INSTITUTO FEDERAL
SANTA CATARINA

Construindo um Jogo para a Web - *Snake*

Programação para a Internet

Prof. Vilson Heck Junior

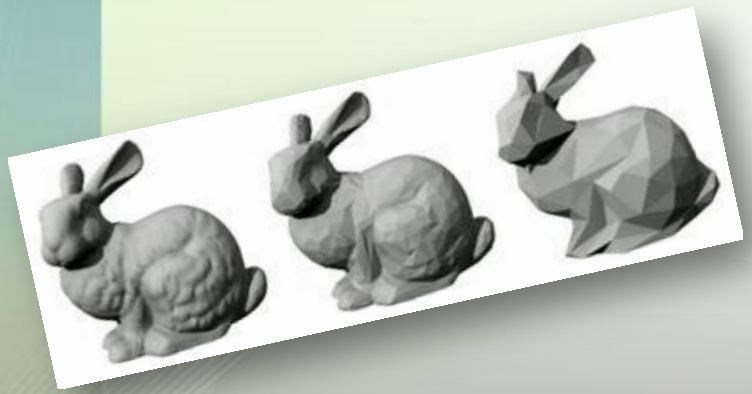
GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA

Tecnologias Necessárias

- Tecnologias já Estudadas:
  - HTML;
  - CSS;
  - JavaScript;
- **Tecnologias Novas:**
  - **Computação Gráfica Básica;**
  - **Noções de Geometria;**
  - **Noções de Física;**
  - **Reprodução de Sons;**
  - **Enredo;**

Computação Gráfica

- É um campo da Ciência da Computação que estuda métodos para sintetizar e manipular digitalmente conteúdo visual:
  - Geração de imagens 2D;
  - Geração de imagens 3D (renderização);
  - Com ou sem animação;



Noções de Geometria

- Gráficos 2D ou 3D são na verdade a composição de pequenas peças geométricas:
- A relação espacial dada entre diferentes objetos existentes em uma cena deve ser respeitada:
  - Dois corpos não podem ocupar um mesmo lugar no espaço!

Noções de Física

- Objetos podem possuir algum tipo de movimento ou interação com outros objetos;
- Para isto, geralmente respeitam alguma(s) regras físicas:
  - Próximas a real: Simulação;
  - Diferenciadas: Arcade;

Reprodução de Sons

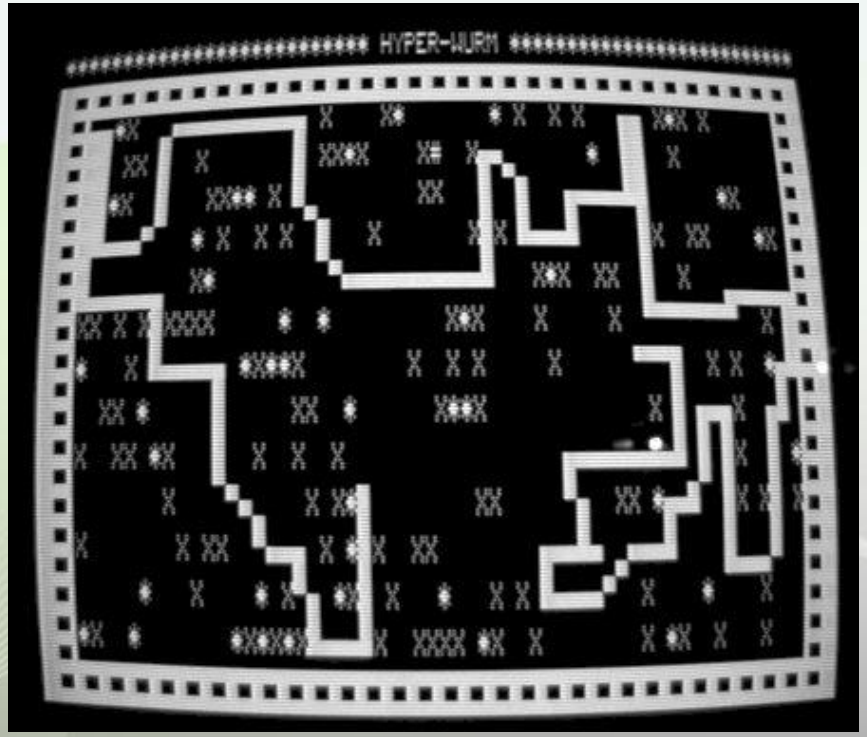
- O som é o elemento responsável por estimular o sentido da audição;
- Não tanto quanto os gráficos, mas os sons são responsáveis por completar uma boa sensação de imersão em jogos e entretenimento;
- Geralmente os sons (músicas ou barulhos) serão escolhidos conforme um determinado contexto ou acontecimento.

Enredo

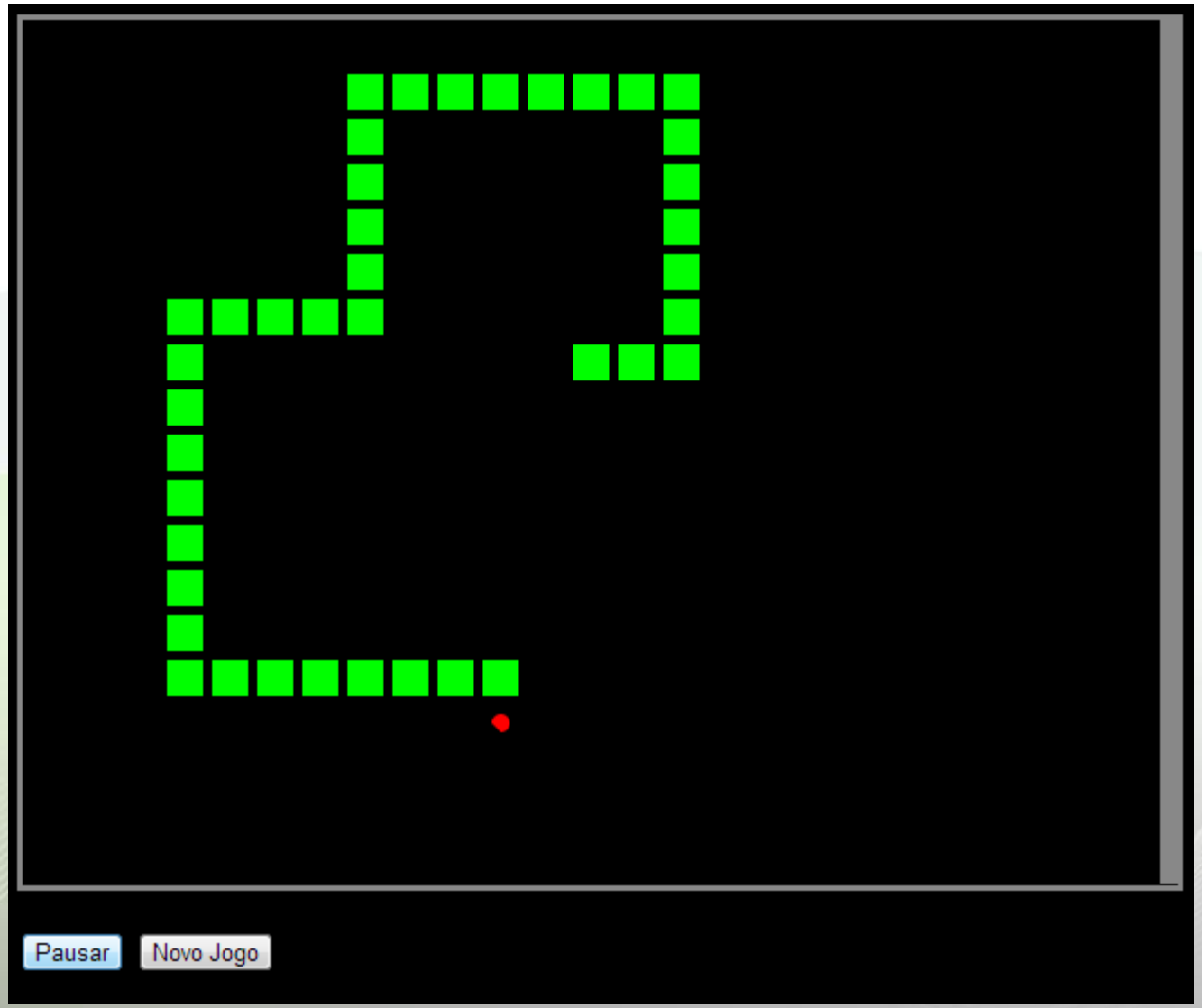
- O enredo irá explicar ao usuário o que deverá ser feito e deve ser o principal responsável por atrair a atenção do jogador:
  - História;
  - Diversão;
  - Desafios;
  - Passatempo;
  - ...

