Computação Gráfica

Canvas Graphics
Scalable Vector Graphics (SVG)

Prof. Rodrigo Martins
rodrigo.martins@francomontoro.com.br

Alguns slides deste material foram cedidos pelo Prof. Jorge Cavalcanti da UNIVASF (UNIVERSIDADE FEDERAL DO VALE DO SÃO FRANCISCO)

HTML Canvas Graphics

- A tag <canvas> é um espaço onde pode-se desenhar elementos específicos, formas geométricas e imagens, utilizando a linguagem javascript.
- Pode-se também usar CSS para a tag.
- Por Javascript acessamos os métodos da canvas que desenham formas, linhas, caracteres e adicionar imagens.

HTML Canvas Graphics

- Deve-se saber que uma canvas é composta por pixels como em um plano cartesiano que começam em x = 0 e y =0 no canto superior esquerdo onde x é crescente para a direita e y é crescente para baixo.

HTML Canvas Graphics

- Seleciona-se a canvas e captura-se o contexto gráfico da mesma, pelo método getContext(), da seguinte forma:

```
1 | var c=document.getElementById("idCanvas");
2 | var ctx=c.getContext("2d");
```

HTML Canvas Graphics

- **Desenhando Linha**

- Usa-se no mínimo três métodos, são eles: moveTo(x,y), lineTo(x,y) e stroke().

```
1  var c=document.getElementById("idCanvas");
2  var ctx=c.getContext("2d");
3  ctx.moveTo(10,10);//ponto inicial
4  ctx.lineTo(150,50);//próximo ponto
5  ctx.stroke();//desenha
```

HTML Canvas Graphics

- **Desenhando um circulo**

- Precisa-se de dois métodos: arc(x,uy,r,aInicial,aFinal) que recebe as coordenadas onde será colocado o centro do circulo, o raio e os ângulos em relação aos quadrantes do círculo, e o já conhecido stroke().

- Usa-se o Math.PI * 2 onde deve-se passar o ângulo final, isso nada mais é que uma forma de desenhar um circulo completo. Mude esses valores para ver como fica.

```
1  var c=document.getElementById("idCanvas");
2  var ctx=c.getContext("2d");
3  ctx.arc(70,18,15,0,Math.PI * 2);
4  ctx.stroke();
```

HTML Canvas Graphics

- **Desenhando um Retângulo**

- No desenho do retângulo, mais um método é apresentado: rect(x,y,largura,altura).
- Em seguida o stroke().

```
1  var c=document.getElementById("idCanvas");
2  var ctx=c.getContext("2d");
3  ctx.rect(20,20,150,50);
4  ctx.stroke();
```

HTML Canvas Graphics

- **Adicionando imagens**

- Para imagens o processo é um pouco diferente, mas é tão simples quanto. Insancia-se um objeto do tipo Image e adiciona-se ao método onload que chama o método drawImage(Image,x,y) e, finalmente, passa-se para o objeto qual a url da imagem.

```
1  var c=document.getElementById("idCanvas");
2  var ctx=c.getContext("2d");
3  var img=new Image();
4  img.onload = function(){
5  ctx.drawImage(img,0,0);
6  };
7  img.src="img.png";
```

HTML Canvas Graphics - Praticando

Exemplo 1

- <canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">Se seu navegador não suportar você verá esta mensagem.
  </canvas>

```
<script>
    var c = document.getElementById("myCanvas");
    var ctx = c.getContext("2d");
    ctx.moveTo(0, 0); //ponto inicial
    ctx.lineTo(200, 100); //próximo ponto
    ctx.stroke(); //desenha
</script>
```

HTML Canvas Graphics - Praticando

Exemplo 2

- <canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;"> Se seu navegador não suportar você verá esta mensagem. </canvas>

```
<script>
    var c = document.getElementById("myCanvas");
    var ctx = c.getContext("2d");
    ctx.beginPath(); //serve para iniciar ou reiniciar o caminho do desenho
    ctx.arc(95, 50, 40, 0, 2 * Math.PI);
    ctx.stroke();
</script>
```

HTML Canvas Graphics - Praticando

Exemplo 3

- <canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">Se seu navegador não suportar você verá esta mensagem. </canvas>

```
<script>
    var c = document.getElementById("myCanvas");
    var ctx = c.getContext("2d");
    ctx.font = "30px Arial";
    ctx.fillText("Hello World", 10, 50);
</script>
```

