Техническая документация для игры  
 Crazy Metal Adventure.

Составил: Филиппов К.Г.

Введение.

Игра представляет собой локально запускаемый сайт на языке HTML, включает в себя динамические элементы, реализованные с использованием CSS, взаимодействия обрабатываются используя JavaScript.

Игра относится к жанру “Dungeon crawler”, визуально состоит из слоя заднего плана, спрайта персонажа, кнопок взаимодействия и различных текстовых информационных слоев.

Браузерная игра-лабиринт, в которой игрок управляет персонажем, перемещаясь по различным сценам. Игра включает элементы случайности и несколько возможных исходов. Главная цель игры: Перейти в состояние победы найдя выход из лабиринта(Выйти на экран “You won!”)

Основные технические положения.

Изначально задумывалось создать продвинутую систему случайной генерации системы лабиринтов с ограничениями:  
-Не более 20 развилок до состояния победы;

-Не менее 10 развилок до состояния победы;

-Не менее 15% шанса на выход в состояние lose(Экран “You lost”) на каждой развилке;

-Не менее 70% шанса на переход к следующей сцене;  
-Не менее двух проходов на каждой развилке;  
-Не более трех проходов на каждой развилке.

От данной системы было решено отказаться в пользу экономии времени разработки, увеличения скорости работы сайта и понижения системных требований. На замену была разработана более простая система:

каждая развилка является своей сценой. На каждой сцене присутствует три тоннеля. При переходе на эту сцену приводится в действие функция ‘generateRandomChances’(Листинг 1), которая присваивает каждому тоннелю случайные шансы для каждого события.

У каждого из этих тоннелей есть шансы:  
-перевести игрока на следующую сцену(Шанс 88-96%);

-перевести игрока в состояние победы(Шанс 2-6%);

-перевести игрока в состояние проигрыша(Шанс 2-6%).

*Листинг 1*

**function generateRandomChances() {**

**return {**

**winChance: (Math.random() \* 0.01 + 0.05).toFixed(2), // 2-6%**

**loseChance: (Math.random() \* 0.01 + 0.05).toFixed(2), // 2-6%**

**nextChance: (Math.random() \* 0.06 + 0.88).toFixed(2) // 88-96%**

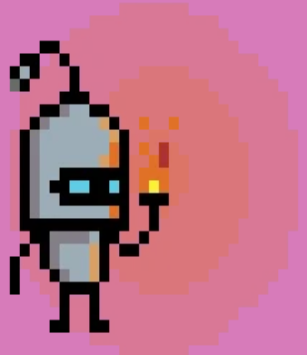
**};**

**}**

Динамические элементы.

Динамические элементы в данном проекте  представлены спрайтом игрока и кнопками взаимодействия.

[Спрайт](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D1%80%D0%B0%D0%B9%D1%82_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0)) игрока представляет собой .WEBP файл, созданный Симоновым Д.А., состоит из анимированного изображения робота и полупрозрачного ореола света от факела в его руке(рис.1).



*Рисунок 1*

Кнопки взаимодействия(Листинг 2) представляют собой элементы, которые позволяют игроку взаимодействовать с игрой. Кнопки позволяют выбирать путь, по которому игрок желает пойти, позволяют начать игру со стартового экрана и перезапустить игру с экранов проигрыша\выигрыша.

Задний фон кнопок меняет насыщенность своего цвета когда на него наводят курсор. Кнопки выбора прохода находятся на координатах проходов(Листинг 3), расположенных на файлах заднего фона. Это возможно, поскольку все изображения и файлы заднего плана были созданы нами и имеют разрешение 1280х800.

После того, как игрок выбрал проход, спрайт персонажа начинает движение в направлении к нажатой кнопке(Первая фаза движения), движения это длится 0.7 секунд. После достижения позиции кнопки, спрайт персонаж замирает у него(Вторая фаза движения) на 0.2 секунды, после чего возвращается на стартовую позицию за 0.2 секунды(Третья фаза движения). Сцена начинает переключаться во время третьей фазы, и когда персонаж оказывается в начальной позиции, задний план уже меняется(Листинг 4).