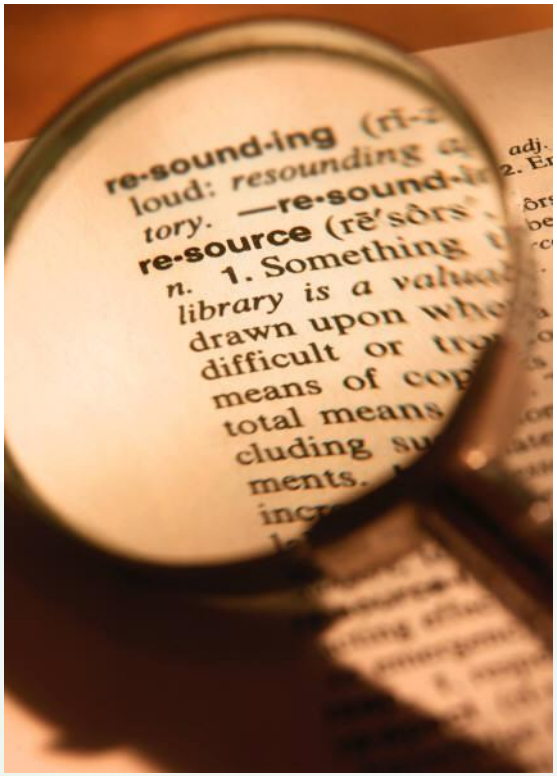
Enredo

- Snake:
  - Fliperama feito pela Gremlin em 1976.
  - Popularizado nos celulares Nokia em 1998.



Nosso Conceito





Snake

LISTA DE RECURSOS INICIAIS

Recursos Iniciais

- Pasta: “Snake”:
  - index.html
    - Construiremos de um documento web, inserindo todos os demais elementos necessários;
  - estilo.css
    - Definiremos algumas configurações de cores, bordas e outros para nossa interface;
  - script.js
    - Faremos todo o processamento do jogo, ou seja, daremos vida aos elementos existentes no documento web.
  - Nodo.js
    - Implementaremos uma Classe Nodo, para as partes da Snake;

index.html

- Crie o arquivo como **doctype** para **html 5**;
- Crie as tags para:
  - **<html>**, **<head>**, **<body>** e **<title>**;
- Estipule um **<link>** com arquivo de estilo;
- Adicione ambos arquivos de **<script>** ao fim do <body>;
  - ***Importante: adicionar Nodo.js antes de script.js, pois o último precisa do primeiro.***

index.html

- Adicione os seguintes Tags com seus atributos dentro do <body>:
  - <canvas>Navegador não suportado!</canvas>
    - id = “tela” width=640 height=480
  - <button>Iniciar</button>
    - type=“button” onClick=“pausa()” id=“btPausa”
  - <button>Novo Jogo</button>
    - type=“button” onClick=“novoJogo()” id=“btNovo”

Snake

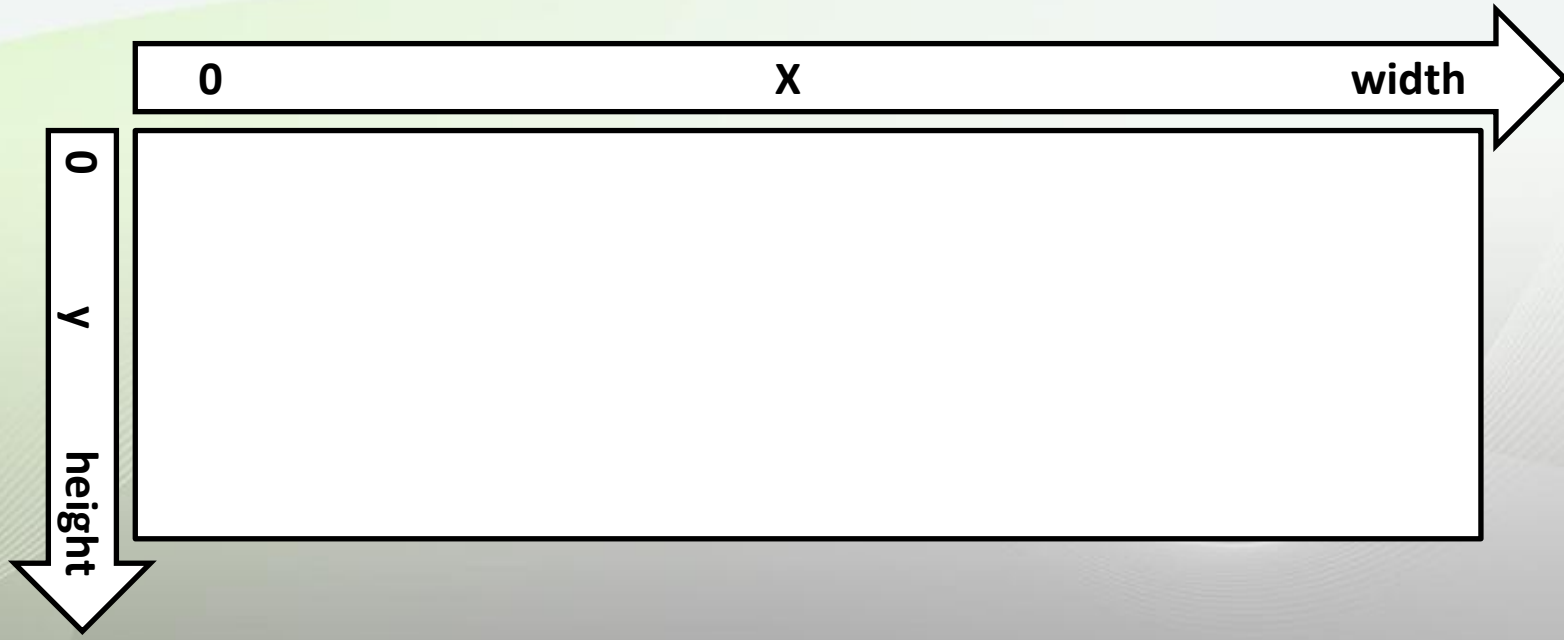
DESENHANDO NO CANVAS

<canvas>

- Canvas é um termo inglês dado a alguns tipos de tela para pintura;
- No nosso caso, será uma área dentro do documento HTML onde poderemos pintar o que precisarmos;
- Nosso pincel e paleta de cores estão disponíveis através de código JavaScript.

