Programação Visual e Autoria Web

HTML5 Canvas (Cont), Listeners, setInterval

Universidade Federal do Rio Grande do Norte

Listeners

- O método addEventListener () anexa um listener de eventos ao elemento especificado sem substituir os listeners de eventos existentes.
- É possível adicionar muitos listeners de eventos a um elemento.
  - Por exemplo, é possível adicionar dois eventos de "clique“ ao mesmo elemento.
- Para remover o listener é utilizada a função removeEventListener ().

Listeners

- A sintaxe do uso de listeners é a seguinte:

```
element.addEventListener(event, function, useCapture);
```

- Event qualquer evento do DOM como “click”, “keydown”.
  - https://www.w3schools.com/jsref/dom_obj_event.asp
- Function é a função que será chamada quando o evento ocorrer.
- useCapture é um booleano que informa se o evento será tratado na capture phase ou bubbling phase.

SetInterval

- Podemos configurar uma função para que ela seja invocada a cada intervalo de tempo.

```
let gamerun = setInterval(draw, 250);
```

- Para cancelar a chamada utilizamos a função clearInterval.

```
clearInterval(gamerun);
```

Canvas

- Podemos utilizar o contexto do canvas para desenhar uma imagem a partir de um arquivo jpg ou png.
- Para isso é necessário ler o arquivo a partir da sua localização e passar tal imagem como parâmetro da função drawImage().
- Em muitos casos, o processo de carregar a imagem é mais lento que o processo de desenhar a imagem na tela, por isso, só podemos desenhar uma imagem se ela estiver carregada.

drawImage(img,sx,sy,swidth,sheight,x,y,width,height)

- img: Especifica a imagem, canvas ou elemento de vídeo.
- sx: Opcional. A coordenada X de onde se inicia o corte na imagem.
- sy: Opcional. A coordenada Y de onde se inicia o corte na imagem.
- swidth: Opcional. A largura da imagem cortada.
- sheight: Opcional. A altura da imagem cortada.
- x: A coordenada X onde a imagem será colocada no canvas.
- y: A coordenada Y onde a imagem será colocada no canvas.
- width: Opcional. A largura que a imagem vai usar.
- height: Opcional. A altura que a imagem vai usar.

```
context.drawImage(img,sx,sy,swidth,sheight,x,y,width,height);
```

drawImage(img, x,y)

```
let canvas = document.getElementById("tela");
let ctx = canvas.getContext("2d");

const imagem = new Image();
imagem.src = "character.png";

ctx.drawImage(imagem, 0, 0);
```

Esse código não vai funcionar, pois a imagem vai carregar depois da invocação da função `drawImage`.

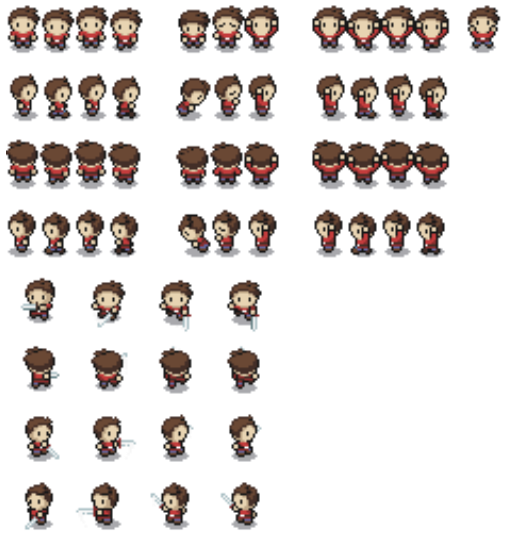
drawImage(img, x,y)

```
let canvas = document.getElementById("tela");
let ctx = canvas.getContext("2d");

const imagem = new Image();
imagem.src = "character.png";

imagem.onload = function() {
    ctx.drawImage(imagem, 0, 0);
};
```

Correção do código com a invocação da função de draw apenas depois que a imagem está carregada.