В последних итерациях был добавлен счетчик пройденных сцен, для упрощения прототипирования и проверки на ошибки между сценами.

*Листинг 1*

**function generateRandomChances() {**

**return {**

**winChance: (Math.random() \* 0.01 + 0.05).toFixed(2), // 2-6%**

**loseChance: (Math.random() \* 0.01 + 0.05).toFixed(2), // 2-6%**

**nextChance: (Math.random() \* 0.06 + 0.88).toFixed(2) // 88-96%**

**};**

**}**

*Листинг 2*

**.path {**

**position: absolute;**

**width: 120px; /\* Ширина кликабельной области \*/**

**height: 60px; /\* Высота кликабельной области \*/**

**background-color: rgba(255, 255, 255, 0.3); /\* Полупрозрачный белый фон \*/**

**border: 2px solid #fff; /\* Белая рамка \*/**

**border-radius: 10px; /\* Закругленные углы \*/**

**cursor: pointer;**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**text-align: center;**

**font-size: 16px; /\* Размер шрифта для надписи \*/**

**color: #fff; /\* Белый текст \*/**

**transition: background-color 0.3s; /\* Плавное изменение фона \*/**

**}**

*Листинг 3*

**// Координаты для туннелей (одинаковые на всех сценах)**

**const tunnelPositions = [**

**{ top: 320, left: 125 }, // Позиция для первого туннеля**

**{ top: 280, left: 575 }, // Позиция для второго туннеля**

**{ top: 320, left: 1030 }  // Позиция для третьего туннеля**

*Листинг 4*

**// Функция для перемещения персонажа к выбранному проходу**

**function movePlayerToPath(pathElement, callback) {**

**const player = document.getElementById('player');**

**const pathRect = pathElement.getBoundingClientRect();**

**const playerRect = player.getBoundingClientRect();**

**const gameRect = document.getElementById('game').getBoundingClientRect();**

**// Вычисляем конечные координаты относительно игрового экрана**

**const targetX = pathRect.left - gameRect.left + pathRect.width / 2 - playerRect.width / 2;**

**const targetY = pathRect.top - gameRect.top + pathRect.height / 2 - playerRect.height / 2;**

**// Вычисляем смещение от текущей позиции**

**const offsetX = targetX - (playerRect.left - gameRect.left);**

**const offsetY = targetY - (playerRect.top - gameRect.top);**

**// Анимация перемещения с использованием translate**

**player.style.transition = 'transform 0.5s';**

**player.style.transform = `translate(${offsetX}px, ${offsetY}px)`;**

**// Вызываем callback после завершения анимации**

**setTimeout(callback, 800); // 0.8 секунды — длительность анимации**

**}**

**// Функция для возвращения персонажа на стартовую позицию**

**function resetPlayerPosition(callback) {**

**const player = document.getElementById('player');**

**player.style.transition = 'transform 0.2s';**

**player.style.transform = 'translate(0, 0)'; // Центр по горизонтали и стартовая позиция снизу**

**// Вызываем callback после завершения анимации**

**setTimeout(callback, 200); // 0.2 секунды — длительность анимации**

**}**

*Листинг 1*

**function generateRandomChances() {**

**return {**

**winChance: (Math.random() \* 0.01 + 0.05).toFixed(2), // 2-6%**

**loseChance: (Math.random() \* 0.01 + 0.05).toFixed(2), // 2-6%**

**nextChance: (Math.random() \* 0.06 + 0.88).toFixed(2) // 88-96%**

**};**

**}**

*Листинг 1*

**function generateRandomChances() {**

**return {**

**winChance: (Math.random() \* 0.01 + 0.05).toFixed(2), // 2-6%**

**loseChance: (Math.random() \* 0.01 + 0.05).toFixed(2), // 2-6%**

**nextChance: (Math.random() \* 0.06 + 0.88).toFixed(2) // 88-96%**

**};**

**}**

ПЛАН:

1. Введение

\*\*Название проекта\*\*: Robot Dungeon

\*\*Описание\*\*: "Robot Dungeon" — это браузерная игра-лабиринт, в которой игрок управляет персонажем, перемещаясь по различным сценам. Игра включает элементы случайности и несколько возможных исходов, таких как победа или поражение.

