MULTI PUZZLE

Diana Parreira
Escola Superior de Gestão e Tecnologia de Santarém
180100237@esg.ipsantarem.pt

Filipe Correia
Escola Superior de Gestão e Tecnologia de Santarém
180100361 esg.ipsantarem.pt

Luís Meneses
Escola Superior de Gestão e Tecnologia de Santarém
180100353@esg.ipsantarem.pt

RESUMO

O Multi Puzzle é um jogo puzzle para a web que se foca em solucionar quebra-cabeças.

PALAVRAS-CHAVE

Puzzle, função, JavaScript, CSS

1. INTRODUÇÃO

Com este trabalho pretendemos desenvolver um jogo web que permite ao jogador solucionar um quebra-cabeças.

O jogador pode escolher 8 imagens para jogar, tendo três níveis de dificuldade. O nível fácil tem um grau de dificuldade baixo, o puzzle é constituído por 15 peças. O nível intermédio é constituído por 35 peças. O nível difícil é constituído por 63 peças.

Para o desenvolvimento do jogo web foram também utilizadas as linguagens de programação CSS, HTML e JavaScript.

2. LINGUAGENS E MÉTODOS

Foram utilizadas as seguintes linguagens: HTML, CSS, JAVASCRIPT.

2.1 Html

Foi criado um contentor para os puzzles disponíveis, onde foram adicionadas as 8 imagens de jogo.

```
<div id="thumbSet" class="tesquerda">
    <img src="images/algarve.png" class="botaoimg" alt="0">
    <img src="images/casino.png" class="botaoimg" alt="1">
    <img src="images/chaves.png" class="botaoimg" alt="2">
    <img src="images/girl.jpg" class="botaoimg" alt="3">
    <img src="images/obidos.jpg" class="botaoimg" alt="4">
    <img src="images/panda.jpg" class="botaoimg" alt="5">
    <img src="images/vchaves.jpg" class="botaoimg selecionada" alt="6" >
    <img src="images/vidago.jpg" class="botaoimg" alt="7">
</div>
```

Figura 1- Contentor de Imagens

2.2 JavaScript

2.2.1 Função Reset

A função reset permite efetuar o reset ao mapa, limpando o número de jogadas e reiniciando o mapa. A função reset por sua vez chama a funções desenharMapa para desenhar o mapa novamente e a função shuffle para baralhar as peças no mapa.

```
function reset(img) {

    for(let i=0;i<8;i++){
    if(document.getElementById( elementId: "img"+i).classList.contains("selecionada")){
        document.getElementById( elementId: "img"+i).classList.remove( tokens: "selecionada");
    }}
    win=false;
    nrI=img;
    nMoves=0;
    document.getElementById( elementId: "contador").innerHTML=nMoves;
    numeroP=0;
    let stage=document.getElementById( elementId: "stage");
    while (stage.firstChild) {
        stage.firstChild.remove();
    }
    mapa=[];
    solvedMap=[];
    mapsolved=false;
    desenharMapa(l,c,nrI);
    shuffle( numero: 15);
    checkPossibleMoves();
    resolvido(l,c,nrI);
    document.getElementById( elementId: "img"+img).classList.add("selecionada");
}
```

Figura 2- Function reset(img)

2.2.2 Função play

A função play permite ao utilizador jogar, apenas as peças estejam adjacentes à peça livre, chamando a função checkPossibleMoves que calcula as peças que são permitidas mover.

```
function play(e){

if(!win){
    console.log(possibleMoves);
    let idDiv=e.currentTarget.id;
    if(possibleMoves.includes(e.currentTarget)){
        sounds.jogar.play();
        trocarPecas(idDiv,numeroP-1);
        nMoves++;
        document.getElementById( elementId: "contador").innerHTML=nMoves;
        checkWin();
        checkPossibleMoves();
    }else{
        sounds.erro.play();
    }
}
}
```

Figura 3-Function play(e)

