.:: Canvas HTML5 :: Criando um Jogo ::.

Agora que já conhecemos o componente CANVAS do HTML5, vamos montar uma aplicação usando este componente.

A nossa aplicação será um jogo onde temos que perseguir um inimigo em um campo de batalha simples. A primeira coisa que faremos é copiar algumas imagens para uma pasta de nome img dentro de nossa pasta de trabalho.

Copie então a pasta img da pública para a sua área, dentro da pasta de desenvolvimento do jogo. Lá teremos as imagens que utilizaremos no desenvolvimento do nosso jogo.

Copiada a pasta das imagens vamos desenvolver o nosso arquivo HTML que conterá o componente CANVAS a ser manipulado. A nossa página terá a seguinte estrutura básica:

```
<!DOCTYPE html>
<html>
  <head>
      <title>Animação</title>

      <script language="JavaScript" type="text/javascript">
      <!--

      -->
      </script>

   </head>

   <body>
   </body>
</html>
```

Agora que já temos a estrutura básica do nosso arquivo, vamos colocar o componente CANVAS nele. No **BODY**, coloque o seguinte código:

```
<canvas id="meuCanvas" width="512"  height="480"
                style="border:1px solid #000000;">
     Seu browser não suporta o elemento CANVAS, atualize-se!!!
</canvas>
```

Sempre que quisermos referenciar o nosso CANVAS, utilizaremos o ID meuCanvas.

O campo de batalha terá 512 pixels de largura e 480 pixels de altura, o que se refere à altura e largura da imagem que vamos carregar no canvas. Configuramos ainda uma borda sólida de 1px de largura e caso queira mudar a cor da borda basta alterar a propriedade referente no atributo **style**. O Texto "Seu browser não suporta o elemento CANVAS, atualize-se!!!" aparecerá apenas se o seu browser não suportar HTML5.

Execute a página HTML e veja que vai aparecer apenas um retângulo com as dimensões indicadas. Este é o canvas !!

O próximo passo é carregarmos a imagem de fundo que representa o campo de batalha do nosso jogo. Faremos isso da seguinte forma:

Na área de desenvolvimento de Script, vamos declarar duas variáveis que na verdade serão objetos. Uma delas representará o nosso canvas e outra representará o contexto do canvas e está será a variável de trabalho sobre a qual vamos desenhar os nossos personagens e o campo de batalha.

```
var objCanvas=null;     // objeto que representa o canvas
var objContexto=null;   // objeto que representa o
                        // Contexto do canvas
```

Vamos criar também um Objeto do tipo Image para representar a imagem de fundo do nosso jogo, o campo de batalha.

```
// Objetos Image para cada coisa que vai aparecer na tela
var imgFundo = new Image();
imgFundo.src = "img/fundo.png";
```

Criamos um objeto imagem usando o comando new Image() onde new é o comando que cria uma instancia de um Objeto e Image() é justamente o objeto do qual queremos criar uma instancia. Os conceitos de Objeto e Instância serão melhor estudados na disciplina de Orientação a Objetos.

O objeto imgFundo representa uma imagem e este objeto possui uma propriedade de nome src onde informamos o nome da imagem a ser utilizada como imagem deste objeto. No nosso caso, a imagem de nome fundo.png deve estar dentro de uma pasta de nome img na mesma pasta onde está o arquivo HTML que você está desenvolvendo.

Até aqui, apenas criamos as variáveis ou objetos de controle e isso pode ser feito na área de desenvolvimento de script sem necessidade de criarmos nenhuma função. O que faremos a seguir, será o desenvolvimento de uma função que vai preencher os objetos e iniciar o desenho. Para isso faremos uma função de nome **Iniciar().**

```
function Iniciar(){
}
```

Vamos então colocar valores nos dois objetos que criamos inicialmente dentro dessa função. O objCanvas representará o nosso componente Canvas, então vamos acessá-lo através do getElementById. Após preencher o objCanvas, vamos acessar o seu contexto 2D, que será a área onde colocaremos o nosso campo de batalha e depois o nosso herói e o nosso vilão.

```
objCanvas   = document.getElementById("meuCanvas");
objContexto = objCanvas.getContext("2d");
```

Perceba que “meuCanvas” é o ID do componente Canvas no HTML.

Até aqui ainda não temos nenhum efeito visual, podemos atualizar a nossa página que nada de diferente acontecerá.