<canvas>

- O Canvas é feito para oferecer suporte a rápido desenho de cenas bidimensionais ou tridimensionais:
  - Geralmente acelerado por Hardware;

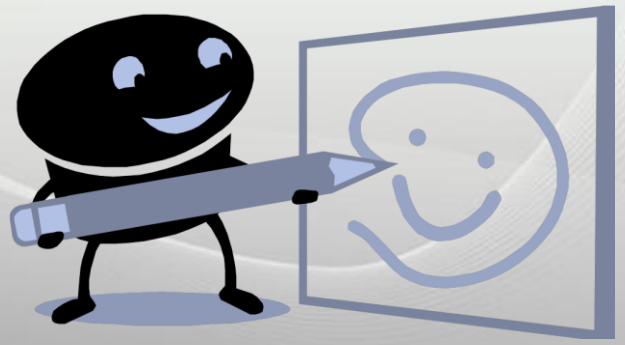


script.js

```
//Recuperando referência dos objetos no documento
var canvas = document.getElementById("tela");
var context = canvas.getContext("2d");
var btPausa = document.getElementById("btPausa");

//Um pequeno teste (remover depois de testar)
context.fillStyle = “#FF0000”; //Usar cor vermelha
context.fillRect(20, 30, 50, 100); //x=20, y=30, w=50 e h=100
```

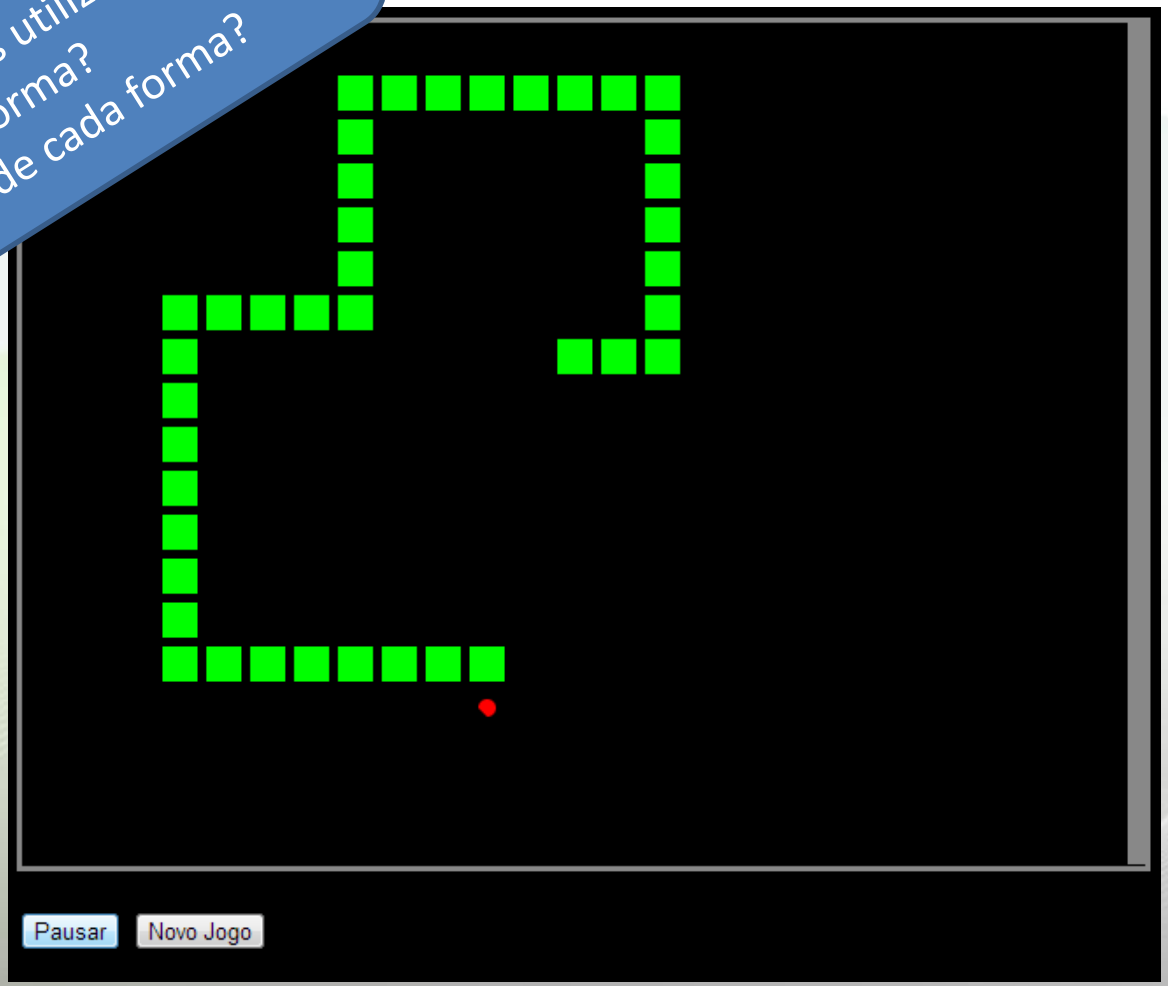
Desenhando

- Temos uma tela para desenho;
- Conhecemos uma primeira ferramenta para pintar algo;
- Temos que utilizar esta ferramenta de forma a construir o cenário inicial do nosso jogo;



Relembrando

- Quais são as formas geométricas utilizadas?
- Qual é a quantidade de cada forma?
- Qual é a posição e tamanho de cada forma?



Posicionamento e Tamanhos

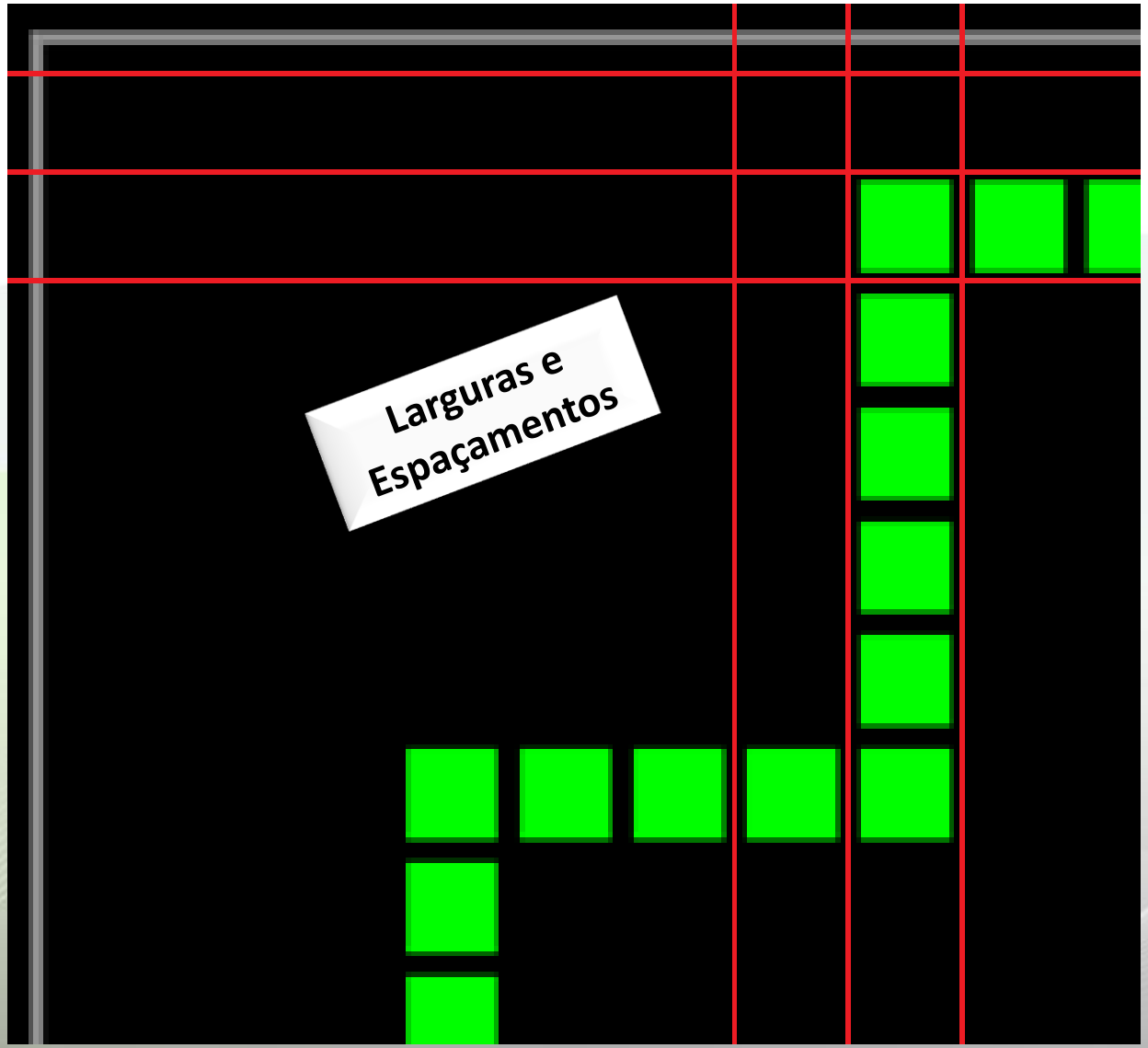
- Encontramos: Diversos pedaços quadrados da Snake e um circulo que representa a Fruta;

