Ramo Estudantil IEEE - UEL

ANA LAURA MACHADO ARAÚJO CAETANO
(ana.lauracaetano.9@uel.br)
DANIEL FIORI SOUZA (daniel.fiori.souza@uel.br)
LIVIA KOUKETSU DA SILVA (livia.kouketsu@uel.br)
THIAGO PEREIRA DE SOUZA (thiagop.070331@uel.br)

RELATÓRIO FINAL:

Projeto Campo Minado

Londrina
2023

Ramo Estudantil IEEE - UEL

ANA LAURA MACHADO ARAÚJO CAETANO
DANIEL FIORI SOUZA
LIVIA KOUKETSU DA SILVA
THIAGO PEREIRA DE SOUZA

RELATÓRIO FINAL:

Projeto Campo Minado

Relatório apresentado ao Ramo Estudantil IEEE da Universidade Estadual de Londrina.

Diretor de Projetos: Nathan Andreani Netzel
Gestores de Projetos: Daniel Tresse Dourado, Levi Monteiro dos Santos

Londrina
2023

Contato do Ramo: sb.uel@ieee.org
Institute of Electrical and Electronics Engineers – IEEE
Universidade Estadual de Londrina - UEL • Paraná - Brasil

CAETANO, Ana Laura Machado Araújo. SOUZA, Daniel Fiori. DA SILVA, Livia Kouketsu. DE SOUZA, Thiago Pereira. **Relatório Final**: Projeto Campo Minado. 2023. 14 folhas. Relatório apresentado ao Ramo Estudantil IEEE da Universidade Estadual de Londrina, Londrina, 2023.

RESUMO

O projeto Campo Minado trata-se de uma recriação de um jogo, o clássico campo minado, na qual os jogadores poderão escolher o nível de dificuldade do jogo, baseando-se no tamanho do mapa a ser jogado. Para desenvolver o projeto, a equipe utilizou a linguagem de programação de web JavaScript. Também foi necessário o conhecimento em HTML, para organizar o conteúdo da página de web, e CSS, para manusear o visual e design da página web.

Palavras-chave: Campo minado. JavaScript. CSS. HTML.

Sumário

1. INTRODUÇÃO

Um dos mais clássicos e conhecidos jogos, o Campo Minado desafia e proporciona entretenimento aos seus jogadores. Ademais, o jogo estimula as habilidades lógicas e estratégicas do jogador, tornando-se não só um instrumento de lazer aos usuários, mas também uma forma de exercitar a mente e a capacidade lógica.

Nesse contexto, o projeto busca recriar o jogo, criando níveis de dificuldade e criando um design moderno e simples aos seus usuários. Além disso, o projeto visa explorar os conceitos de programação envolvidos, instigando a equipe a utilizar os conhecimentos já adquiridos e buscar um maior aprendizado sobre a linguagem utilizada, a aplicação de estruturas de dados, algoritmos, e todos os outros conhecimentos necessários para o desenvolvimento do projeto.

Ao longo deste artigo, será abordado todo o processo de desenvolvimento do jogo, desde a criação da estrutura básica do tabuleiro até a implementação que possibilita a interatividade, expondo de forma abrangente o processo de criação de jogos.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 CAMPO MINADO

O jogo Campo Minado consiste em um quadro de células, chamadas de “campos”, alguns destes campos possuem minas e outros não. O tabuleiro começa com todos os campos cobertos, ao iniciar a partida, o jogador deve revelar um campo. Se o campo descoberto acionar uma mina, esta explode e o jogo termina. O objetivo do jogador é revelar todas os campos que não possuem minas

2.2 JAVASCRIPT

A linguagem JavaScript é uma linguagem de programação de web, de alto nível e usada na maioria dos sites atuais. Por ser uma linguagem interpretada, qualquer navegador ou plataforma pode ler o código programado, desde que possua um interpretador de JavaScript, motivo pelo qual a equipe optou por usá-la. Para a realização do projeto, foram utilizadas funções, em JavaScript, necessárias ao bom funcionamento do jogo, que serão explicadas na seção 3.

2.3 HTML

HTML, ou *Hyper Text Markup Language*, consiste em uma linguagem para marcação de hipertexto, isto é, um documento para web capaz de se interligar com outros documentos da web.

2.4 CSS

CSS, ou *Cascading Style Sheets*, consiste em uma folha de estilo em cascata. O CSS é um recurso aplicado a linguagens de marcação como HTML e tem esse nome pois aplica-se a estruturas por meio de regras de prioridade. Esse recurso permite garantir uma boa estética e interatividade ao site.

