**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

курсовая работа

**по дисциплине «Web-технологии»**

Тема: Разработка игры на JavaScript

|  |  |  |
| --- | --- | --- |
| Студентка гр. 9304 |  | Рослова Л.С. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2021

**ЗАДАНИЕ**

**курсовой проект**

|  |  |  |
| --- | --- | --- |
| Студентка Рослова Л.С. | | |
| Группа 9304 | | |
| Тема проекта: Разработка игры на JavaScript | | |
| Исходные данные:  Написать игру на языке программирования JavaScript, содержащую 2 уровня, препятствия, «интеллектуальных» противников, «бонусы» и таблицу рекордов. | | |
| Содержание пояснительной записки:  «Содержание»  «Введение»  «Описание кода программы»  «Внешний вид реализованной игры»  «Заключение»  «Список использованных источников» | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: 06.09.2021 | | |
| Дата сдачи реферата: 20.12.2021 | | |
| Дата защиты реферата: 20.12.2021 | | |
| Студентка | \_\_\_\_\_\_\_\_\_\_\_\_ | Рослова Л.С. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_ | Беляев С.А. |

**Аннотация**

В данной курсовой работе была написана игра на языке программирования JavaScript. Цель игры – победить всех врагов-ниндзя и собрать необходимое количество очков для прохождения на следующий уровень. Для создания карты использовался редактор Tiled, для корректного отображения объектов и элементов карты проводилась работа со спрайтами.

**Summary**

In this course work, a game was written in the JavaScript programming language. The goal of the game is to defeat all the anti-ninjas and collect the required number of points to pass to the next level. To create the map, the Tiled editor was used; for the correct display of objects and map elements, work with sprites was carried out.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | Задача | 6 |
| 2. | Описание кода программы | 7 |
| 2.1. | Менеджер управления картой | 7 |
| 2.2. | Менеджер управления спрайтами | 8 |
| 2.3. | Менеджер управления игрой | 8 |
| 2.4. | Менеджер управления физикой | 9 |
| 2.5. | Менеджер управления звуком | 9 |
| 2.6 | Менеджер управления событиями | 10 |
| 2.7 | Классы объектов игры | 11 |
| 2.8 | Обработка таблицы рекордов | 11 |
| 2.9 | Обработка страницы регистрации | 12 |
| 3. | Внешний вид реализованной игры | 13 |
| 4. | Заключение | 15 |
| 5. | Список использованных источников | 16 |

**введение**

**Цель работы:**

Изучить возможности языка программирования JavaScript и написать на нём игру в соответствии с учебным пособием.

Для достижения поставленной цели требуется решить следующие задачи:

1. Изучить язык программирования JavaScript.
2. Реализовать все менеджеры в соответствии с учебным пособием.

**1. задача**

Написать игру на языке программирования JavaScript.

Основные требования:

1. Минимум 2 уровня игры
2. Реализованы все менеджеры в соответствии с УП
3. Есть таблица рекордов
4. Есть препятствия
5. Есть «интеллектуальные» противники и «бонусы»
6. Используются tiles с редактором Tiled в соответствии с УП
7. **Описание кода программы**

**2.1. Менеджер управления картой**

Менеджер управления картой **mapManager** отвечает за считывание карты из формата JSON с последующим парсингом и отрисовкой.

**mapManager** содержит такие поля, как:

* ***mapData*** – хранит карту
* ***tLayer*** – хранит ссылки на блоки карты
* ***xCount*** – количество блоков по горизонтали
* ***yCount*** – количество блоков по вертикали
* ***tSize*** – размер одного блока
* ***mapSize*** – размер карты в пикселях
* ***tilesets*** – массив описаний блоков карты
* ***imgLoadCount*** – количество загруженных изображений
* ***imgLoaded*** – показывает были ли загружены изображения
* ***jsonLoaded*** – показывает был ли загружен json
* ***view*** – видимая часть карты
* **enemyCount** — счетчик противников на уровне
* **nextLevel** – путь к следующему уровню

Методы:

