JavaScript

Tiago Lopes Telecken

telecken@gmail.com

Estrutura, Fundo, Responsividade

Todos elementos do jogo(personagens, obstaculos, tiros, inimigos,...) devem ser um elemento do HTML (normalmente div).

Todos elementos devem estar dentro de um container (normalmente uma div. A div de fundo do jogo).

A div de fundo do jogo:

- **container aonde todos os elementos do jogo estão inseridos. Ao mover esta div os elementos internos acompanham pois o posicionamento absoluto é em relação ao elemento pai.**
- **Ao redimensionar a div, os elementos internos são redimensionados se estiverem usando medidas como % ou vw.**
- **Normalmente as movimentações são melhores em px para não perder a precisão. (dimensões e posições em vw. Movimento em px)**

Responsividade: Colocar as dimensões dos elementos em vw ou % proporciona responsividade.

Movimento

- **Movimento:** Os elementos que se movem devem ter **posicionamento absoluto** para não gerar movimentos em outros elementos
- Todos elementos devem se movimentar dentro do container “Fundo”

- Pode-se movimentar elementos utilizando a função **setInterval**

  timerPersonagem = setInterval(moverPersonagem, 600)

- Para parar os movimentos deve-se usar a função **clearInterval**

  clearInterval(timerPersonagem)

- Para mudar um personagem de direção deve-se parar antes. Ou seja fazer um **clearInterval** antes de redirecionar um elemento com outro **setInterval**
- Uma alternativa é mover elementos com animações css

Alterar/Acessar propriedades de um elemento

```
//alterar o valor de uma propriedade css e acessar uma propriedade css de um elemento HTML

div1.style.width = parseInt(getComputedStyle(div1).width)+5;

// o div1.style.width = altera a propriedade width do elemento div1
// o getComputedStyle(div1).width retorna a propriedade width do elemento div1

//uma alternativa ao getComputedStyle é o getBoundingClientRect.
//Somente distâncias e tamanhos da box do elemento.
nave = div1.getBoundingClientRect();
NaveTamanho= nave.width
```

Adicionar, criar personagens ou objetos

```
//adicionar um personagem
NovoItem= document.createElement("div");
NovoItem.setAttribute("class", "item");
pai.appendChild(NovoItem);

//deletar um personagem
pai.removeChild(item);
```

Imagens, sons

```
//audio
musica = new Audio('../audio/musica.mp3');
musica.play();
musica.pause();

musica.load();
musica.play();
musica.volume = 0.05;
musica.loop = true;
**********************************
/* imagem: se colocar a imagem dentro de uma div a imagem acompanha o tamanho e movimento da div. Uma alternativa é colocar a imagem como background da div no arquivo css*/
<div id="div1">
    <img alt="" src="./imagens/nave.jpg">
</div>
```

Gerar números aleatórios

- **Abaixo comando que retorna um número aleatório, inteiro entre 0 e 499**

```
function random(){
  return Math.floor(Math.random() * 500);
}
```

Capturar posição touch/mouse

```
//Pegar a posição de onde o usuário tocou na tela
document.addEventListener('touchmove', moverJogador);
function moverJogador(event) {
let posicaoJogador = event.touches[0].clientX
}

//Pegar a posição do mouse
 document.addEventListener('mousemove', moverJogador);
function moverJogador(event) {
let posicaoJogador = event.clientX
}
```

Capturar Eventos do teclado

```
//define manipuladores de evento;
element.addEventListener("onkeydown", (e)=>{verificaTecla(e)});


//funçao chamada pelo manipulador de evento
function verificaTecla(e) {
        if (e.keyCode == '38') {
                if (timer != null) {
                        para();
                        timer = setInterval("cima()",10);
                }
        }
}
```

Adicionar classes CSS com JS (Alternativa)

- **div2.classList.add("nave-subindo"); // adiciona e remove uma class**
- **div2.classList.remove("nave-subindo"); // CSS em um elemento html**
  // Pode-se adicionar várias classes
  // em um mesmo elemento

- **Pode-se criar visuais de elementos com comandos CSS e adicionar/alterar esses visuais com classList.add ou classList.remove**

- **Para movimentar personagens ou elementos na tela pode-se utilizar animações css (@keyframes, animation-duration, etc). Os testes de colisão e demais operações do JS irão funcionar nestes personagens).**
- **Os movimentos ou animações podem ficar armazenados em CSS e serem ativados com classList.add("mover") ou classList.remove("mover")**

Carregar página, botão iniciar

/* ****Ao carregar a página o código que está fora de funções será executado
O botão iniciar deverá ter um evento click que ao ser clicado chama a função iniciar
document.querySelector("#btnIniciar").addEventListener("click", () => { iniciar() });

Os procedimentos que vc quer executar somente uma vez, ao carregar a página ficam fora das funções.
Os procedimentos que vc quer executar toda vez que reiniciar o jogo devem estar na função iniciar/reiniciar

********** Botão iniciar
Colocar todos procedimentos iniciais em uma função iniciar.
Esta função é chamada pelo botão iniciar
Para que as variáveis declaradas dentro da função tenham visibilidade global deve-se declará-las globalmente antes.

Fim de jogo

********Para terminar o jogo
-mostrar imagens e mensagens de fim de jogo
-Fechar todos timers com clear interval

```
timer = setInterval("func()", 100);
...
clearInterval(timer);
```

-Terminar os procedimentos necessários
-mostrar o botão/div reiniciar