\*\*Цель\*\*: Создать интерактивную игру, которая будет запускаться локально и предоставлять пользователю увлекательный опыт.

## 2. Структура проекта

### Корневая папка: `RobotDungeon`

- \*\*Файлы\*\*:

- `RobotDungeon.html`: Основной HTML файл, содержащий структуру игры.

- `style.css`: Файл стилей для оформления элементов игры.

- `script.js`: JavaScript файл, содержащий логику игры.

- \*\*Папка `assets`\*\*:

- \*\*Подпапки\*\*:

- `lose`: Изображения и анимации для экрана поражения.

- `win`: Изображения и анимации для экрана победы.

- `backgrounds`: Фоновые изображения для сцен.

- `player`: Спрайты и анимации игрока.

## 3. Функциональные требования

### Стартовый экран

- \*\*Описание\*\*: Первый экран, который видит пользователь при запуске игры.

- \*\*Функциональность\*\*:

- Отображение заднего фона и спрайта игрока.

- Кнопка "Начать игру" для перехода к игровому экрану.

```html

<div id="start-screen">

<img id="start-background" src="assets/Title\_V2.gif" alt="Start Background">

<img id="start-player" src="assets/Bouncing\_Bobbo.gif" alt="Start Player">

<button id="start-button">Начать игру</button>

</div>

```

### Игровой экран

- \*\*Описание\*\*: Основной экран, на котором происходит игровой процесс.

- \*\*Функциональность\*\*:

- Отображение текущей сцены с фоновым изображением.

- Спрайт игрока, который может перемещаться к выбранным проходам.

- Счетчик пройденных сцен.

- Случайные пути с различными исходами (победа, поражение, переход на следующую сцену).

```html

<div id="game" style="display: none;">

<div id="score-counter">Пройдено сцен: 0</div>

<img id="background" src="" alt="Background">

<img id="player" src="assets/player/Bobbo v3.webp" alt="Player">

</div>

```

### Экран победы

- \*\*Описание\*\*: Экран, который показывается при победе игрока.

- \*\*Функциональность\*\*:

- Отображение анимации победы.

- Кнопка "Играть снова" для рестарта игры.

```html

<div id="win-screen" style="display: none;">

<h1>You Won!</h1>

<img id="win-player" src="assets/win/Bobby\_dancing.gif" alt="Bobby\_dancing.gif">

<button id="restart-win-button">Играть снова</button>

</div>

```

### Экран поражения

- \*\*Описание\*\*: Экран, который показывается при поражении игрока.

- \*\*Функциональность\*\*:

- Отображение анимации поражения.

- Кнопка "Играть снова" для рестарта игры.

```html

<div id="lose-screen" style="display: none;">

<h1>Game Over</h1>

<img id="lose-player" src="assets/lose/Bobby\_dead.gif" alt="Bobby\_dead.gif">

<button id="restart-lose-button">Играть снова</button>

</div>

```

## 4. Технические детали

### HTML (`RobotDungeon.html`)

- \*\*Описание\*\*: Содержит структуру стартового, игрового, победного и проигрышного экранов.

- \*\*Подключение ресурсов\*\*:

- Подключает внешние файлы стилей и скриптов.

```html

<link rel="stylesheet" href="style.css">

<script src="script.js"></script>

```

### CSS (`style.css`)

- \*\*Описание\*\*: Определяет стили для всех экранов и элементов игры.

- \*\*Ключевые стили\*\*:

- Стили для экранов, кнопок, счетчика и интерактивных элементов.