* ***loadMap(path)*** – принимает путь к файлу с картой и загружает её с использованием AJAX
* ***parseMap(tilesJSON)*** – парсит json файл и инициализирует все поля
* ***draw(ctx)*** – отвечает за отрисовку карты
* ***getTile(tileIndex)*** – принимает индекс блока и возвращает координаты и изображение этого блока
* ***getTileset(tileIndex)*** – возвращает tileset по переданному индексу
* ***isVisible(x,y,width,height)*** – проверяет находится ли переданный блок в видимой части карты
* ***parseEntities()*** – парсинг слоя с объектами и инициализация поля entities gameManager’а
* ***getTilesetIdx(x,y)*** – возвращает блок по переданным координатам
* **centerAt(x,y)** – центрирует положение камеры на карте для видмой области.
* **gameOver()** – приостанавливает обновление карты.

**2.2. Менеджер управления спрайтами**

Менеджер управления спрайтами **spriteManager** отвечает за отрисовку объектов на карте.

**spriteManager** содержит такие поля, как:

* ***image*** – рисунок с объектами
* ***sprites*** – массив объектов
* ***imgLoaded*** – показывает были ли загружены изображения
* ***jsonLoaded*** – показывает был ли загружен json

Методы:

* ***loadAtlas(atlasJSON,atlasImg)*** – принимает путь к файлу и изображение со спрайтами и загружает их с использованием AJAX
* ***loadImg(imgName)*** – загрузка переданного изображения
* ***drawSprite(ctx,name,x,y)*** – отвечает за отрисовку спрайта по переданным имени и координатам
* ***getSprite(name)*** – возвращает спрайт по названию
* ***parseAtlas(atlasJSON)*** – парсит json файл и инициализирует поля: sprites и jsonLoaded

**2.3. Менеджер управления игрой**

Менеджер управления игрой **gameManager** отвечает за логику игры.

**gameManager** содержит такие поля, как:

* ***factory*** – фабрика объектов карте
* ***entities*** – объекты на карте
* ***fireNum*** – идентификатор выстрела
* ***player*** – указатель на объект игрока
* ***laterKill*** – отложенное уничтожение объектов
* ***interval*** – интервал обновления
* ***level –*** уровень

Методы:

* ***initPlayer(obj) –*** инициализация игрока
* ***kill(obj)*** – добавление уничтоженного объекта в массив *laterKill*
* ***loadAll(level)*** *–* производит загрузку карты, а также спрайтов всех объектов, для исправного рисования карты и объектов в режиме реального времени
* ***play()*** – при помощи функции *setInterval* запускается метод ***upDateWorld()***
* ***upDateWorld()*** – вызывает метод ***update()***
* ***update()*** – вызывает у каждого объекта на карте метод обновления, тем самым показывая движение объектов и прочее взаимодействие. Также, в этом методе происходит удаление объектов, которые в процессе игры были уничтожены
* ***draw(ctx)*** – вызывает метод ***draw*** у объектов на карте

**2.4. Менеджер управления физикой**

Менеджер управления физикой **physicManager** используется для того, чтобы реализовать движение объектов.

**physicManager** содержит два метода:

* ***update(obj)*** – вычисляет новые координаты переданного объекта и анализирует, может ли он туда двигаться.
* ***entityAtXY(obj,x,y)*** – по переданным координатам проверяет будет ли столкновение объекта с другим объектом.

**2.5. Менеджер управления звуком**

Менеджер управления звуком **soundManager** выполняет задачу по проигрыванию звука.

**soundManager** содержит такие поля, как:

* ***clips*** – звуковые эффекты
* ***context*** – аудиоконтекст
* ***gainNode*** – главный узел
* ***loaded*** – был ли загружен звук

Методы:

* ***init() –*** инициализация менеджера звука
* ***load(path,callback)*** – загрузка одного аудиофайла
* ***loadArray(array)*** *–* загрузка массива звуков
* ***play(path)*** – проигрывание файла
* ***setVolume(val)*** – изменение громкости звука

**2.6. Менеджер управления событиями**

Менеджер управления событиями **eventsManager** обрабатывает нажатия клавиш на клавиатуре.