3. METODOLOGIA

Inicialmente, para gerar a matriz correspondente ao quadro de células, ou seja, ao tabuleiro do jogo, foi criada a função gerarMatriz (Figura 1). Esta função cria uma matriz de ‘1’ linha e ‘c’ colunas.

Figura 1: função gerarMatriz

```
function gerarMatriz(l, c) {
    matriz = [];
    for (var i = 0; i < l; i++) {
        matriz[i] = new Array(c).fill(0);
    }
    console.log(matriz);
}
```

Fonte: o próprio autor

Após a criação da matriz, criou-se a função gerarTabela (Figura 2), responsável por estruturar a representação visual do tabuleiro do jogo, através de uma tabela HTML. Cada célula criada na tabela equivale a um campo dentro da matriz do jogo.

Figura 2: função gerarTabela

```
function gerarTabela(l, c) {
    gerarMatriz(l, c);
    var str = "";
    for (var i = 0; i < l; i++) {
        str += "<tr>";
        for (var j = 0; j < c; j++) {
            str += "<td class='blocked'></td>";
        }
        str += "</tr>";
    }
    tabela.innerHTML = str;
}
```

Fonte: o próprio autor

Após isso, foi incrementada a função mostrarMatriz (Figura 3). Basicamente, esta função verifica cada célula da tabela e, se o valor correspondente da matriz for “-1”, exibe um ícone de bomba, se não, exibe um valor numérico armazenado na matriz.

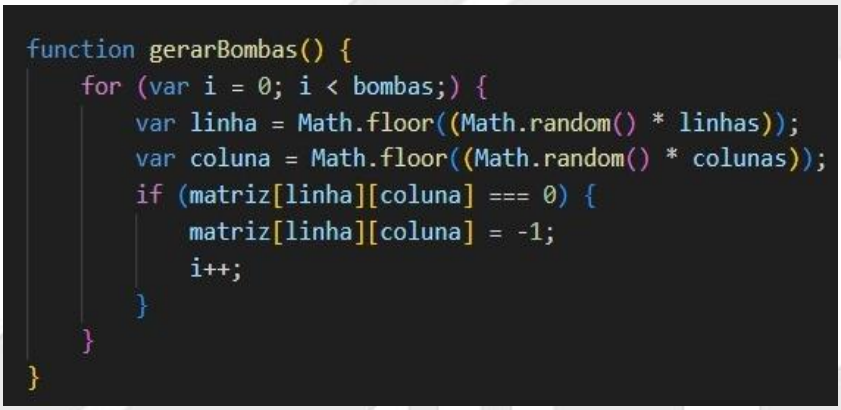
Figura 3: função mostrarMatriz

```
function mostrarMatriz() {
    for (var i = 0; i < linhas; i++) {
        for (var j = 0; j < colunas; j++) {
            if (matriz[i][j] === -1) {
                tabela.rows[i].cells[j].innerHTML = "&#128163;";
            } else {
                tabela.rows[i].cells[j].innerHTML = matriz[i][j];
            }
        }
    }
}
```

Fonte: o próprio autor

Em seguida, foi feita uma função para gerar as bombas no programa. Esta função (Figura 4) é responsável por gerar aleatoriamente o local em que as bombas serão armazenadas na matriz.

Figura 4: função gerarBombas



```
function gerarBombas() {
    for (var i = 0; i < bombas;) {
        var linha = Math.floor((Math.random() * linhas));
        var coluna = Math.floor((Math.random() * colunas));
        if (matriz[linha][coluna] === 0) {
            matriz[linha][coluna] = -1;
            i++;
        }
    }
}
```

Fonte: o próprio autor

Após gerar as bombas, foi criada a função gerarNumero(1, c) (Figura 5). Essa função tem com finalidade calcular o número de bombas adjacentes a uma célula específica, e armazenar esse valor na matriz. Ela verifica as células ao redor da célula

atual e conta quantas bombas existem ao redor desta.

Figura 5: gerarNumero(1, c).

```
function gerarNumero(l, c) {
    var count = 0;
    for (var i = l - 1; i <= l + 1; i++) {
        for (var j = c - 1; j <= c + 1; j++) {
            if (i >= 0 && i < linhas && j >= 0 && j < colunas) {
                if (matriz[i][j] === -1) {
                    count++;
                }
            }
        }
    }
```

Fonte: o próprio autor

Em seguida, foi incrementada a função bandeira(event) (Figura 6). Esta função lida com eventos de clique com o botão direito do mouse (evento de contexto). Ela alternadamente marca uma célula como uma bandeira ou a desmarca, trocando a classe da célula entre "blocked" e "flag" e exibindo um ícone de bandeira quando marcada.