- Iremos utilizar variáveis para guardar a posição (X, Y) de cada objeto, bem como as dimensões e outras propriedades das formas geométricas utilizadas.

Nodo.js

```
//Sentidos de movimento
var dcima = 1;
var ddireita = 2;
var dbaixo = 3;
var desquerda = 4;

function Nodo(px, py, dir) {
   var x, y, direc;
   this.x = px;
   this.y = py;
   this.direc = dir;
   this.Mover = function() {
      switch (this.direc) {
      case dcima:
         this.y -= 1;
         break;
      case dbaixo:
         this.y += 1;
         break;
      case ddireita:
         this.x += 1;
         break;
      case desquerda:
         this.x -= 1;
         break;
      }
   };
}
```

Grade



script.js

```
//Informações sobre o tabuleiro
var nx = 0; //Número de quadros em X
var ny = 0; //Número de quadros em Y
var largura = 20; //Largura dos quadros
var distancia = 5; //Distância entre os quadros
var borda_x, borda_y; //Posições das bordas

//Inicializações
criarTabuleiro();
novoJogo();

function criarTabuleiro() {
    nx = Math.floor((canvas.width - distancia) / (largura + distancia));
    ny = Math.floor((canvas.height - distancia) / (largura + distancia));
    borda_x = nx * (distancia + largura) + distancia;
    borda_y = ny * (distancia + largura) + distancia;
}
```

script.js

```
function novoJogo() {
   btPausa.innerHTML = "Iniciar";
   btPausa.disabled = false;
   desenhar();
}

function desenhar() {
   //Variáveis auxiliares para desenhar
   var xi, yi;
   //Limpar a tela
   context.clearRect(0, 0, canvas.width, canvas.height);
   //Desenhar bordas
   context.fillStyle = "#888888";
   context.fillRect(borda_x, 0, canvas.width - 1, canvas.height - 1);
   context.fillRect(0, borda_y, canvas.width - 1, canvas.height - 1);
}
```

estilo.css

```
#tela
{
   background-color: #000000;
   border-style: solid;
   border-color: #888888;
}

body
{
   background-color: #000000;
}
```

Área do jogo!

script.js

Inserindo a cobra (no bom sentido, é claro)

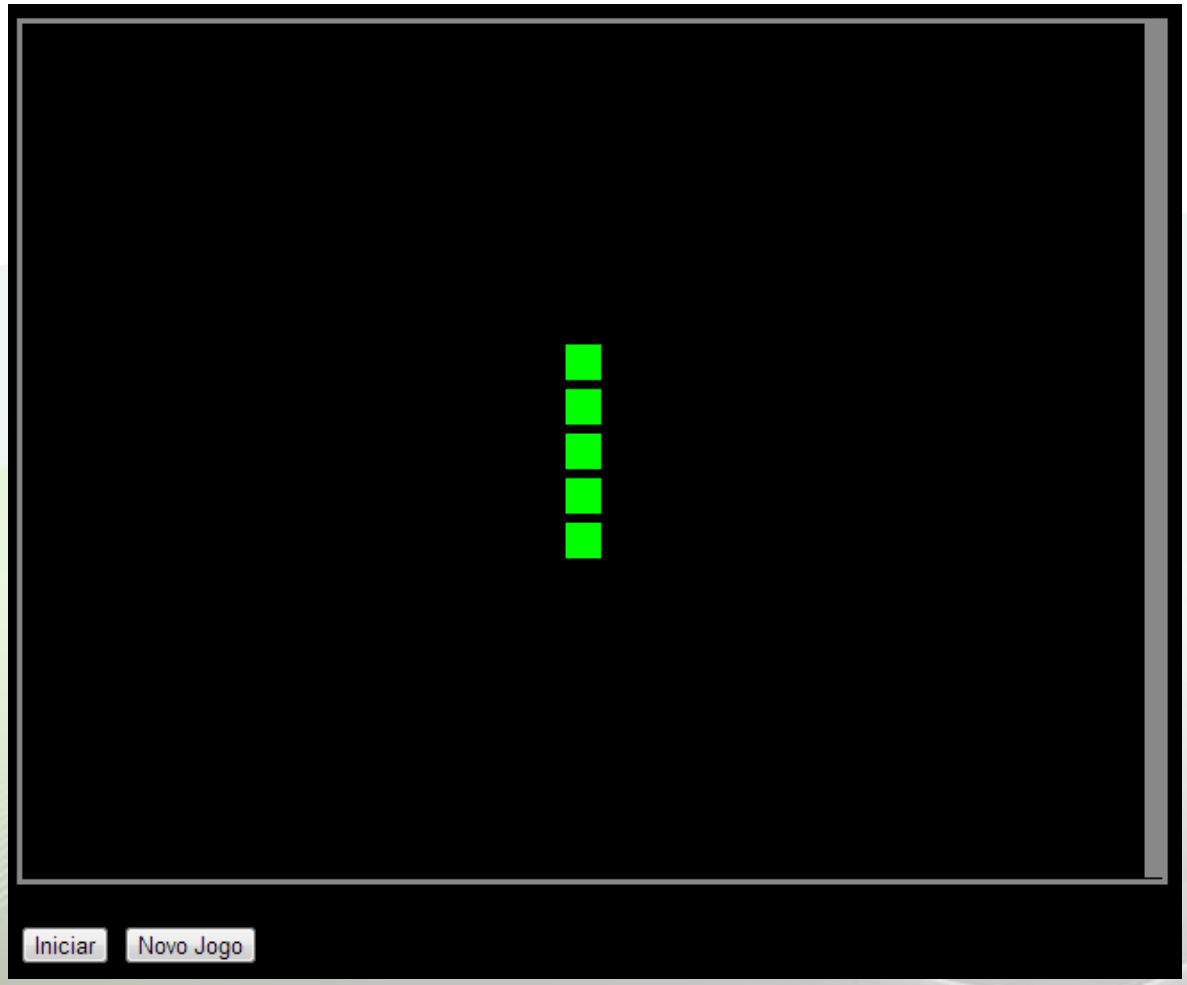
```
//Array contendo todos os nodos da Snake
var nodos = new Array();
nodos.length = 0;

function novoJogo() {
    var xcenter = Math.floor(nx / 2);
    var ycenter = Math.floor(ny / 2);
    nodos.length = 0;
    nodos.push(new Nodo(xcenter, ycenter + 2, dbaixo));
    nodos.push(new Nodo(xcenter, ycenter + 1, dbaixo));
    nodos.push(new Nodo(xcenter, ycenter, dbaixo));
    nodos.push(new Nodo(xcenter, ycenter - 1, dbaixo));
    nodos.push(new Nodo(xcenter, ycenter - 2, dbaixo));
    btPausa.innerHTML = "Iniciar";
    btPausa.disabled = false;
    desenhar();
}
```