HTML Canvas Graphics - Praticando

Exemplo 4

- <canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">Se seu navegador não suportar você verá esta mensagem.
  </canvas>

```
<script>
    var c = document.getElementById("myCanvas");
    var ctx = c.getContext("2d");
    ctx.rect(40, 40, 300, 100);
    ctx.stroke();
</script>
```

HTML Canvas Graphics - Praticando

Exemplo 5

- <canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">Se seu navegador não suportar você verá esta mensagem. </canvas>

```
<script>
    var c = document.getElementById("myCanvas");
    var ctx = c.getContext("2d");
    // gradiente
    var grd = ctx.createLinearGradient(0, 0, 400, 0);
    grd.addColorStop(0, "red");
    grd.addColorStop(1, "white");
    ctx.fillStyle = grd;
    ctx.fillRect(20, 20, 300, 160);
</script>
```

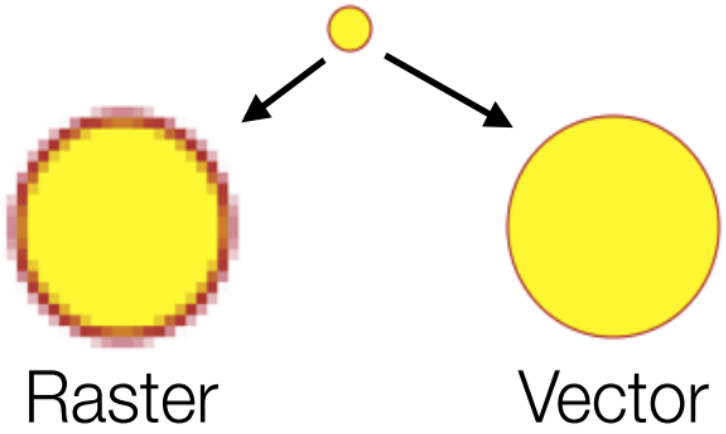
HTML Canvas Graphics - Praticando

Exemplo 6

```
<p>Imagem para usar:</p>
  <img id="scream" src="img1.png" alt="The Scream" width="220" height="277">

 <p>Tela para preencher:</p>
  <canvas id="myCanvas" width="250" height="300" style="border:1px solid #d3d3d3;">
   Se seu navegador não suportar você verá esta mensagem.
  </canvas>

  <p><button onclick="myCanvas()">Clique para preencher</button></p>

  <script>
   function myCanvas() {
     var c = document.getElementById("myCanvas");
     var ctx = c.getContext("2d");
     var img = document.getElementById("scream");
     ctx.drawImage(img, 10, 10);
   }
  </script>
```

Scalable Vector Graphics (SVG)

- Gráficos de Rasterização vs. Gráficos Vetoriais
  - Grid de pixels vs. Comandos de desenho
- Por que usar gráficos vetoriais?



Scalable Vector Graphics (SVG)

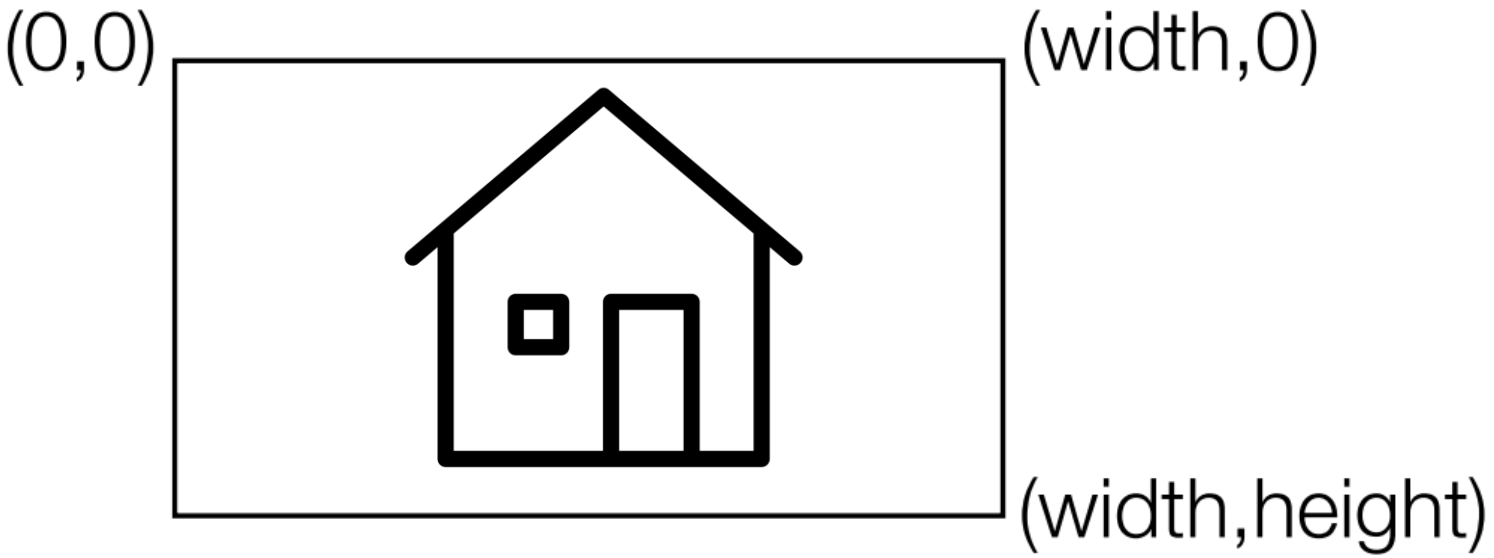
- Gráficos
  - Linhas, polígonos, figuras, texto, filtros, máscaras, efeitos
- Escaláveis
  - Zoom eficiente e rápido: amplia e reduz sem perder qualidade
- Vetoriais
  - Armazena as informações gráficas para desenhar a imagem (em vez de mapa de pixels).
  - Tamanho em bytes depende da complexidade gráfica
  - É XML! Objetos DOM são manipuláveis por CSS e scripts
  - Podem ser gerados/lidos por ferramentas de ilustração