- Анимации и эффекты наведения.

```css

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background-color: #000;

color: #fff;

font-family: Arial, sans-serif;

}

#game {

position: relative;

max-width: 100%;

max-height: 100%;

overflow: hidden;

background-color: #000;

display: flex;

justify-content: center;

align-items: center;

}

#player {

position: absolute;

width: 180px;

height: 230px;

bottom: 20px;

left: 50%;

transform: translateX(-50%);

transition: left 0.5s, top 0.5s;

}

```

### JavaScript (`script.js`)

- \*\*Описание\*\*: Реализует логику игры, включая переходы между сценами, обработку кликов и обновление счетчика.

- \*\*Ключевые функции\*\*:

- \*\*Инициализация игры\*\*:

```javascript

function initializeGame() {

scenesPassed = 0;

updateScoreCounter();

loadScene(currentScene);

}

```

- \*\*Генерация случайных шансов для путей\*\*:

```javascript

function generateRandomChances() {

return {

winChance: (Math.random() \* 0.01 + 0.05).toFixed(2),

loseChance: (Math.random() \* 0.01 + 0.05).toFixed(2),

nextChance: (Math.random() \* 0.06 + 0.88).toFixed(2)

};

}

```

- \*\*Обновление счетчика пройденных сцен\*\*:

```javascript

function updateScoreCounter() {

const scoreCounter = document.getElementById('score-counter');

scoreCounter.textContent = `Пройдено сцен: ${scenesPassed}`;

}

```

- \*\*Загрузка сцены\*\*:

```javascript

function loadScene(sceneKey) {

const scene = maze[sceneKey];

if (!scene) return;

const backgroundElement = document.getElementById('background');

const backgroundsFolder = 'assets/backgrounds';

const backgroundImages = ['background1.png', 'background2.png', 'background3.png'];

const randomBackground = backgroundImages[Math.floor(Math.random() \* backgroundImages.length)];

if (randomBackground) {

backgroundElement.src = `${backgroundsFolder}/${randomBackground}`;

backgroundElement.classList.add('visible');

} else {

backgroundElement.classList.remove('visible');

}

scene.paths = [

{ label: "Выбрать путь 1", ...generateRandomChances() },

{ label: "Выбрать путь 2", ...generateRandomChances() },

{ label: "Выбрать путь 3", ...generateRandomChances() }

];

const gameDiv = document.getElementById('game');

document.querySelectorAll('.path').forEach(path => path.remove());

scene.paths.forEach((path, index) => {

const pathDiv = document.createElement('div');

pathDiv.className = 'path';

pathDiv.style.top = `${tunnelPositions[index].top}px`;

pathDiv.style.left = `${tunnelPositions[index].left}px`;

pathDiv.setAttribute('data-win', path.winChance);

pathDiv.setAttribute('data-lose', path.loseChance);

pathDiv.setAttribute('data-next', path.nextChance);

const label = document.createElement('span');

label.textContent = path.label;

pathDiv.appendChild(label);

gameDiv.appendChild(pathDiv);

});

document.querySelectorAll('.path').forEach(path => {

path.addEventListener('click', function() {

movePlayerToPath(this, function() {

const winChance = parseFloat(this.getAttribute('data-win'));

const loseChance = parseFloat(this.getAttribute('data-lose'));

const random = Math.random();

if (random < winChance) {

showScreen('win-screen');

} else if (random < winChance + loseChance) {

showScreen('lose-screen');

} else {

if (currentScene !== 'start') {

scenesPassed++;

updateScoreCounter();

}

resetPlayerPosition(function() {

currentScene = getRandomScene();

loadScene(currentScene);

});

}

}.bind(this));

});

});

}

```

## 5. Логика игры

### Структура лабиринта

- \*\*Описание\*\*: Лабиринт представлен в виде объекта `maze`, где каждая сцена имеет свои пути.

- \*\*Пример структуры\*\*:

```javascript

const maze = {

'start': { paths: [] },

'scene1': { paths: [] },

'scene2': { paths: [] },

// ... другие сцены

};

```

### Перемещение игрока

- \*\*Описание\*\*: Игрок может выбирать пути, нажимая на интерактивные элементы.

