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

ОТЧЕТ

по курсовой работе

по дисциплине «Web-технологии»

Тема: Разработка игр на языке JavaScript

Студент гр. 1303 _____ Коренев Д.А.

Преподаватель _____ Беляев С.А.

Санкт-Петербург

2023

Постановка задачи.

Необходимо выполнить курсовую работу в соответствии с учебным пособием Беляев С.А. «Разработка игр на языке JavaScript».Основные требования:

1. Минимум 2 уровня игры
2. Реализованы все менеджеры в соответствии с учебным пособием
3. Есть таблица рекордов
4. Есть препятствия
5. Есть «интеллектуальные» противники и «бонусы»
6. Используются tiles с редактором Tiled (www.mapeditor.org) в соответствии с УП.

Описание решения(программы).

Для создания карты использовался редактор карт Tiled, в результате работы которого получался Json файл, содержащий информацию о карте.

Для работы с картой был создан класс MapManager. Функция loadMap(path) посылает асинхронный запрос на чтение Json файла, созданного редактором карт, а при загрузке вызывает функцию parseMap(tilesJSON) с данными. Функция parseMap(tilesJSON) принимает на вход содержимое файла и сохраняет его содержимое в полях, а также отправляет запросы на загрузку изображений, использующихся для отрисовки карты. Функция isVisible(x,y,width,height) определяет, не выходит ли тайл для отрисовки за границы canvas’a. Функция getTileset(tileIndex) по индексу тайла в массиве определяет к какому тайлсету он принадлежит. Функция getTile(tileIndex) по индексу тайла возвращает структуру с изображением, координатами для отрисовки внутри canvas. Функция draw(ctx) рисует в canvas видимую область карты. Функция drawOnPlayer() так же рисует видимую область карты, но только те тайлы, которые рисуются поверх сущностей. Функция parseEntities() в соответствии с расположением объектов в Json создает объекты и засовывает их в массив сущностей GameManager. Функция finish() приводит объект

менеджера к первоначальному виду. Функция isCallision(x, y, obj) принимает координаты и объект, и проверяет, можно ли объекту двигаться в выбранном направлении. Функция centerAt(x,y) центрирует область для отрисовки относительно позиции игрока. Функция getCoord(x, y) принимает координаты и возвращает индексы в двумерном массиве карты, которой эта точка принадлежи

Для рисования спрайтов был создан SpriteManager. Функция loadAtlas (atlasMap) по заданным путям отправляет запрос на загрузку файлов: картинки со спрайтами и Json который содержит информацию об этих спрайтах. Функция parseAtlas(atlasJson) сохраняет в поле класса информацию о каждом спрайте. Функция loadImg(imgName) загружает изображение спрайта в соответсвии его json файлу. Функция drawSprite(ctx,name,x,y) в соответствии с названием спрайта, его позицией рисует спрайт на canvas. Функция getSprite(name) возвращает спрайт по его названию.

Для реализации физики взаимодействия и передвижения объектов создан PhysicsManager. Функции entityBulletAtXY(obj, x, y), entityBonusAtXY(obj, x, y) и entityExitAtXY(obj, x, y) по координатам определяет, столкнулся ли объект, в переданный в параметре с каким либо другим объектом, и если да, то возвращает его. Функция canMoveByY (touchLeftX, touchY, width) принимает координату x левой грани, ширину объекта и координату y (верхней/нижней) грани, и проверяет сможет ли объект переместиться по y. Функция update(obj) пытается передвинуть объект в соответствии с его скоростью, и проверяет столкнулся ли объект с другим объектом, и если да, то вызывает специальный метод у объекта, который коснулся.

Для регистрации нажатий, и сохранения информации о текущем действии создан EventsManager. Функция onKeyDown(event) при нажатии клавиши в объекте actions ставит правду для действия, соответствующего нажатой кнопке в объекте bind. Функция onKeyUp(event) ставит false при отжатии. Функция setup(canvas) устанавливает действия для кнопок и устанавливает обработчики

событий на тело страницы при нажатиии onKeyDown(event) и при отжатии onKeyUp(event) соответственно.

Для воспроизведения звуков был создан SoundManager. Функция init() нужна для инициализации менеджера звуков. Функция load(path, callback) посылает запрос на звуковой файл и при его загрузке вызывает callback функцию. Функция loadArray(array) отправляет запросы для каждого звукового файла из массива звуков и после загрузки всех файлов помечает, что все файлы были загружены. Функция play(path, settings) воспроизводит звуковой файл с заданными настройками(запускать заново и громкость).