Scalable Vector Graphics (SVG)

- Uma linguagem de marcação:
  - Descreve formas, pontos, cores, espessura...
  - **Exemplo 7 - Círculo**

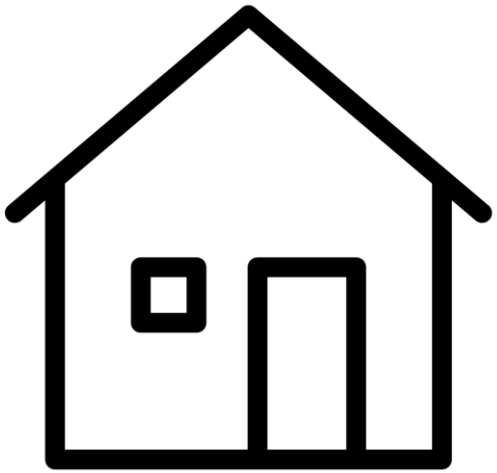
```
<svg width="200" height="200">
    <circle cx="100" cy="100" r="80" stroke="green" stroke-width="4" fill="yellow" />
</svg>
```

SVG Sistema de Coordenadas



Scalable Vector Graphics (SVG)

- Primitivas de Desenho:
  - Linhas, círculos, retângulos, elipses, textos, linhas poligonais, caminhos
- O princípio é descrever uma imagem através de informações sobre primitivas de desenho



Primitiva gráficas em SVG

- **<rect>** – Retângulo
- **<circle>** – Círculo
- **<ellipse>** – Elipse
- **<line>** – Linha reta
- **<polyline>** – Linha com múltiplos segmentos
- **<polygon>** – Polígono
- **<path>** - Caminho arbitrário (curvas, linhas, etc.)
- **<image>** - Imagem bitmap
- **<text>** - Texto

Preenchimento

- Dois atributos para preencher as primitivas:
  - Preenchimento (**fill**)
  - Contorno, ou traço (**stroke**) - desenhado pelo centro da borda do objeto
- Três tipos de "tinta"
  - cores sólidas (sRGB) - mesma especificação do CSS.
    - Ex: red, lightblue, #a09, #AA0099, rgb(128,64,32)
  - Gradientes
  - Texturas (patterns)

Preenchimento

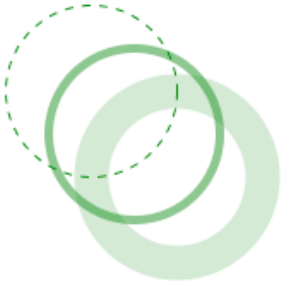
- Use fill (atributo ou CSS) para cor de preenchimento:
  - <rect ... fill="rgb(255,255,0%)"/>
  - <rect ... style="fill: rgb(100%,100%,0%)"/>