- \*\*Анимация перемещения\*\*:

```javascript

function movePlayerToPath(pathElement, callback) {

const player = document.getElementById('player');

const pathRect = pathElement.getBoundingClientRect();

const playerRect = player.getBoundingClientRect();

const gameRect = document.getElementById('game').getBoundingClientRect();

const targetX = pathRect.left - gameRect.left + pathRect.width / 2 - playerRect.width / 2;

const targetY = pathRect.top - gameRect.top + pathRect.height / 2 - playerRect.height / 2;

const offsetX = targetX - (playerRect.left - gameRect.left);

const offsetY = targetY - (playerRect.top - gameRect.top);

player.style.transition = 'transform 0.5s';

player.style.transform = `translate(${offsetX}px, ${offsetY}px)`;

setTimeout(callback, 800);

}

```

### Случайные исходы

- \*\*Описание\*\*: При выборе пути генерируется случайное число, определяющее исход (победа, поражение, переход).

- \*\*Логика случайных исходов\*\*:

```javascript

path.addEventListener('click', function() {

const winChance = parseFloat(this.getAttribute('data-win'));

const loseChance = parseFloat(this.getAttribute('data-lose'));

const random = Math.random();

if (random < winChance) {

showScreen('win-screen');

} else if (random < winChance + loseChance) {

showScreen('lose-screen');

} else {

if (currentScene !== 'start') {

scenesPassed++;

updateScoreCounter();

}

resetPlayerPosition(function() {

currentScene = getRandomScene();

loadScene(currentScene);

});

}

});

```

## 6. Интерфейс пользователя

### Элементы управления

- \*\*Кнопки\*\*: Для начала игры и рестарта на экранах победы и поражения.

- \*\*Интерактивные элементы\*\*: Для выбора путей на игровом экране.

### Визуальные эффекты

- \*\*Анимации\*\*: Для перемещения игрока и отображения исходов.

- \*\*Эффекты наведения\*\*: Для кнопок и интерактивных элементов.

## 7. Тестирование

### Функциональное тестирование

- \*\*Проверка переходов\*\*: Убедитесь, что переходы между экранами работают корректно.

- \*\*Логика случайных исходов\*\*: Проверьте, что исходы (победа, поражение, переход) определяются правильно.

- \*\*Счетчик пройденных сцен\*\*: Убедитесь, что счетчик обновляется при переходе на новую сцену.

### Тестирование интерфейса

- \*\*Отображение элементов\*\*: Проверьте, что все элементы отображаются корректно на различных устройствах.

- \*\*Интерактивность\*\*: Убедитесь, что все кнопки и интерактивные элементы работают должным образом.

- \*\*Анимации и эффекты\*\*: Проверьте, что все визуальные эффекты работают плавно.

## 8. Заключение

## Интерфейс пользователя

Интерфейс пользователя в игре "Robot Dungeon" состоит из нескольких ключевых элементов, которые обеспечивают взаимодействие с игроком. Эти элементы включают в себя экраны, кнопки, счетчики и интерактивные области.

### 1. Экраны

Игра состоит из четырех основных экранов:

- \*\*Стартовый экран\*\*: Первый экран, который видит пользователь.

- \*\*Игровой экран\*\*: Основной экран, на котором происходит игровой процесс.

- \*\*Экран победы\*\*: Показывается при победе игрока.

- \*\*Экран поражения\*\*: Показывается при поражении игрока.

#### HTML-структура экранов

```html

<!-- Стартовый экран -->

<div id="start-screen">

<img id="start-background" src="assets/Title\_V2.gif" alt="Start Background">

<img id="start-player" src="assets/Bouncing\_Bobbo.gif" alt="Start Player">

<button id="start-button">Начать игру</button>

</div>

<!-- Игровой экран -->

<div id="game" style="display: none;">

<div id="score-counter">Пройдено сцен: 0</div>

<img id="background" src="" alt="Background">

<img id="player" src="assets/player/Bobbo v3.webp" alt="Player">

</div>

<!-- Экран победы -->

<div id="win-screen" style="display: none;">

<h1>You Won!</h1>

<img id="win-player" src="assets/win/Bobby\_dancing.gif" alt="Bobby\_dancing.gif">

<button id="restart-win-button">Играть снова</button>

</div>

<!-- Экран поражения -->

<div id="lose-screen" style="display: none;">

<h1>Game Over</h1>

<img id="lose-player" src="assets/lose/Bobby\_dead.gif" alt="Bobby\_dead.gif">

<button id="restart-lose-button">Играть снова</button>

</div>