2.2.3 Função desenharArray

A função desenharArray adiciona dentro da stage o array de linhas e as colunas.

```
function desenharArray(linhas,colunas,array){
    let stage=document.getElementById( elementId: 'stage');
    while (stage.firstChild) {
        stage.firstChild.remove();
    }

    for (let linha = 0; linha < linhas; linha++) {
        for (let coluna = 0; coluna < colunas; coluna++) {
            stage.appendChild(array[linha][coluna]);
        }
    }
}
```

Figura 4-Function desenharArray(linhas,colunas,array)

2.2.4 Função checkPossibleMoves

Esta função encontra as coordenadas das peças que estão em cima, em baixo e dos lados do espaço em branco e adiciona as mesmas ao array possibleMoves, este array será mais tarde usado para verificar se as peças se podem mover ou não.

```
function checkPossibleMoves() {
    possibleMoves=[];
    let whiteBox = searchNoArray( id: numeroP - 1);
    let whiteBoxRow = whiteBox[0];
    let whiteBoxColumn = whiteBox[1];
    try {...} catch (e) {

    }

    try {
        if (mapa[whiteBoxRow + 1][whiteBoxColumn] !== undefined) { //bottom
            possibleMoves.push(mapa[whiteBoxRow + 1][whiteBoxColumn]);
        }
    } catch (e) {

    }

    try {
        if (mapa[whiteBoxRow][whiteBoxColumn - 1] !== undefined) { //left
            possibleMoves.push(mapa[whiteBoxRow][whiteBoxColumn - 1]);
        }
    } catch (e) {
    }
    try {
        if (mapa[whiteBoxRow][whiteBoxColumn + 1] !== undefined) { //right
            possibleMoves.push(mapa[whiteBoxRow][whiteBoxColumn + 1]);
        }
    } catch (e) {
```

Figura 5-Function checkPossibleMoves()

2.2.5 Função criardivsjogo

De forma que a imagem seja divida em várias imagens foi criada a função criardivsjogo, que permite criar a partir da imagem original várias imagens através da criação de novas secções, onde cada div corresponde contem uma imagem.

```
function criardivsjogo(linhas, colunas) {
    tamanhodiv = (366 / colunas);
    let stage = document.getElementById( elementId: "stage");
    for (let linha = 0; linha < linhas; linha++) {
        let arraycolunas = [];
        for (let coluna = 0; coluna < colunas; coluna++) {
            let cel = document.createElement( tagName: "div");
            cel.id = numeroP;
            cel.style.float = "left";
            cel.style.width = tamanhodiv + "px";
            cel.style.height = tamanhodiv + "px";

            cel.addEventListener( type: "click", play, options: false);

            stage.appendChild(cel);
            arraycolunas.push(cel);
            numeroP++;
        }
        mapa.push(arraycolunas);
    }
}
```

Figura 6-Function criardivsjogo(linhas,colunas)

2.2.6 Função resolvido

A função resolvido(linhas, colunas, imagem) desenha um mapa resolvido de linhas, ou seja permite criar o array com imagens da solução do puzzle.

```
function resolvido(linhas, colunas, imagem) {

    tamanhoImg = (366 / colunas);
    criarArraysolucao(linhas, colunas);
    let num = 0;
    for (let linha = 0; linha < linhas; linha++) {
        let colunaArr = solvedMap[linha];
        for (let coluna = 0; coluna < colunas; coluna++) {

            let celula = document.createElement( tagName: "img");
            celula.classList.add("cell");
            celula.style.backgroundImage = "url('images/" + imagens[imagem] + "')";
            celula.style.backgroundPositionX = -(Math.floor( x: num % colunas)) * tamanhoImg + "px";
            celula.style.backgroundPositionY = -(Math.floor( x: num / colunas)) * tamanhoImg + "px";
            celula.style.width = tamanhoImg + "px";
            celula.style.height = tamanhoImg + "px";
            colunaArr[coluna].appendChild(celula);
            num++;
        }
    }

    while (solvedMap.firstChild) {
        solvedMap.firstChild.remove();
    }
}
```

Figura 7-Function resolvido(linhas,colunas,imagem)