- Use fill-opacity para o componente alfa (transparência)
  - Varia de 0 (transparente) a 1 (opaco).

```
<circle r="50" cx="100" cy="100" fill="green"/>
<circle r="50" cx="125" cy="125" fill="#008000"
                                 fill-opacity="0.5"/>
<circle r="50" cx="150" cy="150" fill="#080"
                                 fill-opacity="0.2"/>
```

Contornos

- stroke: cor do traço
- stroke-width: espessura
- stroke-opacity: transparência (alfa)
- stroke-dasharray
  - Lista de valores para tracejado (sequência de traços e vazios)



```
<circle r="50" cx="100" cy="100" stroke="green" fill-opacity="0"
    stroke-width="1"  stroke-dasharray="5 5"/>
<circle r="50" cx="125" cy="125" stroke="green" fill-opacity="0"
    stroke-width="5"  stroke-opacity="0.5"/>
<circle r="50" cx="150" cy="150" stroke="green" fill-opacity="0"
    stroke-width="20" stroke-opacity="0.2"/>
```

Retângulo

Exemplo 8

- <svg width="400" height="100">
  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke width:10;stroke:rgb(0,0,0)" />
  </svg>

Elipse

Exemplo 9

- <svg width="400" height="200">
  <ellipse cx="200" cy="100" rx="100" ry="50" style="fill:red; stroke:black; stroke-width:5px" />
  </svg>

Linhas e Texto

Exemplo 10

```
<svg width="400" height="200">
  <line x1="30" y1="30" x2="200" y2="80" style="stroke:red" />
  <line x1="30" y1="50" x2="150" y2="120" style="stroke:red" />
</svg>

<svg width="400" height="100">
  <text x="60" y="60">Computação Gráfica</text>
  <text x="60" y="100">Primitivas Gráficas</text>
```

Caminhos

- Sequências de comandos (letras) + coordenadas
  - Ex: **M** 50,50 **L** 120,120 **z**
  - Comando afeta coordenadas seguintes (até novo comando)
  - Maiúsculas = coords absolutas / Minúsculas = relativas
- Quatro tipos de movimentos
  - **M**: move até um ponto sem desenhar
  - **L, H, V**: desenha linha reta até um ponto
  - **C, S, Q, T, A** (*curve to*): desenha curva a um ponto; pode ser:
  - Bézier cúbica com dois pontos tangenciais (**C, c, S, s**)
  - Bézier quadrática com um ponto tangencial (Q, q, T, t)
  - Arco elíptico ou circular (A, a)
- Z: fecha a figura

Caminhos

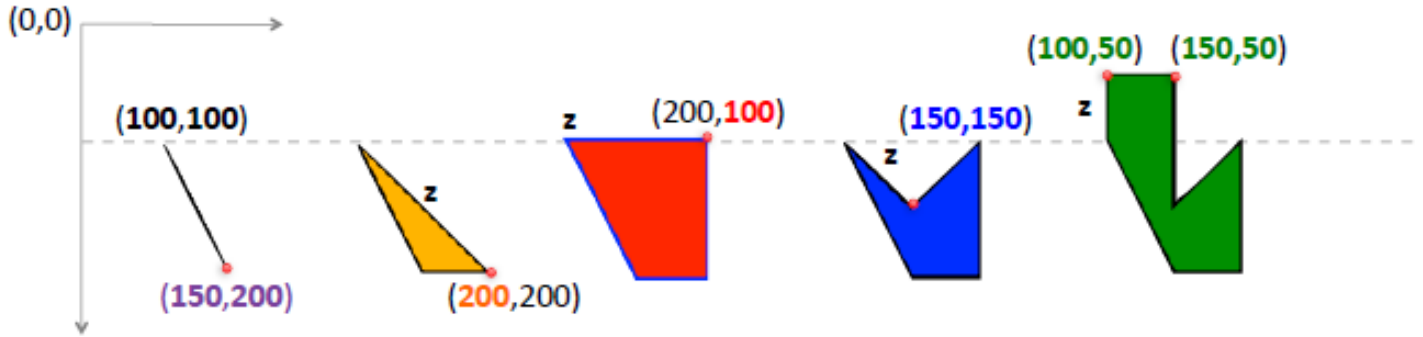
Exemplo 11

```
<svg width="200" height="60" style="stroke:blue; fill:none">
        <path d="M 10 10 L 50 10 L 50 50 L 100 50 L 100 10 C
150 50 150 50 150 10"/>
</svg>
```