**eventsManager** содержит такие поля, как:

* ***bind*** – сопоставление клавиш действиям
* ***action*** – действия

Методы:

* ***setup()*** – устанавливает значения в переменные, а также метод обработки нажатия/отпускания клавиш
* ***onKeyDown(event) –*** отслеживает нажатие на кнопку и заполняет массив действий значением true
* ***onKeyUp(event) –*** отслеживает отпустили ли кнопку и заполняет массив действий значениями false
* **onMouseDown(event)** – отслеживает нажатие на кнопку мыши и заполняет массив действий значением true
* **onMouseUp(event)** – отслеживает отпустили ли кнопку мыши и заполняет массив действий значениями false

**2.7. Классы объектов игры**

В игре есть базовый класс **Entity**, который задает позицию и размеры всем производным классам. От класса **Entity** наследуются два основных класса, **Player** и **Enemy,** которые расширяют функционал и добавляют методы для взаимодействия с картой**/**объектами. Оба класса имеют запас здоровья, скорость движения **speed**, направление движения **move\_x**/**move\_y**, счетчик анимации **AnimationCounter** и флаг стрельбы **flag**.

Все враги движутся случайно, а выстрел или столкновение с препятствием меняет его. Враги и игрок способны порождать другой объект – **Fireball**. Этот класс отвечает за отрисовку и взаимодействие огненных снарядов. Враги производят выстрел в игрока когда тот попадает на линию их стрельбы. После выстрела флаг стрельбы переходит в положение false, а возвращение в состояние true происходит посредством **setTimeout()**, таким образом контролируется скорость стрельбы.

Если количество здоровья какого-либо объекта становится равным нулю, то объект помещается на очередь в массив удаления **laterKill** и удаляется с карты.

**2.8. Обработка таблицы рекордов**

Для работы с таблицей рекордов написаны две функции:

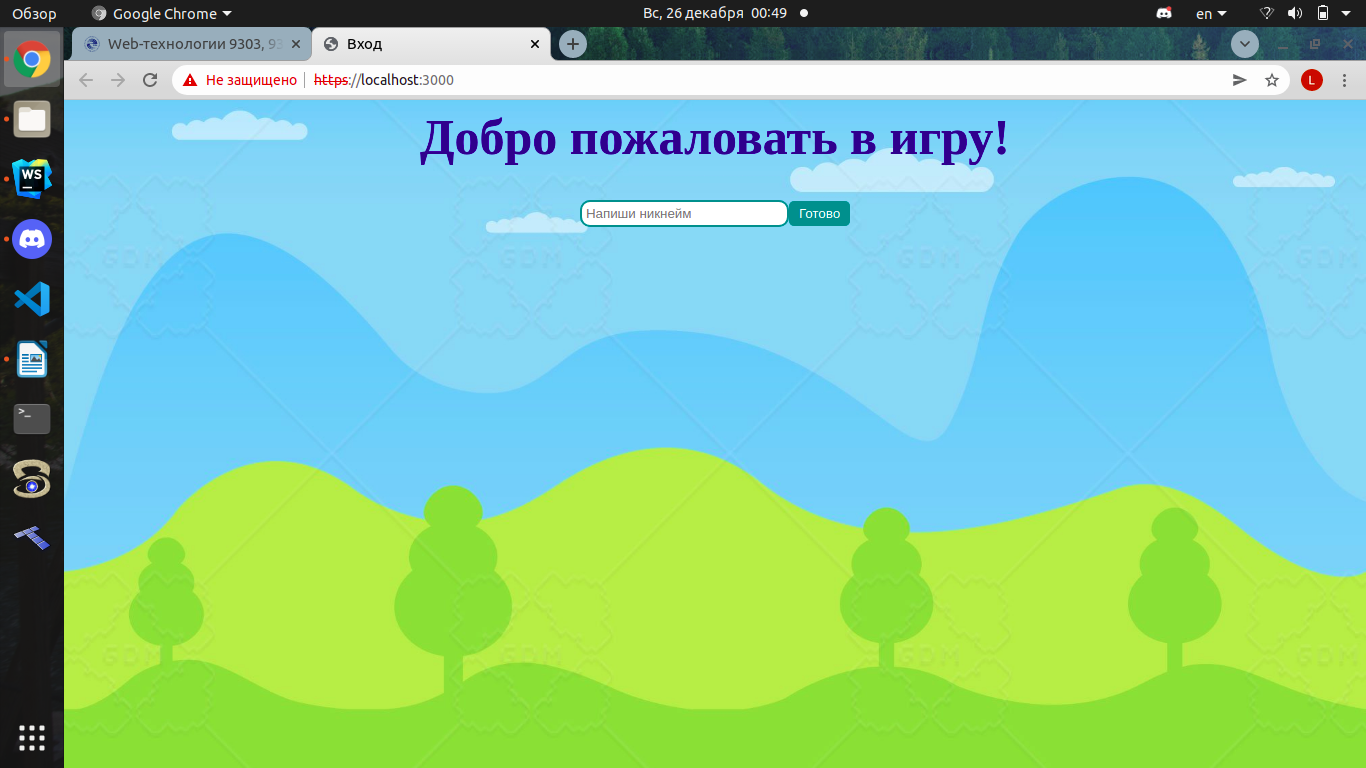
* ***updateRecordTable()*** – обновляет данные о рекордах в *localStorage*
* ***addRecord()*** – добавляет текущий результат в таблицу рекордов.