Cadê a cobra?

script.js

```
function desenhar() {
    //Variáveis auxiliares para desenhar
    var xi, yi;
    //Limpar a tela
    context.clearRect(0, 0, canvas.width, canvas.height);
    //Desenhar bordas
    context.fillStyle = "#888888";
    context.fillRect(borda_x, 0, canvas.width - 1, canvas.height - 1);
    context.fillRect(0, borda_y, canvas.width - 1, canvas.height - 1);
    //Desenhar a Snake
    context.fillStyle = "#00FF00";
    for (i = 0; i < nodos.length; i++) {
        xi = distancia + nodos[i].x * (largura + distancia);
        yi = distancia + nodos[i].y * (largura + distancia);
        context.fillRect(xi, yi, largura, largura);
    }
}
```

Olá! Meu nome é Snake.



O que mais falta?

- Movimentos?
- Sons?
- Fruta?
- O que mais?

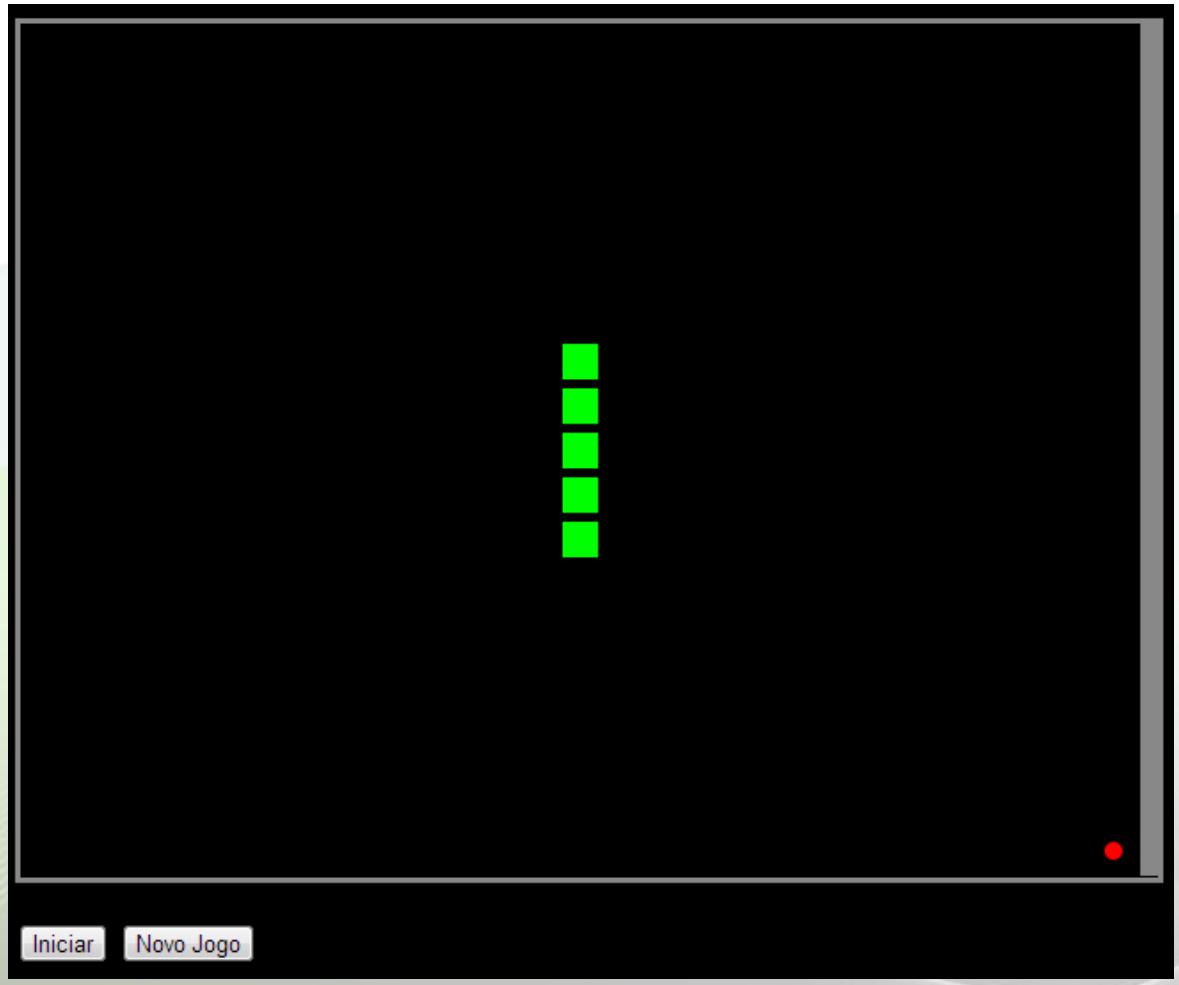
script.js

```
//Informações sobre estado atual do jogo
var rodando = false;
var xfruta;
var yfruta;
```
[Inicio do documento]

```
xfruta = nx - 1;
yfruta = ny - 1;
```
[Inicio de novoJogo()]

```
//Desenhar a fruta
context.fillStyle = "#FF0000";
xi = distancia + (xfruta * (largura + distancia)) + Math.floor(largura / 2);
yi = distancia + (yfruta * (largura + distancia)) + Math.floor(largura / 2);
context.beginPath();
context.arc(xi, yi, distancia, 0, Math.PI * 2, true);
context.closePath();
context.fill();
```
[Fim de desenhar()]

Uma fruta vermelha! *Nham*!



Snake

COLOCANDO VIDA

O que precisamos?

- Fazer a snake se movimentar:
  - Com qual intervalo de tempo?
  - Para qual direção?
- E quando a snake bater em uma Parede?
- E quando a snake bater em seu próprio corpo?
- E quando a snake encontrar com a fruta?
- E quando o usuário pressionar alguma Seta?

Snake

MOVIMENTAÇÃO DA SNAKE

Movendo a Snake

- Todo tipo de movimento tem uma velocidade;
- Como determinamos a velocidade de algum objeto?
  - Medida Espacial / Tempo!
    - KM/h
    - m/s
    - ...

Controlando o Tempo

- Como já definimos, a unidade de espaço de cada movimento da snake será um quadro;
- Agora precisamos determinar o *intervalo de tempo* que nosso jogo ira usar para fazer cada movimento da snake;
- Como nosso jogo gira em torno principalmente da snake, este tempo será como um guia para todo o jogo.