Vamos agora desenhar no contexto do Canvas, uma imagem que representará o fundo do jogo, o campo de batalha. Para isso, utilizaremos a função drawImage() do contexto para

desenharmos uma imagem. A função drawImage tem três parâmetros. O primeiro é a Imagem a ser carregada, o segundo é a coordenada X onde a imagem deve ser desenhada e o terceiro é a coordenada Y onde a imagem deve ser desenhada. A imagem deve ser um objeto Image, então a primeira coisa a fazer é criar esse objeto e faze-lo referenciar a imagem de fundo.

Agora que temos o objeto Image, podemos desenhá-lo como fundo do nosso canvas utilizando-o como primeiro parâmetro da função drawImage aplicada sobre o contexto do nosso canvas:

```
objContexto.drawImage(imgFundo,0,0);
```

Para que esses comandos que estão dentro da função tenham efeito, vamos executar a função no Load no nosso arquivo HTML.

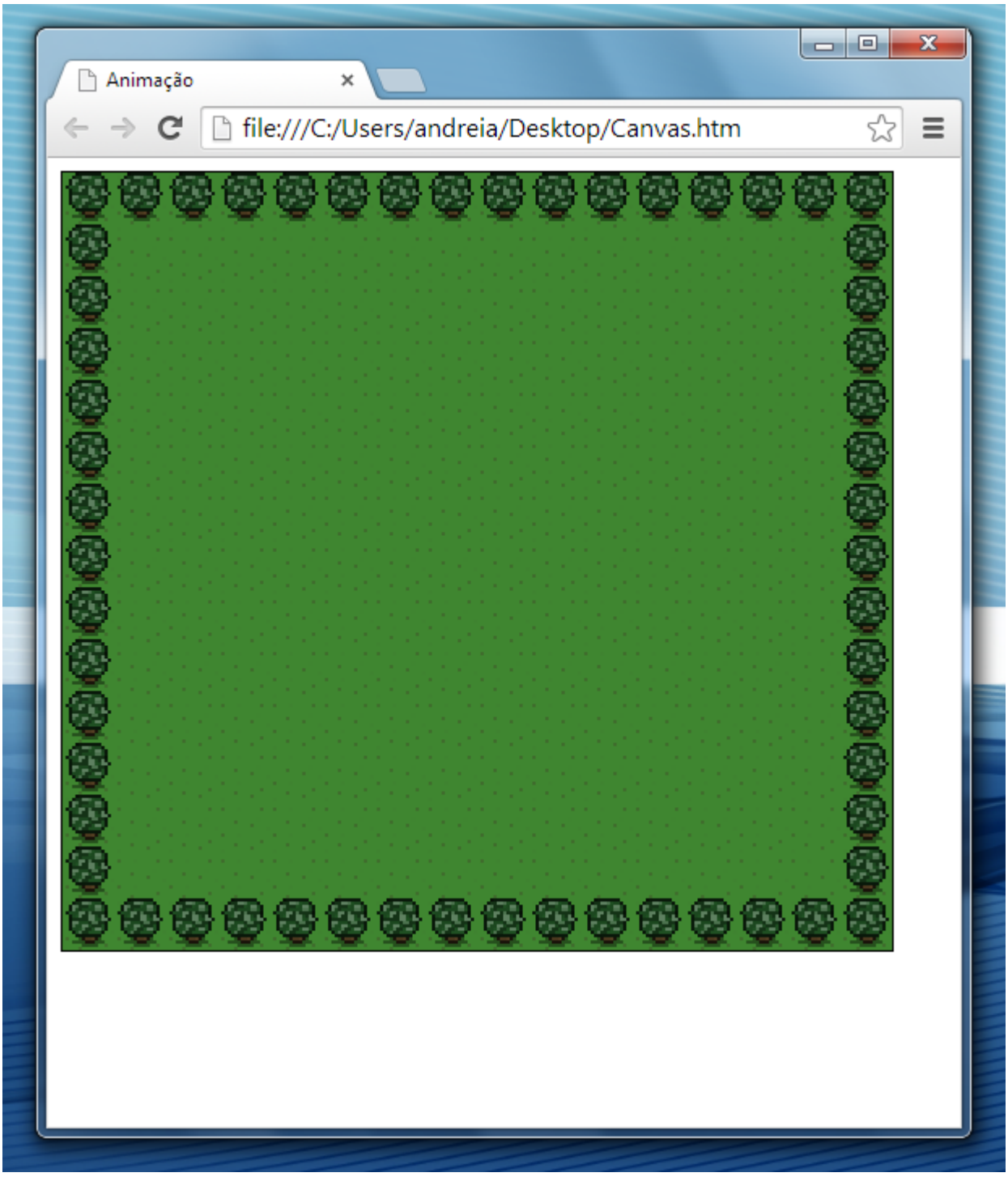
Neste caso, vamos desenhar a imagem a partir da posição 0,0 do nosso canvas, assim ela vai cobrir toda a extensão do canvas, pois tem as mesmas dimensões deste.

