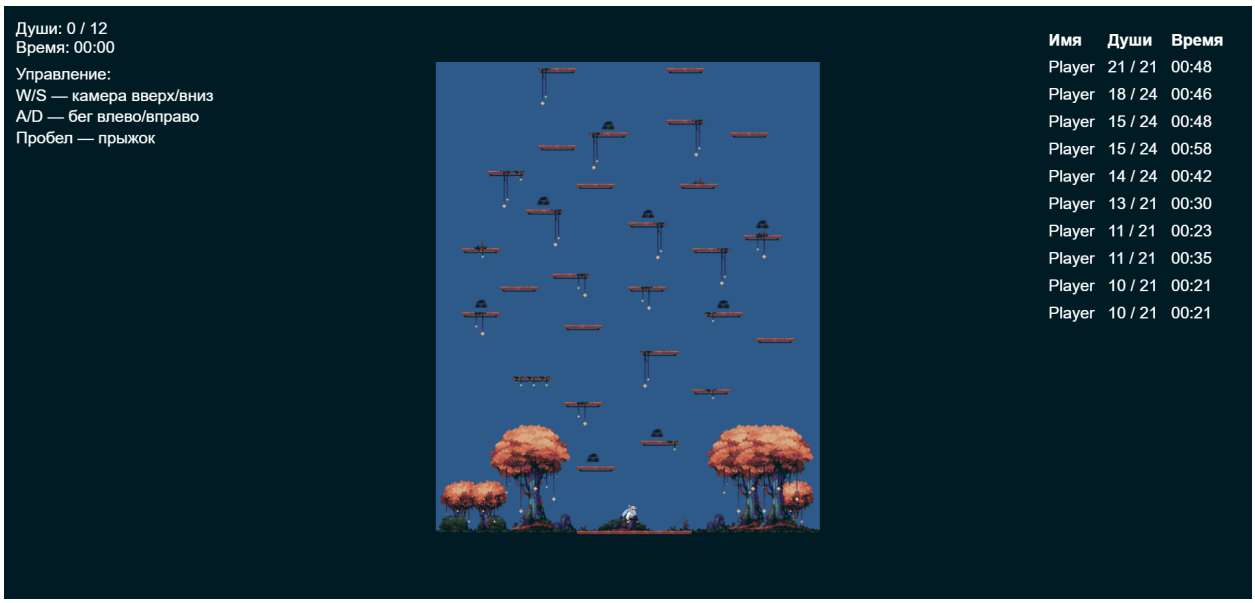

Рисунок 1 – Интерфейс игры

Работа с редактором Tiled

Карты создаются в редакторе Tiled, экспортируются в формате JSON и содержат несколько слоёв: тайловый (платформы, стены) и объектный (души, враги, старт игрока). Размер тайла — 32x32 пикселя, карты могут быть любого размера, что позволяет создавать как компактные, так и очень большие уровни. При загрузке карты mapManager автоматически определяет размеры, слои и объекты, что позволяет легко добавлять новые уровни без изменения кода. Такой подход обеспечивает гибкость и масштабируемость проекта, а также ускоряет процесс разработки и тестирования новых игровых механик.

Особенности реализации

В проекте реализованы: поддержка нескольких уровней, плавный переход между ними, сохранение и отображение таблицы рекордов, анимация исчезновения душ, плавная анимация победы, поддержка отрицательного количества душ, корректная обработка рестарта и сброса статистики, а также подробные комментарии по всему коду. Архитектура легко расширяется: для добавления новых типов объектов или врагов достаточно реализовать новый

класс и зарегистрировать его в фабрике gameManager. Такой подход обеспечивает высокую гибкость, простоту поддержки и возможность дальнейшего развития проекта без необходимости переписывать существующий код.

Вывод

В результате работы создана полнофункциональная браузерная игра, демонстрирующая возможности современных веб-технологий для разработки интерактивных приложений. Проект реализует ключевые элементы платформера — управление персонажем, физику движения, искусственный интеллект врагов, систему уровней и удобный интерфейс. Использование редактора Tiled ускорило проектирование контента и подчеркнуло ценность специализированных инструментов в геймдизайне.

Модульная архитектура и объектно-ориентированный подход с прототипным наследованием JavaScript обеспечивают масштабируемость, поддержку и эффективную организацию кода. Игра полностью работает в браузере, подтверждая потенциал веб-технологий для создания сложных игровых систем.

В ходе проекта получен опыт разработки игровой логики, работы с Canvas API, физикой и коллизиями, созданием AI, интеграцией внешних инструментов, проектированием интерфейса, применением паттернов и оптимизацией производительности. Итоговый продукт может служить основой для дальнейшего развития и демонстрирует практическое применение веб-разработки в создании игр.