**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по курсовой работе**

**по дисциплине «Веб-технологии»**

Тема: Разработка игры на языке Javascript

|  |  |  |
| --- | --- | --- |
| Студент гр. 7383 |  | Зуев Д.В. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2019

**Цель работы**

Закрепление знаний в области современных интернет-технологий; развитие практических навыков разработки современных интернет-приложений с помощью языка разметки HTML, каскадных таблиц стилей CSS и языка программирования Javascript.

**Постановка задачи**

Необходимо разработать игру в соответствии с учебным пособием на «чистом» JavaScript (ES6). Необходимо реализовать минимум 2 уровня игры, а также реализовать все менеджеры, описанные в пособии, и таблицу рекордов.

В учебном пособии приведены подходы к разработке однопользовательской 2D игры на Javascript. Используя приведенный материал может быть разработан универсальный «движок», включающий в себя набор менеджеров и позволяющий реализовывать игры.

**Основные теоретические сведения**

1. Отображение карты игры (MapManager).

Карта отображает 2D игровое поле, в котором разворачивается сюжет игры. В случае игры со множеством уровней, которые могут разрабатываться дизайнером независимо от разработчика, целесообразно использовать соответствующие инструменты.

В качестве бесплатного инструмента, позволяющего создавать карты независимо от программного кода, можно воспользоваться редактором карт Tiled, который позволяет сохранять результаты своей работы в формате JSON.

Редактор Tiled в качестве источника информации использует рисунки, которые разбиваются на прямоугольники заданного размера, так называемые «Наборы тайлов» (блоков). Именно из этих наборов блоков формируется карта. Карта может состоять из нескольких слоев.

Основные свойства JSON формата карты, построенной в Tiled, height и width описывают количество блоков по горизонтали и по вертикали, из которых состоит карта, соответственно tileheight и tilewidth показывают размеры одного блока. Массив layers содержит объекты для каждого слоя, массив tilesets — объекты, описывающие каждый рисунок, из которого строится карта игры.

Массив layers хранит объекты двух типов (поле type): tilelayer и objectgroup. Объекты типа tilelayer описывают слой блоков карты, а объекты типа objectgroup описывают объекты, размещенные на карте. Поля height и width объекта типа tilelayer содержат количество блоков, помещающихся, соответственно, по высоте и по ширине в данном слое, name — его имя, и самое главное поле — массив data, который содержит номера всех блоков, отображаемых на карте. Несмотря на то что массив одномерный, он хранит информацию обо всей карте: известно количество блоков по горизонтали, соответственно, первые width блоков относятся к первой строке блоков на карте, вторые — ко второй и т. д. По номеру блока можно восстановить его координаты и в каком изображении (tilesets) он находится. Объект типа objectgroup в поле objects хранит массив объектов, отображаемых на карте. Каждый объект имеет имя (name), тип (type) и координаты (x, y).

1. Отображение объектов (Entities + SpiteManager).

Отображение объектов на холсте подразумевает описание объекта, определение его местоположения и отображение. В случае размещения уникальных объектов разработчику необходимо каждый из них создать для последующего отображения, но в большинстве игр объект одного и того же типа может встретиться несколько раз, тогда после описания объекта появляется задача его тиражирования и размещения копий в заданных местах. Размещение копий тоже может выполняться по-разному: объекты могут размещаться дизайнером игры при подготовке карты либо создаваться на этапе выполнения игры в ответ на действия пользователя. Следующий шаг — анимация объектов, которая включает отображение рисунков в заданной

последовательности с учетом логики работы игры. Чем сложнее анимация, тем больше рисунков придется использовать для отображения.

Практика показывает, что для игры средней сложности может потребоваться загрузка более 1000 файлов в браузер. Загрузка большого количества мелких файлов требует значительно большего времени, чем загрузка нескольких объемных файлов. Соответственно, можно предложить ускорение процесса загрузки множества изображений, если разместить их в одном файле.

Размещение множества изображений в одном файле позволяет существенно уменьшить время их загрузки в браузер. При этом одновременно с файлом изображения необходимо сформировать описание, в котором будет храниться информация о размещении каждого отдельного изображения. Общее описание при этом принято называть «атлас», а каждое отдельное изображение — «спрайт». Атласы могут быть представлены в самых разных форматах — это текст, CSS, XML, JSON и т.д.

1. Взаимодействие с пользователем (EventsManager).

Взаимодействие с пользователем подразумевает получение от него управляющих воздействий с помощью клавиатуры или мыши, например, для управления объектом игрока, который будет перемещаться по карте и выполнять требуемые действия.

Обработку нажатий на клавиатуру и/или мышь предлагается вынести в отдельный менеджер EventsManager. Тогда разработчик может не только сопоставлять коды клавиш действиям, но и при необходимости изменять их.

1. Реализация логики поведения объектов (PhysicManager).

При создании двухмерной игры возникает вопрос реалистичности создаваемого игрового пространства. Анимация игры подразумевает движение главного героя и объектов игры.

При создании объектов у каждого из них могут быть свои особенности отображения, изменения состояния, влияния на другие объекты, но законы движения с большой вероятностью будут общие. В связи с этим целесообразно общую логику по обновлению вынести в специальный объект – менеджер физики объектов.

1. Менеджер игры (GameManager).

Возникает необходимость объединить все элементы игры в единое целое и запустить исполнение. В качестве инструмента объединения предлагается использовать менеджер игры. Менеджер игры должен обеспечить инициализацию, загрузку всех необходимых ресурсов, хранение и управление всеми объектами игры, регулярное обновление и отображение пользователю игрового мира.