O código completo até o momento fica da seguinte forma:

```
<!DOCTYPE HTML>

<html>
<head>
<title>Animação</title>

<script language="JavaScript" type="text/javascript">
<!--
     var objCanvas=null;    // objeto que representa o canvas
     var objContexto=null;  // objeto que representa o

     // Objetos Image para cada coisa que vai aparecer na tela
     var imgFundo = new Image();
     imgFundo.src = "img/fundo.png";

     function Iniciar(){
          objCanvas   = document.getElementById("meuCanvas");
          objContexto = objCanvas.getContext("2d");

          objContexto.drawImage(imgFundo,0,0);
    }
-->
</script>

</head>

<body onLoad="Iniciar();">
     <canvas id="meuCanvas" width="512"
             height="480"
             style="border:1px solid #000000;">
          Seu browser não suporta o elemento CANVAS, atualize-se!!!
     </canvas><BR>
```

```
</body>
</html>
```

Atualize a página e veja o resultado.



Chegamos agora no ponto de colocarmos o nosso herói e o nosso mostro no campo de batalha. Como ambos os personagens irão se mexer pelo fundo, precisaremos de variáveis para representar as coordenadas X e Y de cada um deles, que serão criadas junto com os nossos objetos na área de script, fora de qualquer função.

```
// Controle de Posicionamento do Heroi
var xHeroi=0;
var yHeroi=0;

// Controle de Posicionamento do Monstro
var xMonstro=0;
var yMonstro=0;
```

Criaremos também nessa área os objetos Image() para cada um dos personagens.

```
var imgHeroi = new Image();
imgHeroi.src = "img/heroi.png";

var imgMonstro = new Image();
imgMonstro.src = "img/monstro.png";
```

Mais uma vez até aqui não temos nenhum efeito visual adicional, estamos apenas criando variáveis e colocando conteúdo nelas.

Neste ponto precisamos pensar um pouco sobre a sequencia de passos e um jogo. A primeira coisa a ser feita é Montar o jogo. Depois, colocamos os personagens e objetos de interação no jogo, depois esperamos a interação do usuário e por fim redesenhamos o jogo com os efeitos da interação do usuário. Bem resumidamente temos o seguinte enlace:

```
Repita
     DesenheOFundo()
     DesenheOsObjetosFixos()
     DesenheOsPersonagens()
     UsuarioRealizaAcao()
     PersonagensAutomaticoRealizaAcão()
Até FinalizarOJogo;
```

No nosso jogo, não temos um comando de repetição que nos prenda, pois não estamos em um ambiente de programação sequencial e sim em um ambiente que responde a eventos. Vamos então organizar o nosso código em funções que permitam realizar as ações acima sem pensar em uma sequencia, apenas nas funções.

Vamos criar uma função para Atualizar a Tela onde desenharemos (ou redesenharemos) a imagem de fundo e os nossos personagens com as suas coordenadas atualizadas. Ela ficará da seguinte forma:

```
function AtualizaTela(){
     objContexto.drawImage(imgFundo,0,0);
     objContexto.drawImage(imgHeroi, xHeroi, yHeroi);
     objContexto.drawImage(imgMonstro, xMonstro, yMonstro);

}
```

A cada interação do usuário apenas alteramos o xHeroi e o yHeroi que ao redesenhar a tela teremos a impressão que o personagem se moveu na tela. A mesma coisa para o monstro, apenas que este terá, a princípio, movimentos automáticos.