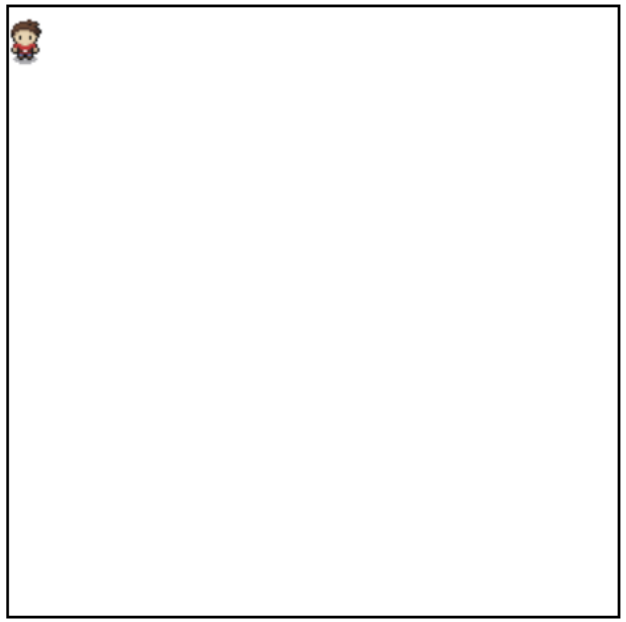
Canvas

- Carregamento de imagens para o canvas sob demanda pode ser problemático.
- Uma estratégia de solução é carregar um grande arquivo de imagem e usar partes desse arquivo quando necessário.
  - Realizar “clipping”



https://opengameart.org/content/zelda-like-tilesets-and-sprites

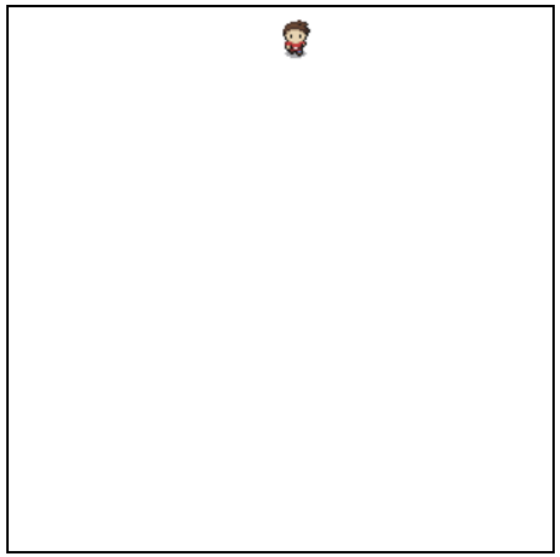
drawImage(img,sx,sy,swidth,sheight,x,y,width,height)



```
let canvas = document.getElementById("tela");
let ctx = canvas.getContext("2d");

const imagem = new Image();
imagem.src = "character.png";

const largura = 16;
const altura = 32;

imagem.onload = function() {
    ctx.drawImage(imagem, 0, 0, largura, altura, 0, 0, largura, altura);
};
```

drawImage(img,sx,sy,swidth,sheight,x,y,width,height)

```
const largura = 16;
const altura = 32;

let estado = 0;
let walkx = 150;
let walky = 0;

function draw() {
    ctx.clearRect(0, 0, 300, 300);
    ctx.drawImage(imagem, estado * largura, 0, largura, altura,
        walkx, walky, largura,altura
    );
    estado++;
    if (estado == 4) {
        estado = 0;
    }
}
let rungame = setInterval(draw, 100);
```



Bibliografia Básica

- DUCKETT, J. **Javascript e Jquery**: desenvolvimento de interfaces web interativas. Rio de Janeiro: Alta Bokks, 2016
- FLANAGAN, D. **JavaScript**: O Guia Definitivo. Bookman, 2012. ROGERS, Y.; SHARP, H.; PREECE, J. **Design de Interação: além da interação homem-computador**. Bookman, 2013. MEYER
- Web Content Accessibility Guidelines (WCAG) 2.1. Disponível em: https://www.w3.org/TR/WCAG21/
- https://htmlreference.io/forms/
- https://developer.mozilla.org/pt-BR/docs/Web/Guide/HTML/Canvas_tutorial/Drawing_shapes
- https://www.w3schools.com/graphics/canvas_coordinates.asp

Bibliografia Complementar

- AMARAL, L. G. **CSS Cascading Style Sheets**: guia de consulta rápida. 2. ed. São Paulo: Novatec, c2006.
- KAWANO, W. **Crie aplicativos Web com HTML, CSS, JavaScript, PHP, PostgreSQL, Bootstrap, AngularJS e Laravel**. Rio de Janeiro: Ciência Moderna, 2016.
- PUREWAL, S. **Aprendendo a Desenvolver Aplicações Web**. Novatec, 2014.
- TONSIG, S. L. **Aplicações na nuvem**: como construir com HTML5, javaScript, CSS, PHP e MYSQL. Rio de Janeiro: Ciência moderna, 2012.
- USABILIDADE.com. Disponível em <http://www.usabilidade.com/>
- TASK-Centered User Interface Design: A Practical Introduction. Disponível em <http://hcibib.org/tcuid/>