Controlando o Tempo

- Função JavaScript:
  - relogio = setInterval(“NomeFuncao()”, intervalo);
    - **relogio** é uma referência ao timer/clock que foi criado;
    - **NomeFuncao()** é a função que será executada a cada intervalo;
    - **intervalo** é um número inteiro representando a quantidade em milissegundos de intervalo entre uma execução e outra da função NomeFuncao().
  - clearInterval(relogio);
    - Para o relógio de repetição;

script.js

```
//Informações sobre estado atual do jogo
var rodando = false;
var xfruta;
var yfruta;
var relogio;
var intervalo;

function pausa() {
    rodando = !rodando;
    if (rodando) {
        btPausa.innerHTML = "Pausar";
        relogio = setInterval("loopPrincipal()" , intervalo);
    }
    else {
        clearInterval(relogio);
        btPausa.innerHTML = "Continuar";
    }
}
```

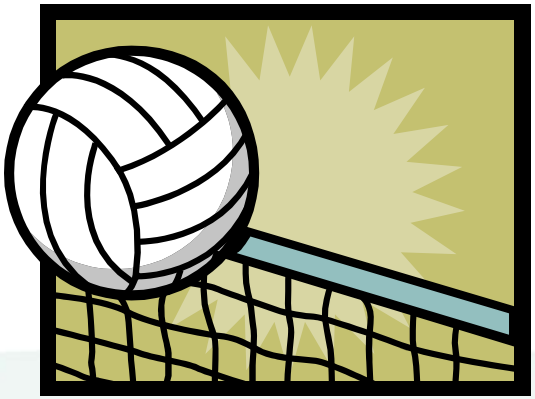
script.js

```
function novoJogo() {
   if (rodando)
      pausa();
   intervalo = 200;
   xfruta = nx - 1;
   yfruta = ny - 1;
   var xcenter = Math.floor(nx / 2);
   var ycenter = Math.floor(ny / 2);
   nodos.length = 0;
   nodos.push(new Nodo(xcenter, ycenter + 2, dbaixo));
   nodos.push(new Nodo(xcenter, ycenter + 1, dbaixo));
   nodos.push(new Nodo(xcenter, ycenter, dbaixo));
   nodos.push(new Nodo(xcenter, ycenter - 1, dbaixo));
   nodos.push(new Nodo(xcenter, ycenter - 2, dbaixo));
   btPausa.innerHTML = "Iniciar";
   btPausa.disabled = false;
   desenhar();
}
```

script.js

```
function loopPrincipal() {
   //atualizar valores
   moverSnake();
   desenhar();
}

function moverSnake() {
   //Mover todos os nodos, exceto cabeça
   for (i = nodos.length - 1; i > 0; i--) {
      nodos[i].x = nodos[i-1].x;
      nodos[i].y = nodos[i-1].y;
      nodos[i].direc = nodos[i-1].direc;
   }
   //Executa movimento da cabeça
   nodos[0].Mover();
}
```

Testar



Snake

DETECTANDO COLISÕES

Colisões

- Durante a trajetória, a snake poderá encontrar:
  - 4 paredes diferentes;
  - Qualquer parte do seu próprio corpo;
  - A fruta;
- Cada um destes acontecimentos deverá causar uma mudança de situação no jogo;

Colisão na Paredes

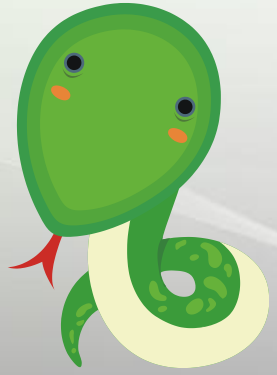
- Para verificar colisões com as paredes, podemos verificar apenas se a posição da cabeça da snake assumiu alguma posição for do tamanho da grade construída.

script.js

```
function loopPrincipal() {
   //atualizar valores
   moverSnake();
   detectarColisoes();
   desenhar();
}

function detectarColisoes() {
 //Colisão da cabeça com alguma parede
   if ((nodos[0].x < 0) || (nodos[0].x >= nx) || (nodos[0].y < 0) || (nodos[0].y >= ny)) {
       executarGameOver();  //Game Over!
   }




}
```

Colisão no Corpo

- Para verificar colisões com qualquer parte do próprio corpo, devemos verificar se a posição da cabeça é a mesma de qualquer um, dentre todos, os nodos do corpo.



script.js

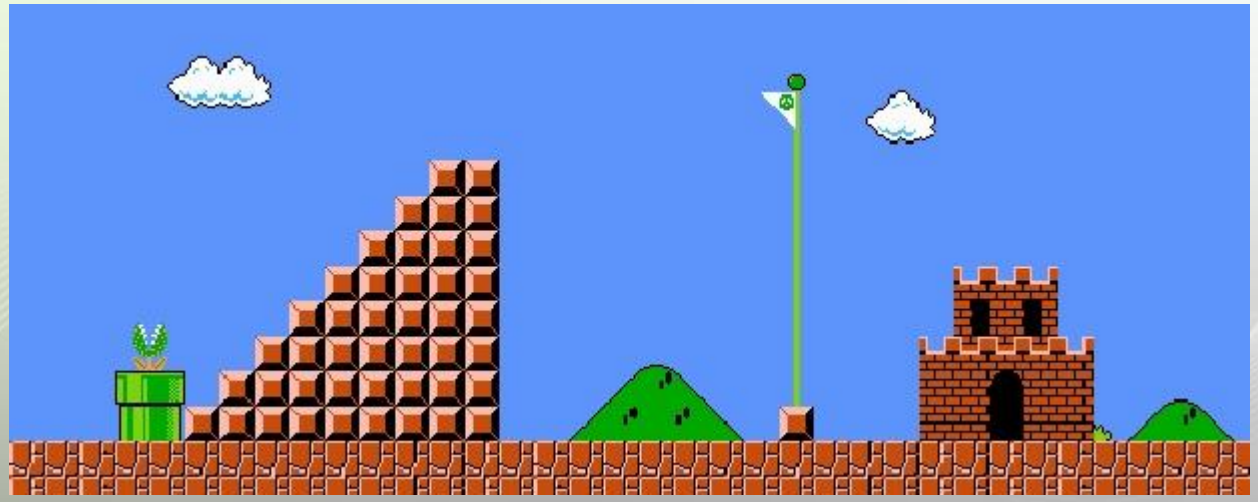
```
function loopPrincipal() {
   //atualizar valores
   detectarColisoes();
   moverSnake();
   desenhar();
}

function detectarColisoes() {
 //Colisão da cabeça com alguma parede
   if ((nodos[0].x < 0) || (nodos[0].x >= nx) || (nodos[0].y < 0) || (nodos[0].y >= ny)) {
       executarGameOver();  //Game Over!
   }
 //Colisão da cabeça com o corpo
   for (i = 1; i < nodos.length; i++) {
      if ((nodos[0].x == nodos[i].x) && (nodos[0].y == nodos[i].y)) {
         executarGameOver();  //Game Over!
      }
   }
}
```