Управляющим элементом игры является GameManager. Функция initPlayer(obj) запоминает объект игрока. Функция update() совершает один тик игрового времени, обновляя каждую сущность, удаляет их если необходимо, заново выполняет отрисовку игрового поля и сущностей. Функция draw() рисует на canvas всех сущностей. Функция loadAll(ctx,canvas) выполняет последовательную корректную загрузку для всех менеджеров. Функция play() является входной точкой в запуск игрового процесса, в ней установлено интервал который каждый так вызывает функцию updateWordl. Функция updateWorld() вызывает метод update. Функция setPlayerName(playerName) устанавливает имя игрока. Функция setLevel(level) устанавливает текущий уровень игры. Функция loseLevel() останавливает игру и вызывает gameOver() у main.js. Функция addEntity(obj) добавляет сущность в массив сущностей. Функция removeEntity(obj) удаляет сущность из массива сущностей.

Так же был создан storageManager к которму можно обращаться за данными таблицы лидеров и записывать туда новые данные.

Сущности представляют собой наследников от главного родительского класса Entity. У любой сущности есть позиция, имя, и размеры, также у каждой сущности по умолчанию определены методы: onTouchEntity(obj) вызывается когда сущность сталкивается с другой сущностью, getSpriteName() возвращает имя спрайта соответствующее сущности, isInside(obj) проверяет находится ли объект в сущности хотя бы частично.

Объект Player – помимо стандартных полей у Entity содержит в себе еще массив названий для состояний и отображения, для создания имени нужного спрайта. У игрока переопределена функция onTouchEntity: при прикосновении с Money, Health у игрока прибавляется число денег или здоровье, при взаимодействии с Exit игрок сообщает gameManager о том что уровень пройден. У игрока есть функция changeSide(move_x, move_y) которая определяет состояние и сторону игрока для корректной отрисовки спрайта. У игрока переопределена функция update: помимо обычного обновления обновляется отрисовываемая для спрайта картинка. Функция fire() создает объект пули, т.е. игрок выстреливает в нужном направлении, так же вызывается метод play у soundManager с соответствующим звуком. Функции getCallisionBiasX и getCallisionBiasY возвращают числа на которые смещен центр игрока, так как позиция определяется по верзнему левому углу спрайта.

Объект Enemy аналогичен объекту Player с небольшими корректировками, учитывая особенность сущности. К таким корректировкам относиться метод update() который помимо обновления позиции сущности ищет игрока с помощью метода playerPathSearch(). Этот метод реализует алгоритм A* и возвращает путь от противника до игрока если расстояние между ними не больше 20. Если противник находится на одной прямой с игроком (определяется с помощью метода isStraightLine) то он выстреливает.

У объектов BonusMoney, BonusHealth и Exit изменено имя, чтобы игрок мг определить с каким объектом он провзаимодействовал.

Точкой входа в игру является файл main.js. В нем открываются диалоговые окна: ввод имени пользователя, перезапуск уровня, конец игры. Так же он отвечает за текущий уровень игры и изменяет его при необходимости.

Выводы.

Выполнена курсовая работа в соответствии с учебным пособием Беляев С.А. «Разработка игр на языке JavaScript». Освоен редактор карт Tiled.