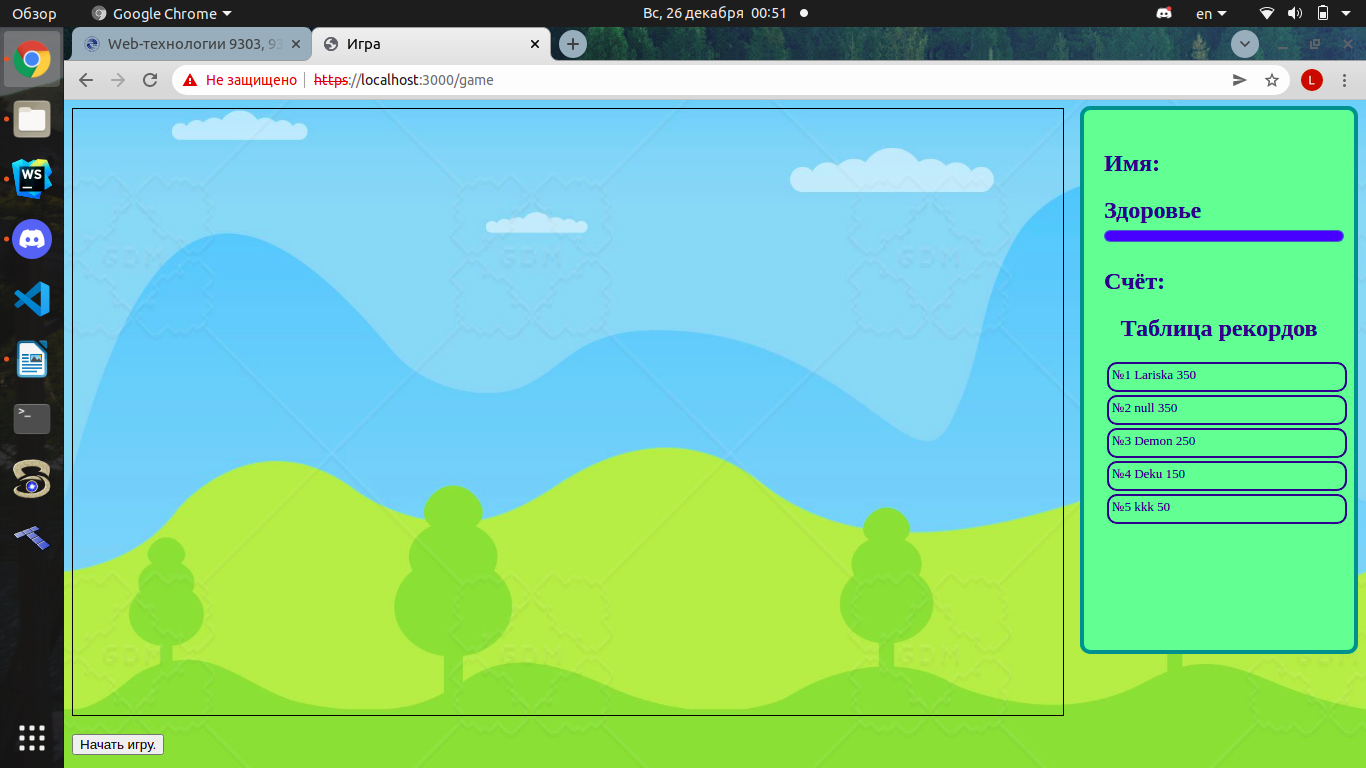
**2.9. Обработка страницы регистрации**

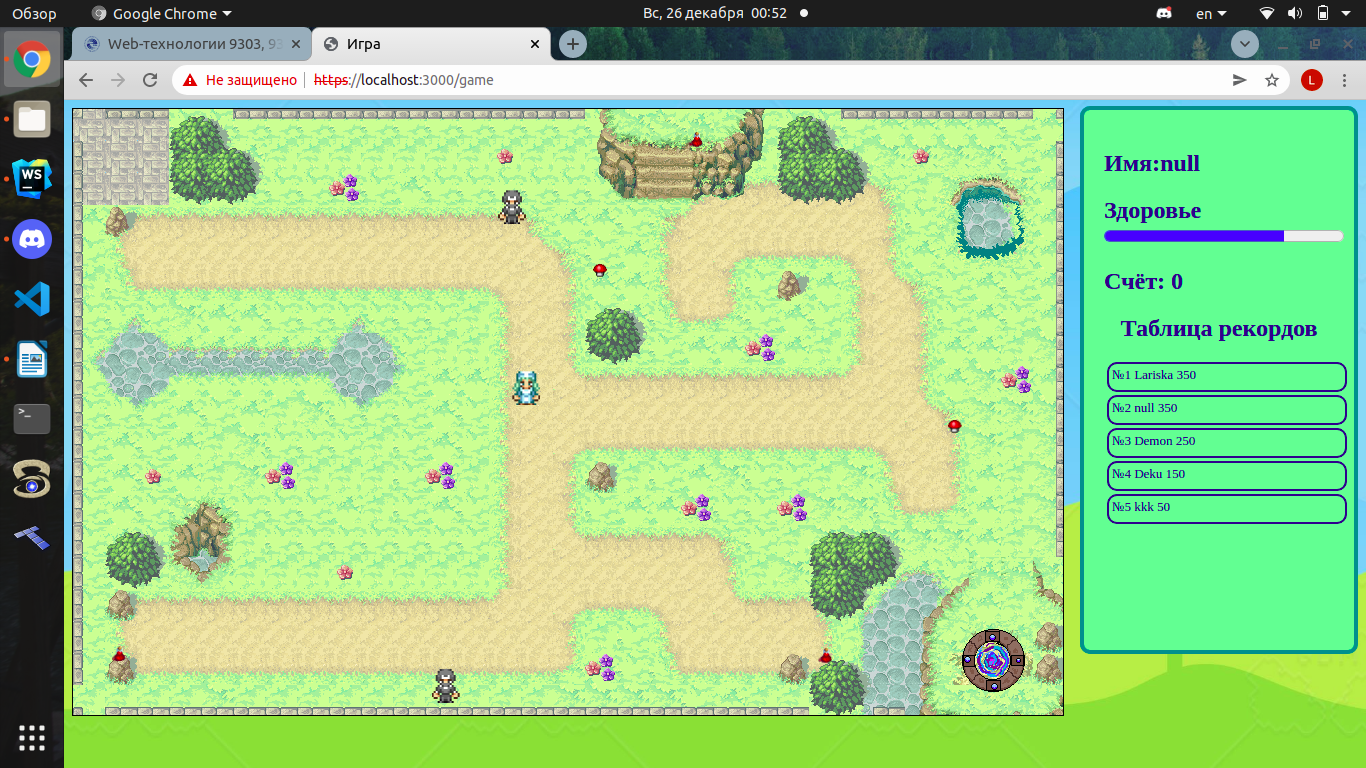
Для работы со страницей регистрации используется одна функция:

* ***check\_name()*** – считывает введённое имя игрока и сохраняет его в *localStorage,* после чего переходит на страницу с игрой.