Como agora temos uma função de atualização de tela, ela deve ser chamada ao final da função iniciar. Como dentro da nossa AtualizaTela já desenhamos o fundo, devemos retirar o comando que desenha o fundo da função Iniciar();

Nosso código fica assim:

```
<!DOCTYPE HTML>

<html>
<head>
<title>Animação</title>

<script language="JavaScript" type="text/javascript">
<!--
     var objCanvas=null;    // objeto que representa o canvas
     var objContexto=null;  // objeto que representa o

     // Controle de Posicionamento do Heroi
     var xHeroi=0;
     var yHeroi=0;

     // Controle de Posicionamento do Monstro
     var xMonstro=0;
     var yMonstro=0;

     // Objetos Image para cada coisa que vai aparecer na tela
     var imgFundo = new Image();
     imgFundo.src = "img/fundo.png";

     var imgHeroi = new Image();
     imgHeroi.src = "img/heroi.png";

     var imgMonstro = new Image();
     imgMonstro.src = "img/monstro.png";


     function AtualizaTela(){
          objContexto.drawImage(imgFundo,0,0);
          objContexto.drawImage(imgHeroi, xHeroi, yHeroi);
          objContexto.drawImage(imgMonstro, xMonstro, yMonstro);

     }


     function Iniciar(){
          objCanvas   = document.getElementById("meuCanvas");
          objContexto = objCanvas.getContext("2d");

          AtualizaTela();
```

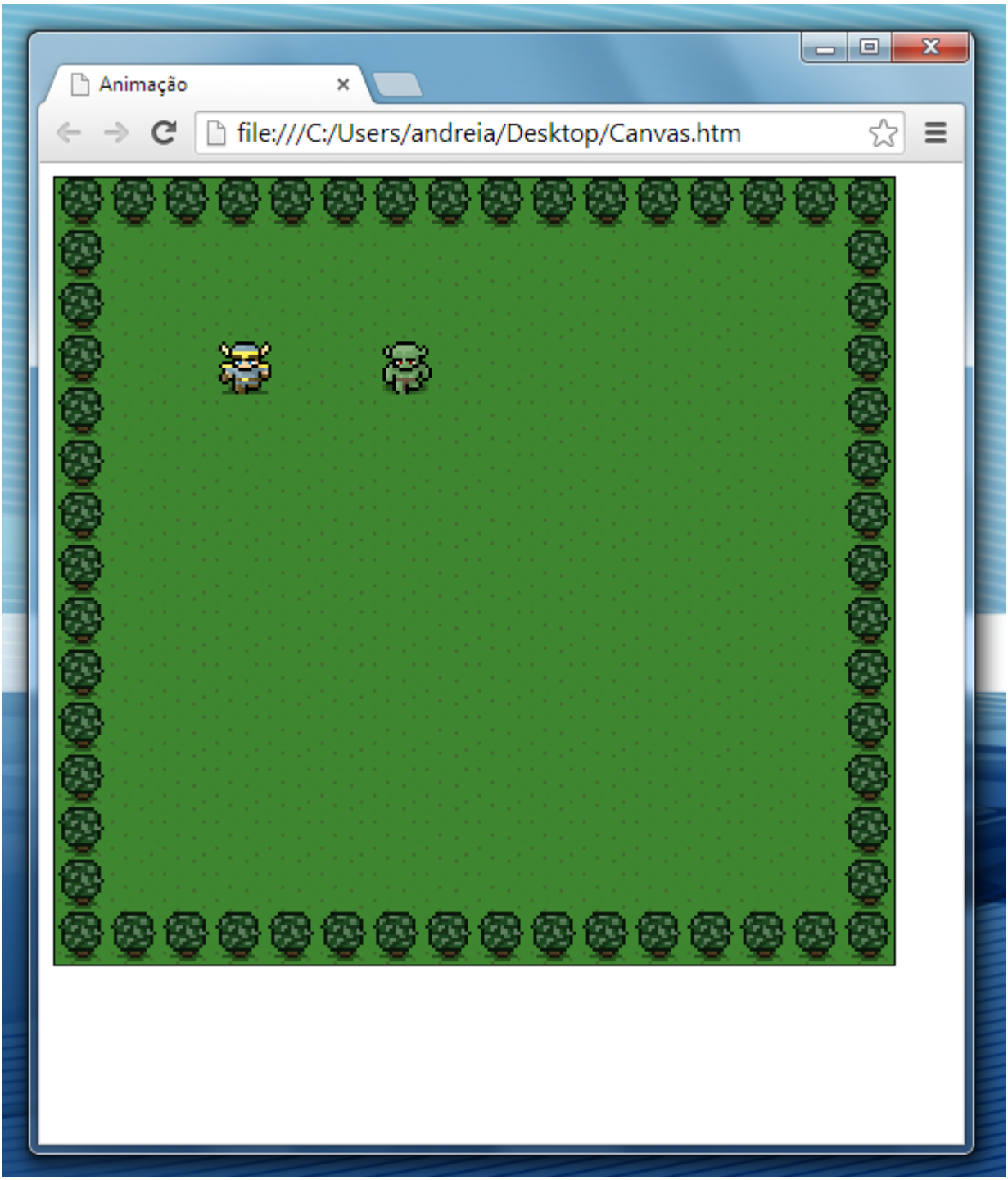
```
      }

-->
</script>

</head>
<body onLoad="Iniciar();">
<canvas id="meuCanvas" width="512"
        height="480"
        style="border:1px solid #000000;">
      Seu browser não suporta o elemento CANVAS, atualize-se!!!
</canvas><BR>


</body>
</html>
```