```

### 2. Кнопки

Кнопки используются для взаимодействия с пользователем. Они позволяют начать игру, а также перезапустить её в случае победы или поражения.

#### Пример кнопки

```html

<button id="start-button">Начать игру</button>

```

- \*\*Атрибуты\*\*:

- `id`: Уникальный идентификатор для стилизации и управления через JavaScript.

- `textContent`: Текст, отображаемый на кнопке.

- \*\*Стилизация\*\*:

- Кнопки стилизованы с помощью CSS для улучшения внешнего вида и удобства использования.

```css

button {

font-size: 24px;

padding: 10px 20px;

background-color: #fff;

color: #000;

border: none;

cursor: pointer;

margin-top: 20%;

}

button:hover {

background-color: #ddd;

}

```

### 3. Счетчик пройденных сцен

Счетчик отображает количество сцен, которые игрок успешно прошел. Он обновляется каждый раз, когда игрок переходит на новую сцену.

#### HTML-структура счетчика

```html

<div id="score-counter">Пройдено сцен: 0</div>

```

- \*\*Обновление счетчика\*\*:

- Счетчик обновляется с помощью JavaScript-функции `updateScoreCounter`.

```javascript

function updateScoreCounter() {

const scoreCounter = document.getElementById('score-counter');

scoreCounter.textContent = `Пройдено сцен: ${scenesPassed}`;

}

```

### 4. Интерактивные области (пути)

Интерактивные области представляют собой пути, которые игрок может выбрать для перехода на новую сцену. Они отображаются на игровом экране и реагируют на клики.

#### HTML-структура пути

```html

<div class="path" data-win="0.05" data-lose="0.05" data-next="0.90">

<span>Выбрать путь 1</span>

</div>

```

- \*\*Атрибуты\*\*:

- `class="path"`: Класс для стилизации и управления через JavaScript.

- `data-win`, `data-lose`, `data-next`: Атрибуты данных, хранящие вероятности исходов.

- \*\*Стилизация\*\*:

- Пути стилизованы для улучшения видимости и интерактивности.

```css

.path {

position: absolute;

width: 120px;

height: 60px;

background-color: rgba(255, 255, 255, 0.3);

border: 2px solid #fff;

border-radius: 10px;

cursor: pointer;

display: flex;

justify-content: center;

align-items: center;

text-align: center;

font-size: 16px;

color: #fff;

transition: background-color 0.3s;

}

.path:hover {

background-color: rgba(255, 255, 255, 0.5);

}

```

### 5. Анимации и визуальные эффекты

Анимации используются для улучшения визуального восприятия игры. Они применяются к перемещению игрока и изменению состояния элементов.

#### Пример анимации перемещения игрока

```javascript

function movePlayerToPath(pathElement, callback) {

const player = document.getElementById('player');

const pathRect = pathElement.getBoundingClientRect();

const playerRect = player.getBoundingClientRect();

const gameRect = document.getElementById('game').getBoundingClientRect();

const targetX = pathRect.left - gameRect.left + pathRect.width / 2 - playerRect.width / 2;

const targetY = pathRect.top - gameRect.top + pathRect.height / 2 - playerRect.height / 2;

const offsetX = targetX - (playerRect.left - gameRect.left);

const offsetY = targetY - (playerRect.top - gameRect.top);

player.style.transition = 'transform 0.5s';

player.style.transform = `translate(${offsetX}px, ${offsetY}px)`;

setTimeout(callback, 800);

}

```