**3. ВНЕШНИЙ ВИД РЕАЛИЗОВАННОЙ ИГРЫ**

Рисунок 1 – Страница с регистрацией

Рисунок 2 – Страница с игрой и таблицей рекордов

Рисунок 3 – Первый уровень игры

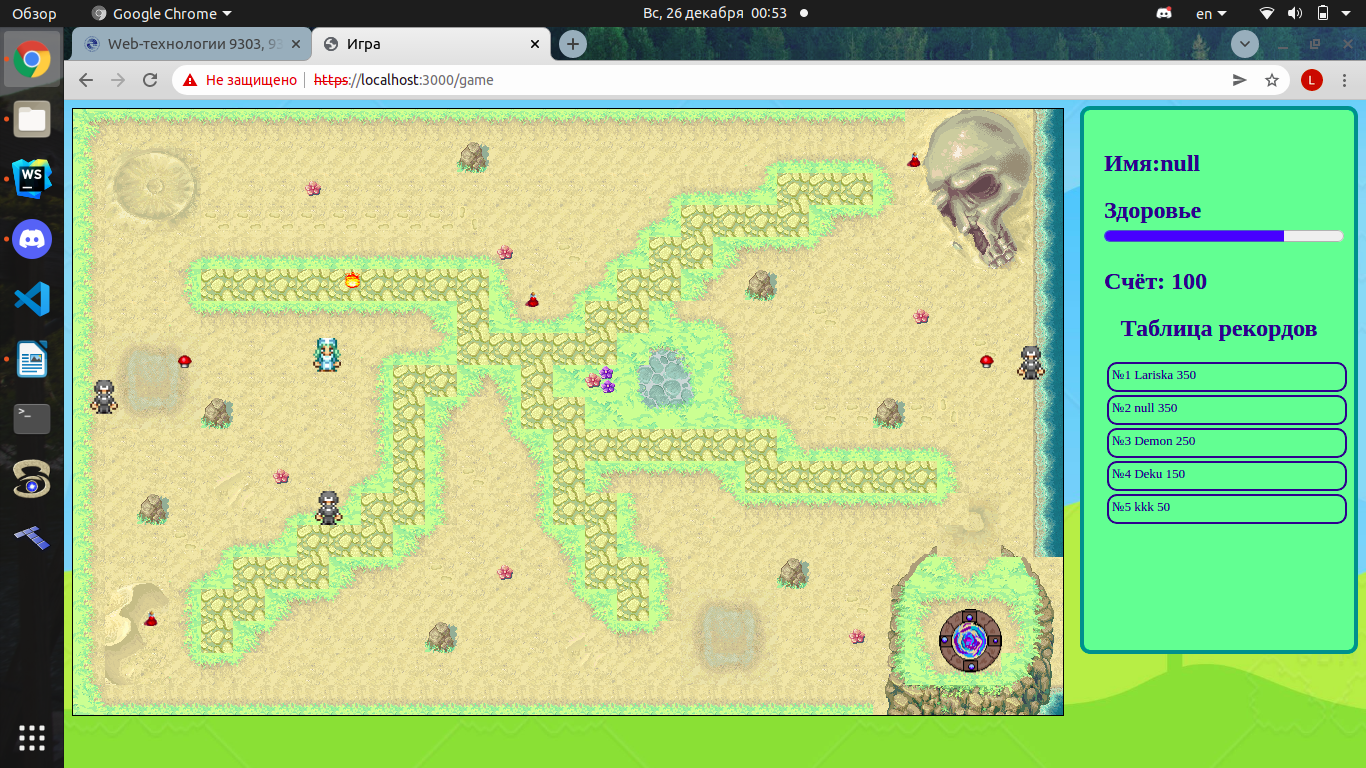
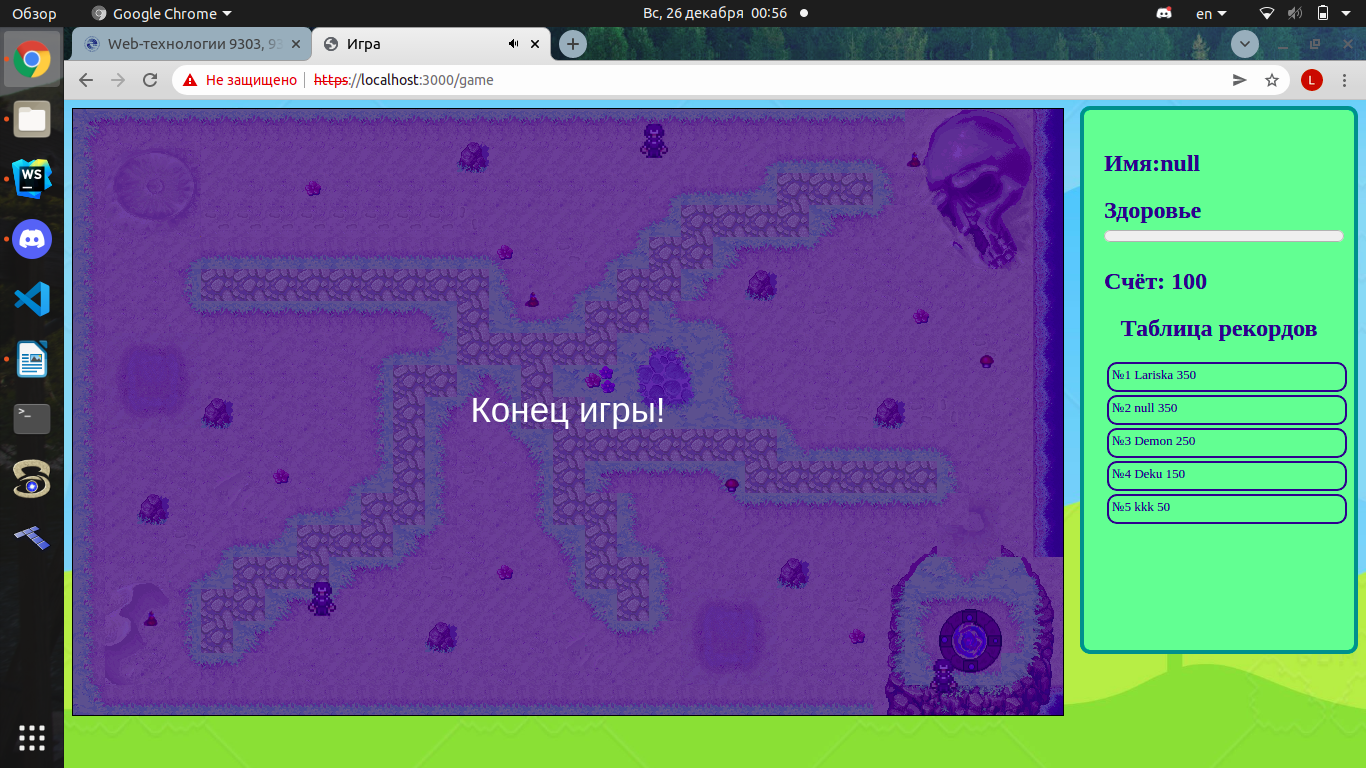
Рисунок 4 – Второй уровень игры



Рисунок 5 – Завершение игры

**заключение**

В ходе работы были изучены возможности JavaScript. Была написана игра в соответствии с учебным пособием.

**список использованных источников**

1. Беляев С.А. Разработка игр на языке JavaScript: Издательство Лань, 2020. 128 с.
2. Справочник по JavaScript // developer.mozilla.org. URL: <https://developer.mozilla.org/ru/docs/Web/JavaScript>