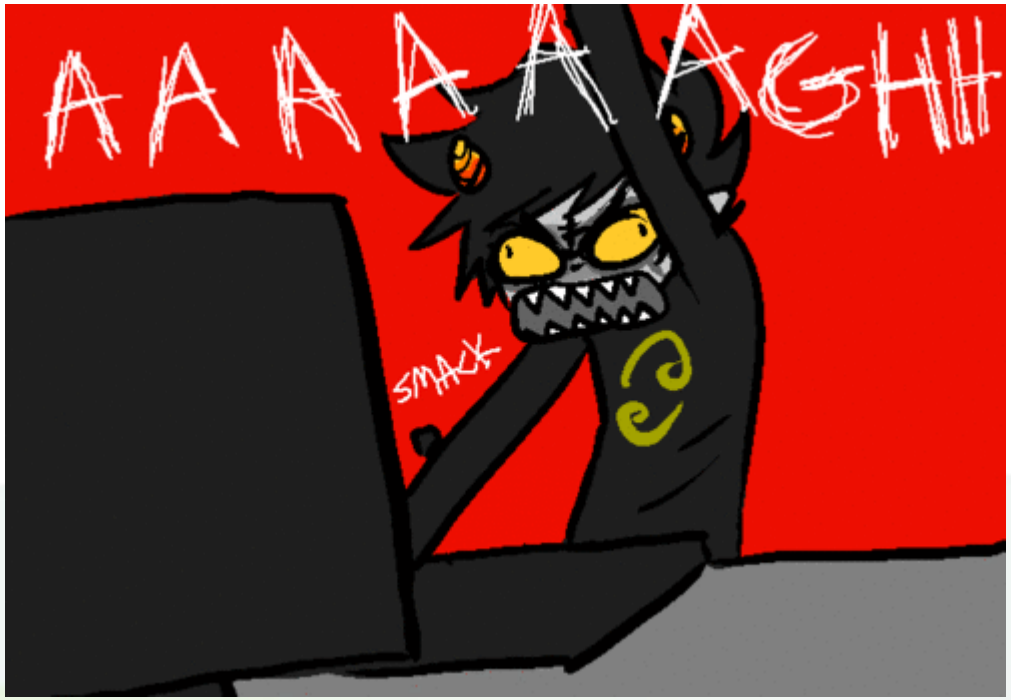
script.js

```
function executarGameOver() {
    btPausa.disabled = true;
    if (rodando)
        pausa();
}
```

Alterações na Iteração

- O que falta alterar?
  - Detecção com fruta?
  - Interação com usuário?
  - Sons?





Snake

INTERAGINDO COM O USUÁRIO

Eventos!

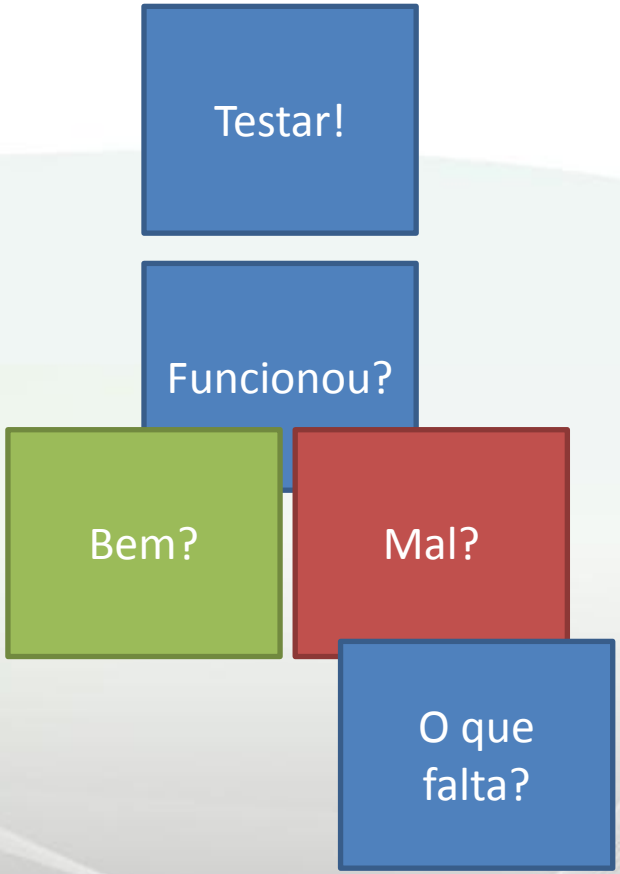
- A interação é dada por uma troca entre a máquina e o usuário;
- A máquina fornece principalmente imagens que descrevem uma situação, onde pode ser necessária a intervenção do usuário;
- O usuário ira intervir basicamente através de comandos!
  - Comandos são captados através de eventos.

Eventos!

- Nosso document possuí propriedades de eventos que podem ser associadas à funções quaisquer;
- Estas funções determinam algo a ser feito quando aquele evento ocorrer:
  - document.onkeydown
    - Ao descer uma tecla qualquer;
  - document.onkeyup
    - Ao soltar uma tecla qualquer;

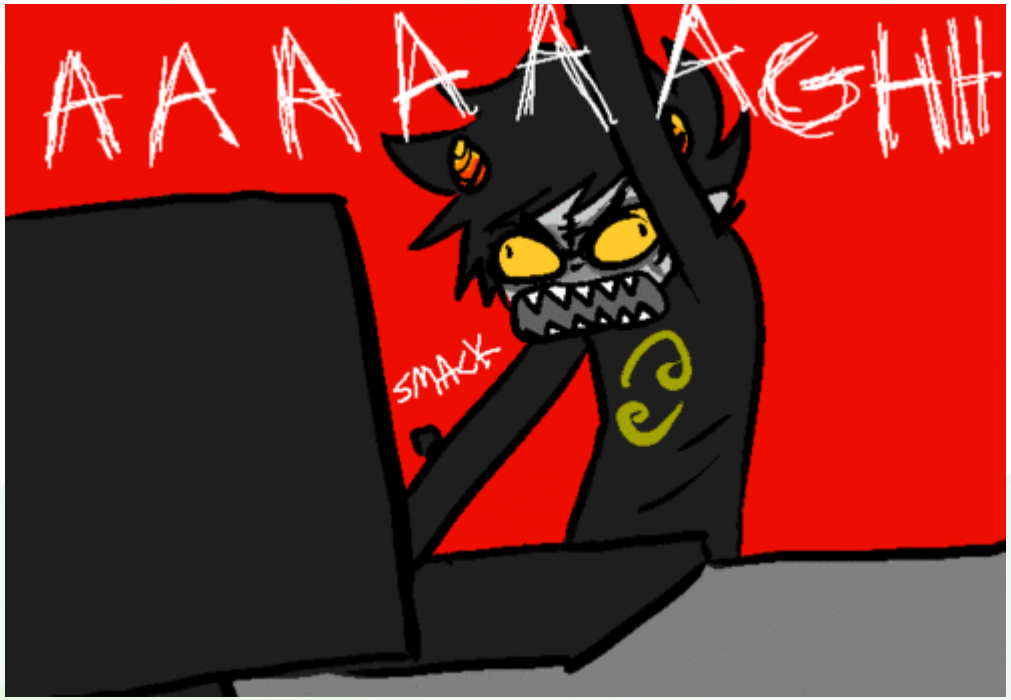
script.js

```
//Eventos
document.onkeydown=onKD;

function onKD(evt) {
      switch (evt.keyCode) {
       case 37: //esquerda
          nodos[0].direc = desquerda;
          break;
       case 38: //cima
          nodos[0].direc = dcima;
          break;
       case 39: //direita
          nodos[0].direc = ddireita;
          break;
       case 40: //baixo
          nodos[0].direc = dbaixo;
          break;
    }
}
```



Comandos

- Conforme definimos nosso “relógio” que executa tarefas a cada intervalo de tempo, os dados serão processados a cada intervalo *X* de tempo;
- Quando dois, ou mais, comandos do usuário forem dados em um espaço de tempo menor do que o intervalo *X*, apenas o último comando antes do fim do intervalo é que será executado.