Caminhos – Linhas Retas

- **H** ou **h** + coordenadas x linhas retas horizontais
- **V** ou **v** + coordenadas y linhas retas verticais
- **L** ou **l** + pares de coords x,y linhas retas em qq direção

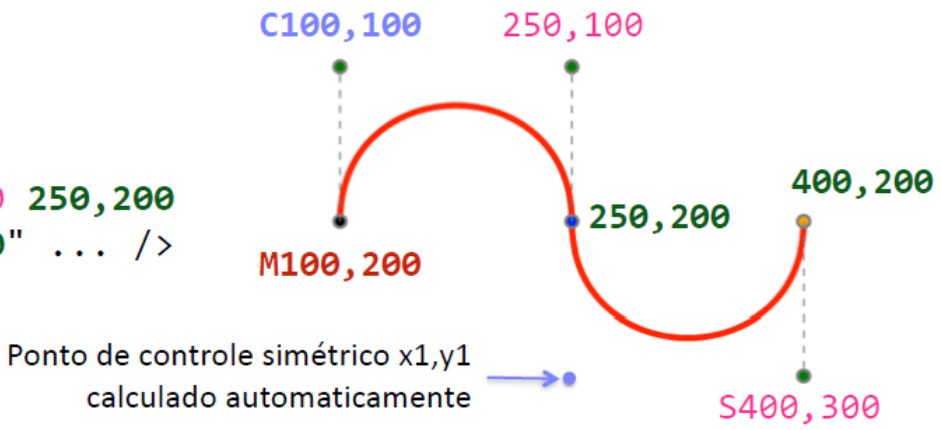
1. `M100,100 L150,200 z`
2. `M100,100 L150,200 h50 z`
3. `M100,100 L150,200 h50 v-100 z`
4. `M100,100 L150,200 h50 v-100 l-50,50 z`
5. `M100,100 L150,200 h50 v-100 l-50,50 L150,50 100,50 z`



Curvas Bézier **cúbicas**

- Curva normal: **C** ou **c** com coordenadas x1,y1 x2,y2 x,y onde:
  - **x,y** são as coordenadas do ponto final da curva
  - **x1,y1** são as coordenadas do controle tangencial do ponto inicial
  - **x2,y2** são as coordenadas do controle tangencial do ponto final
- Curva suave: S ou s com coordenadas x2,y2 x,y
  - O controle tangencial do ponto inicial será simétrico ao controle final da curva anterior, fazendo com que a junção seja **suave**

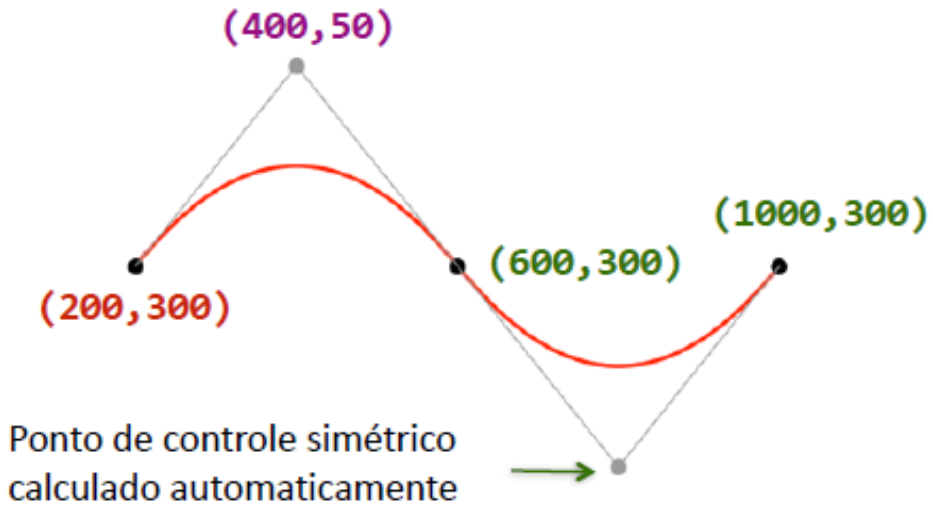
```
<path
 d="M100,200
    C100,100 250,100 250,200
    S400,300 400,200" ... />
```



Curvas Bézier **Quadráticas**

- Curva normal: **Q** ou **q** com coordenadas x1,y1 x,y onde:
  - **x,y** são as coordenadas do ponto final da curva
  - **x1,y1** são as coordenadas do controle tangencial
- Curva suave: T ou t com coordenadas x,y
  - O controle tangencial será simétrico ao controle tangencial da curva anterior, fazendo com que a junção seja **suave**