1. Управление звуком (SoundManager).

Для погружения пользователя в игру необходимо использовать не только сюжетную линию и эффектную графику, но и звуковое сопровождение. Для управления звуком в игре целесообразно создать менеджер звука, который загрузит все звуки, будет их хранить и проигрывать по мере необходимости.

**Ход работы**

1. C помощью программы Tiled были составлены два уровня игры, на которых были размещены тайлы и игровые объекты. Затем они были сохранены в формате JSON в папку «public/Maps».
2. Был реализован менеджер для управления картой MapManager, который загружает карту из JSON файла, парсит ее, после чего сохраняет элементы карты в соответствующие свойства класса, с проверкой того, чтобы попытка отобразить карту не была совершена до загрузки всех изображений для каждого набора блоков. Также реализован метод для отображения карты.
3. Были созданы объекты, которые будут задействованы в ходе игры в качестве сущностей (Entities). Все объекты имеют общие свойства, которые вынесены в соответствующий класс-родитель, также у каждого объекта есть методы с одинаковыми названиями, но разными реализациями в зависимости от целей использования объекта. Таким образом реализуется полиморфизм для сущностей.
4. Для взаимодействия с пользователем (нажатие пользователем клавиш на клавиатуре) был написан менеджер событий (ЕventsManager). Данный объект получает от пользователя управляющие воздействия с помощью определенных клавиш клавиатуры для управления объектом игрока.
5. Логика поведения объектов была вынесена в отдельный менеджер физики (PhysicManager). Менеджер физики объектов содержит две функции: основную update для обновления состояния.
6. Для объединения всех менеджеров реализован менеджер игры (GameManager). Данный менеджер обеспечивает инициализацию, загрузку всех необходимых ресурсов, хранение и управление всеми объектами игры, а также регулярное обновление и отображение пользователю игровой карты.
7. Для воспроизведения звуков в процессе игры реализован менеджер звуков (SoundManager). Данный менеджер умеет загружать и хранить звуки и проигрывать их по мере необходимости. Затем эти звуки воспроизводятся в соответствующие моменты игры.
8. Была создана страница с полем ввода имени игрока. Эта страница представлена на рисунке 1.

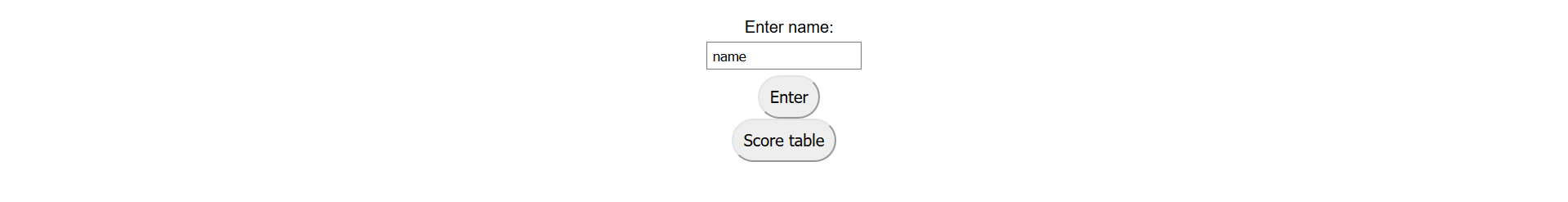


Рисунок 1 – Стартовая страница

1. Была создана страница с игрой. На ней отображается текущий счет и количество очков. Эта страница представлена на рисунке 2.

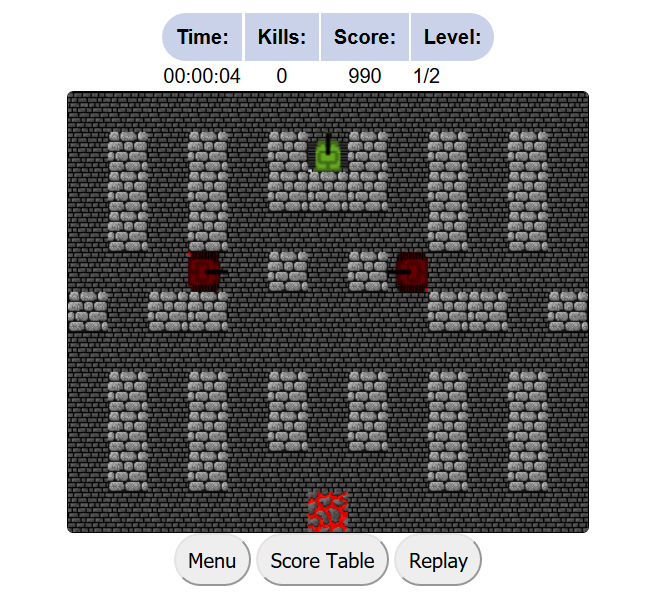


Рисунок 2 – Страница игры

1. Было создано окно для таблицы рекордов. Таблица представлена на рисунке 3.

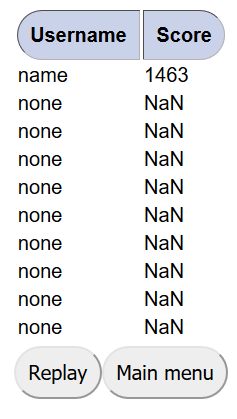


Рисунок 3 – Страница рекордов

**Вывод**

В ходе данной курсовой работы была разработана 2D игра. В ходе разработки были использованы все менеджеры, описанные в пособии, а также реализована таблица рекордов и переход между двумя созданными уровнями.