2.2.7 Função desenharMapa

A função desenharMapa desenha em função do tamanho da imagem original o mapa de jogo, com menos uma peça, para que se possam mover as restantes peças, de forma a encontrar a solução de jogo.

```
function desenharMapa(linhas, colunas, imagem) {
    tamanhoImg = (366 / colunas);
    criardivsjogo(linhas, colunas);
    let num = 0;
    for (let linha = 0; linha < linhas; linha++) {
        let colunaArr = mapa[linha];
        for (let coluna = 0; coluna < colunas; coluna++) {

            let celula = document.createElement( tagName: "img");
            celula.classList.add("cell");
            celula.style.backgroundImage = "url('images/" + imagens[imagem] + "')";
            celula.style.backgroundPositionX = -(Math.floor( x: num % colunas)) * tamanhoImg + "px";
            celula.style.backgroundPositionY = -(Math.floor( x: num / colunas)) * tamanhoImg + "px";
            celula.style.width = tamanhoImg + "px";
            celula.style.height = tamanhoImg + "px";
            colunaArr[coluna].appendChild(celula);
            num++;
        }
    }
    let colunaArr = mapa[linhas - 1];
    colunaArr[colunas - 1].firstChild.remove();
}
```

Figura 8-Function desenharMapa(linhas,colunas,imagem)

2.2.8 Função trocarPecas

A função permite que seja feita a troca das peças durante o jogo, esta função faz com que ao remover a imagem div1, ela seja posteriormente adicionada na div2, consequentemente a div2 é adicionada na div1.

```
function trocarPecas(first, second) {
    let img1 = document.getElementById(first).firstChild;
    let img2 = document.getElementById(second).firstChild; //imagens das divs
    let div1 = document.getElementById(first);
    let div2 = document.getElementById(second);  //divs

    if (first === numeroP - 1 || second === numeroP - 1) {
        if (first === numeroP - 1) {
            document.getElementById(second).firstChild.remove();
            document.getElementById(first).appendChild(img2);


        } else {
            document.getElementById(first).firstChild.remove();
            document.getElementById(second).appendChild(img1);
        }


    } else {

        document.getElementById(second).firstChild.remove();
        document.getElementById(first).firstChild.remove();  //remover as images das divs


        document.getElementById(first).appendChild(img2); //meter a segunda imagem na primeira div
        document.getElementById(second).appendChild(img1); //meter a primeira imagem na segunda div
        sounds.success.play();
    }


    div1.id = second;  //mudar o nome da primeira para o da segunda
    div2.id = first;  //vice versa do q está cá em cima
```

Figura 9-Function trocarPecas(first,second)

2.2 CSS

Para animação do jogo foi utilizada a linguagem CSS, onde foi utilizado o hover para realçar a imagem que o jogador pretende escolher. Para a imagem selecionada foi adiciona um keyframe rotation que permite que enquanto o jogador estiver a jogar a imagem selecionada, este mantenha se em rotação.

```css
.botaoimg{
  float: right;
  border-radius: 50%;
  display: block;
  width: 40px;
  height: 40px;
  background-position: 0 0;
  background-repeat: no-repeat;
  background-size: cover;
  margin: 3px 5px 3px 0px;
  cursor: pointer;
}

.botaoimg:hover {
  width: 60px;!important;
  height: 60px;!important;
}

.selecionada{
  animation: rotation 5s infinite linear;
}

@keyframes rotation {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(359deg);
  }
}
```

Figura 10-CSS animações

Resultados

Criamos um puzzle que permite ao utilizador escolher entre 8 imagens e três graus de dificuldade qual o que pretende jogar.

O som funciona corretamente apenas no browser Mozilla Firefox, os restantes browsers não tem permissão de alterar o auto-play.

3. CONCLUSÃO

Este jogo web permite aos seus utilizadores jogaram o jogo MultiPuzzle com várias imagens e vários níveis de dificuldade.

REFERÊNCIAS

Informação disponibilizada pelo Professor Claúdio Barradas através do material disponibilizado no moodle: https://moodle.esgt.ipsantarem.pt/course/view.php?id=1521