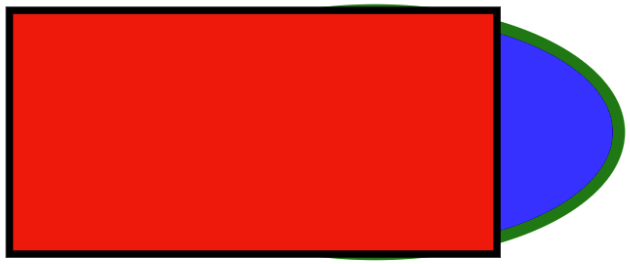
```
<path
 d="M 200,300
    Q 400,50 600,300
    T 1000,300" ... />
```



Ordenação **Exemplo 12**

- O que acontece quando dois elementos se interceptam?

```
<svg width="400" height="200">
<ellipse cx="200" cy="100" rx="100" ry="50"
style="fill:blue; stroke:green; stroke-width:5px"/>
<rect x="50" y="50" width="200" height="100"
style="fill:red; stroke:black; stroke-width:3px"/>
</svg>
```



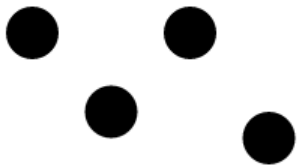
Grupos em SVG

- Mecanismo para organização de elementos svg
- Servem para aplicar operações em múltiplos elementos

Grupos em SVG

Exemplo 13

```
<svg width="200" height="200">
    <g>
        <circle cx="50" cy="50" r="10"/>
        <circle cx="80" cy="80" r="10"/>
        <circle cx="110" cy="50" r="10"/>
        <circle cx="140" cy="90" r="10"/>
    </g>
</svg>
```



Transformações em SVG

Translation

- Mover elemento pro ponto *translate(x,y)*

Rotation

- Rotacionar um elemento em *rotate(graus)*

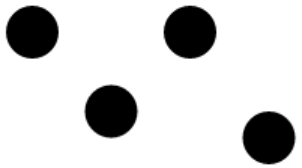
Scaling

- Mudar o tamanho de um elemento em *scale(x,y)*

Transformações em SVG

Exemplo 14

```
<svg width="200" height="200">
        <g transform="translate(0, 200) scale(1, -1)">
        <circle cx="50" cy="50" r="10"/>
        <circle cx="80" cy="80" r="10"/>
        <circle cx="110" cy="50" r="10"/>
        <circle cx="140" cy="90" r="10"/>
        </g>
</svg>
```



Transformações em SVG

Exemplo 15

```
<svg width="400" height="400">
   <g transform="translate(300,0) scale(-1, 1)">
     <text x="10" y="20">Computação Gráfica</text>
     <text x="150" y="50">Primitivas Gréficas</text>
     <text x="40" y="80">no</text>
     <text x="100" y="90">HTML</text>
   </g>
 </svg>
```

Incluindo arquivos SVG externos

- Como referenciar arquivos SVG externos

Exemplo 16

```
<h1>Incluindo SVG</h1>
  <p>Como imagem</p>
  <img width="400" height="400" src="img2.svg" />
  <p>Como objeto</p>
  <object width="400" height="400" data="img3.svg"></object>
  <p>Como Embed</p>
  <embed width="400" height="400" src="img4.svg" />
```

Diferenças entre <svg> e <canvas>

- Tanto <svg> quanto <canvas> criam contextos gráficos dentro do navegador. A diferença primordial entre eles é que o contexto de <svg> é vetorial, enquanto o de <canvas> é bitmap.

- A segunda diferença é que o contexto <svg> possui um conjunto de instruções próprias, estruturadas em formato e sintaxe XML (Extensible Markup Language) para ser manipulado, enquanto <canvas> é manipulado apenas por programação Javascript.

Referências desta aula

- https://www.devmedia.com.br/html5-a-tag-canvas/25413. Acesso em 16/09/2022
- https://www.w3schools.com/html/html5_svg.asp. Acesso em 16/09/2022
- https://www.ranoya.com/books/public/html/tag-svg.php?embed=plain. Acesso em 16/09/2022
- AZEVEDO, Eduardo; CONCI, Aura. 2007. Computação Gráfica: Teoria e Prática. Elsevier, Vol. 2, 2007.
- Aula montada com base no material do Prof. Jorge Cavalcanti - UNIVASF.