Figura 6: bandeira (event)

```
function bandeira(event) {
    var cell = event.target;
    var linha = cell.parentNode.rowIndex;
    var coluna = cell.cellIndex;
    if (cell.className === "blocked") {
        cell.className = "flag";
        cell.innerHTML = "&#128681;";
    } else if (cell.className === "flag") {
        cell.className = "blocked";
        cell.innerHTML = "";
    }
    return false;
}
```

Fonte: o próprio autor

Em seguida, criou-se a função init() (Figura 7). Esta função é chamada quando a página é carregada. Ela inicializa a tabela, gera as bombas, calcula os números das células adjacentes e configura os eventos de clique para a tabela.

Figura 7: init():

```
function init() {
    tabela = document.getElementById("tabela");
    tabela.onclick = verificar;
    tabela.oncontextmenu = bandeira;

    gerarTabela(linhas, colunas);
    gerarBombas();
    gerarNumeros();
}
```

Fonte: o próprio autor

A função limparCelulas(l, c) (Figura 8) é usada para "limpar" células vazias no jogo. Quando uma célula vazia (com valor 0) é clicada, ela se espalha para as células adjacentes vazias, revelando mais células vazias e números.

Figura 8: limparCelulas

```
function limparCelulas(l, c) {
    for (var i = l - 1; i <= l + 1; i++) {
        for (var j = c - 1; j <= c + 1; j++) {
            if (i >= 0 && i < linhas && j >= 0 && j < colunas) {
                var cell = tabela.rows[i].cells[j];
                if (cell.className !== "blank") {
                    switch (matriz[i][j]) {
                        case -1:
                            break;
                        case 0:
                            cell.innerHTML = "";
                            cell.className = "blank";
                            limparCelulas(i, j);
                            break;
                        default:
                            cell.innerHTML = matriz[i][j];
                            cell.className = "n" + matriz[i][j];
                    }
                }
            }
        }
    }
}
```

Fonte: o próprio autor

Por fim, foi criada a função “fimDejogo” (Figura 9). Esta função verifica se todas as células bloqueadas ou marcadas como bandeira foram reveladas ou marcadas corretamente, indicando uma vitória. Se todas as bombas forem marcadas corretamente, o jogo é encerrado como uma vitória.

Figura 9: fimDejogo

```
function fimDeJogo() {
    var cells = document.querySelectorAll(".blocked, .flag");
    if (cells.length === bombas) {
        mostrarBombas();
        tabela.onclick = undefined;
        tabela.oncontextmenu = undefined;
        mostrarResultado("vitoria");
    }
}
```

Fonte: o próprio autor

4. RESULTADOS E DISCUSSÕES

Após criar todas as funções, a equipe realizou o teste de funcionamento do jogo e verificou que todas as funções criadas foram bem executadas e desempenharam seus objetivos.

A tela inicial mostrou da forma prevista os níveis de dificuldades, o tamanho dos tabuleiros e quantidades de bombas em cada nível de dificuldade estavam iguais aos esperados, e os campos descobertos mostraram as bombas, os números das bombas ao redor e realizaram o efeito cascata, de acordo com o caso específico de cada célula.

Entretanto, é válido ressaltar que existem melhorias que poderiam ser feitas para tornar o jogo mais completo, como por exemplo um recurso para salvar o tabuleiro do jogador caso este quisesse pausar o jogo. Porém, essas melhorias não são fundamentais para a desenvolvimento do jogo, sendo apenas funcionalidades extras que tornariam a experiência do jogador mais completa.

5. CONCLUSÕES

Após a realização do teste, verificou-se que o projeto foi executado de acordo com a expectativa da equipe, desempenhando todos os passos necessários para o bom funcionamento do jogo. Assim, o projeto foi finalizado.

REFERÊNCIAS BIBLIOGRÁFICAS

Campo Minado. Disponível em: https://campo-minado.com/introdu%C3%A7%C3%A3o/. Acesso em: 22 set. 2023.

Flanagan, D. (2012). JavaScript: O Guia Definitivo. Bookman Editora. ISBN 9788565837484.

Silva, M. S. (2008). *Criando Sites com HTML: Sites de Alta Qualidade com HTML e CSS*. Novatec Editora. ISBN 8575221663, 9788575221662. 432 páginas.

CSS. Disponível em: https://coodesh.com/blog/dicionario/o-que-e-css/#:~:text=CSS%20significa%20Cascading%20Style%20Sheets,marca%C3%A7%C3%A3o%20HTML%2C%20XML%20e%20XHTML. Acesso em: 22 set. 2023.