Atualize a página e veja que o herói e o monstro aparecem no canto superior da tela. Talvez você não veja o Herói pois o Monstro o sobrepõe, ambos estão como o mesmo X e Y!! Altere o xHeroi e o xMonstro e veja onde eles aparecem!

Alterando as variáveis de posicionamento como abaixo

```
// Controle de Posicionamento do Heroi
var xHeroi=100;
var yHeroi=100;

// Controle de Posicionamento do Monstro
var xMonstro=200;
var yMonstro=100;
```

...teremos o seguinte resultado:



Pronto!! Agora temos todos os elementos em tela!!!

Basta-nos agora programação a interação entre eles.

Fazendo o Monstro se mover

Para simular o movimento do monstro, vamos criar uma função de nome MovimentoDoMonstro() onde faremos a alteração das coordenadas de X e/ou Y do monstro de forma aleatória.

A primeira coisa a fazer é verificar para qual direção o mostro deve se mover. Vamos usar a função random da biblioteca Math do javaScript para sortear um número de 0 a 3. Vamos definir que os seguintes movimentos de acordo com o valor gerado:

- se o número gerado for 0, o monstro vai se mover no eixo X para a direita.
- se o número gerado for 1, o monstro vai se mover no eixo X para a esquerda.
- se o número gerado for 2, o monstro vai se mover no eixo Y para a direita.
- se o número gerado for 3, o monstro vai se mover no eixo Y para a esquerda.

Para gerar valores de 0 a 3 vamos usar a seguinte forma:

```
Math.random()*10 % 4
```

A função Math.random() gera números entre 0 e 1. Ao multiplicarmos esse número por 10, estamos com números entre 0 e 10. Ao pegar o resto da divisão desse número por 4, restringimos os valores entre 0 e 3.

Façamos uma simulação. Imagine que o número gerado pelo random foi 0,976. Ao multiplicarmos por 10 teremos o número 9,76. Ao fazermos a divisão inteira por 4 e pegarmos o resto teremos o valor 1, pois 9 dividido por 4 dá 2 e sobra 1 como resto. Podemos usar essa formula sempre que quisermos restringir uma faixa de valores a serem gerados.

Faremos então a nossa função da seguinte forma:

```
function MovimentoDoMonstro (){
     // sorteia uma direção para a qual o monstro vai se dirigir
     var direcao = parseInt((Math.random()*10) % 4);

     switch (direcao){
         case 0:xMonstro=(xMonstro + 10); break;
         case 1:yMonstro=(yMonstro + 10); break;
         case 2:xMonstro=(xMonstro - 10); break;
         case 3:yMonstro=(yMonstro - 10); break;
     }
     atualizaTela();
}
```

Ao final da alteração do posicionamento, vamos chamar a nossa função de atualização de Tela para que o monstro possa aparecer em sua nova coordenada.

Para testar, coloque um botão na tela e ao seu click (onClick) faça a chamada da função MovimentoDoMontro().

Próximos Passos

Agora basta implementar a movimentação do Herói.

Como primeiro passo, crie quatro botões que controlem a movimentação. Depois, tente implementar a movimentação via teclado com as setas.

Agora precisamos limitar a área de movimentação tanto do monstro como do herói para que ele não passe sobre as árvores e nem saia da área de visão do jogo.

O próximo passo é implementar a perseguição e controlar a captura do monstro para calcular a pontuação e a finalização do jogo.

Mãos a Obra!!!!