СКРИНШОТЫ ПРОГРАММЫ

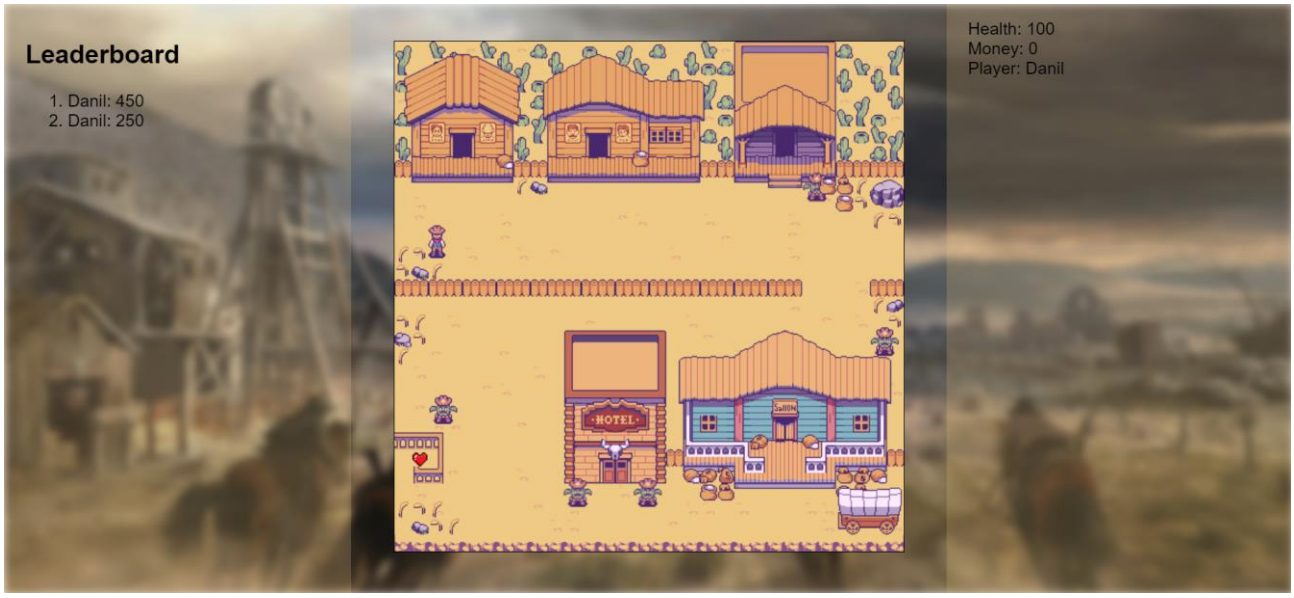

Рисунок 1. – Уровень 1.

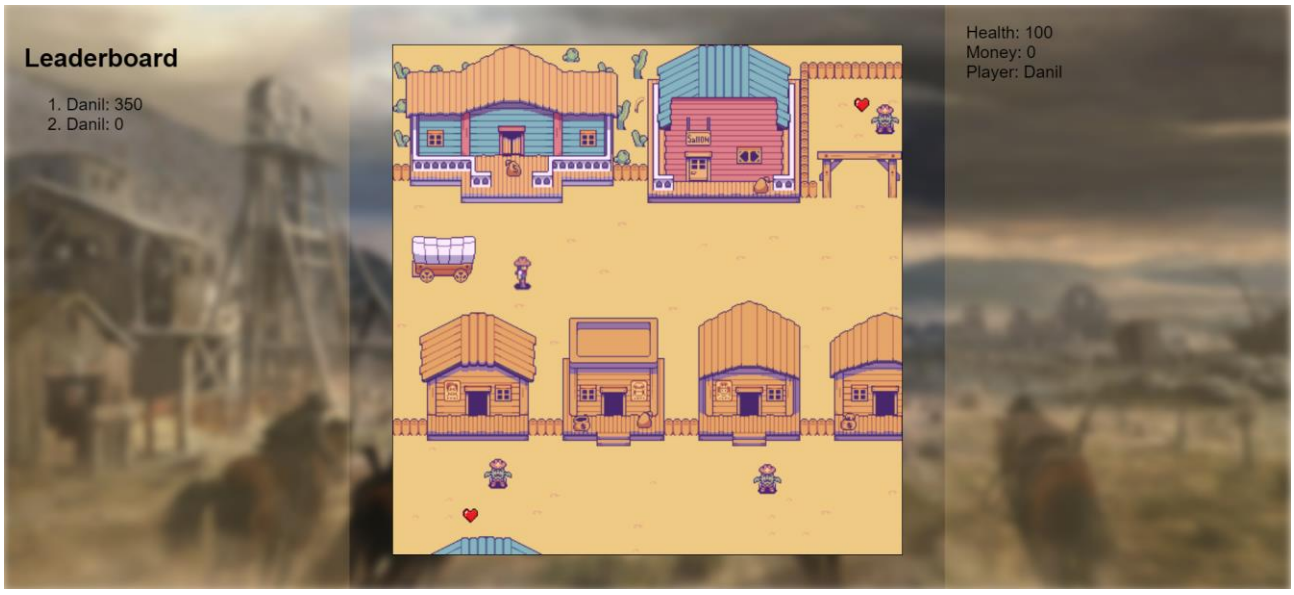

Рисунок 2. – Уровень 2.