Snake

A FILA DE COMANDOS

A Fila de Comandos

- Para resolver este problema, podemos guardar cada comando digitado pelo usuário em uma fila de direções;
- A cada movimento, a *Snake* deverá retirar a próxima direção e a executar;
  - É importante que ao mover a *Snake* se descartem todas as direções do inicio da fila que sejam iguais a atual, pois o usuário pode ter pressionado diversas vezes a mesma tecla.

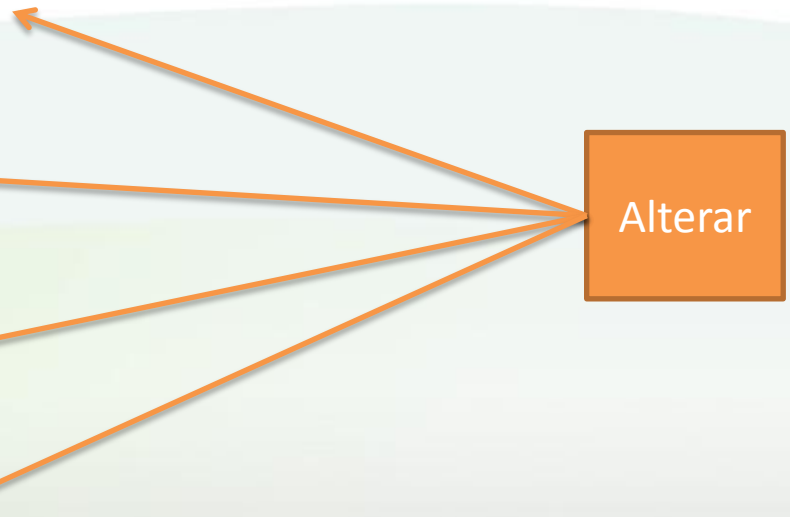
script.js

```
//Informações sobre estado atual do jogo
var rodando = false;
var xfruta;
var yfruta;
var relogio;
var intervalo;
var proxDirec = new Array();
proxDirec.length = 0;
```

Inserir

script.js

```
function onKD(evt) {
      switch(evt.keyCode) {
       case 37: //esquerda
          proxDirec.push(desquerda);
          break;
       case 38: //cima
          proxDirec.push(dcima);
          break;
       case 39: //direita
          proxDirec.push(ddireita);
          break;
       case 40: //baixo
          proxDirec.push(dbaixo);
          break;
   }
}
```



script.js

```
function moverSnake() {
    //Mover todos os nodos, exceto cabeça
    for (i = nodos.length - 1; i > 0; i--) {
        nodos[i].x = nodos[i-1].x;
        nodos[i].y = nodos[i-1].y;
        nodos[i].direc = nodos[i-1].direc;
    }
    //Se lista de comandos não estiver vazia
    if (proxDirec.length > 0)
        //Se há uma direção diferente da atual
        if (nodos[0].direc != proxDirec[0])
            //Alterar a direção
            nodos[0].direc = proxDirec[0];
    //Executa movimento da cabeça
    nodos[0].Mover();
    //Enquanto houverem comandos na lista
    while (proxDirec.length > 0) {
        //Se o comando é redundante
        if (proxDirec[0] == nodos[0].direc)
            //Remove o comando do inicio da lista
            proxDirec.shift();
        else
            //Se não for, para a repetição
            break;
    }
}
```

Snake

ESTÍMULOS SONOROS

Estímulos Sonoros

- Conforme comentado anteriormente, quanto mais estimularmos, de forma positiva, os sentidos dos jogadores, maior a probabilidade dele se sentir como parte do jogo;
- Para isto, iremos adicionar alguns pequenos sons associados a eventos como colisões;
- Baixe os arquivos e salve na subpasta **snd**:
  - br.mp_ e br.ogg;
  - cp.mp_ e cp.ogg; e
  - ha.mp_ e ha.ogg.
- ~~Renomeie as extensões .mp_ para .mp3~~

<audio> e <source>

- HTML 5!
- MIME Types:
  - MP3 – audio/mpeg
  - Ogg – audio/ogg
  - Wav – audio/wav
- Suporte:
  - Ps.: Múltiplos <source> fornecem redundância!

Browser	MP3	Wav	Ogg
IE 9+	Sim	Não	Não
Chrome 6+	Sim	Sim	Sim
Firefox 3.6+	Não	Sim	Sim
Safari 5+	Sim	Sim	Não
Opera 10+	Não	Sim	Sim

index.html

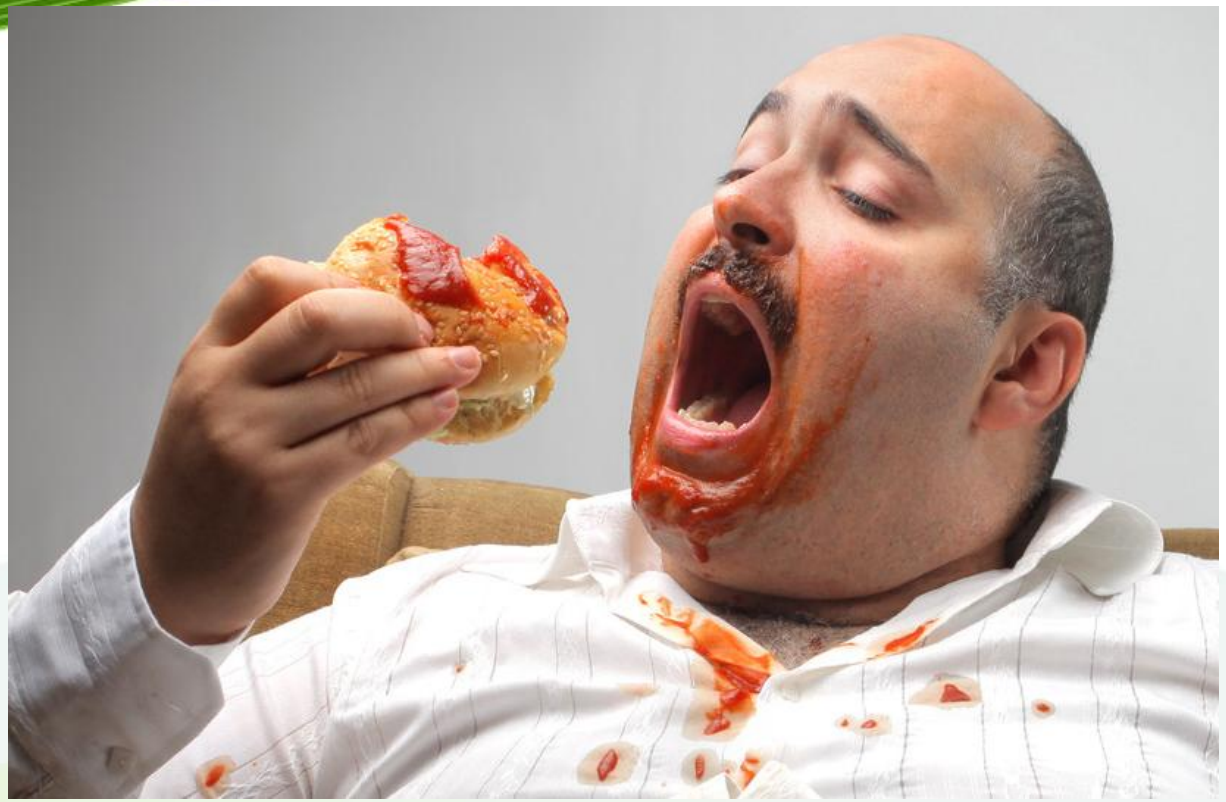
- Adicionar dentro do <body>:

```
<audio controls id= "comer1">
    <source src= "snd/cp.mp_" type="audio/mpeg">
    <source src= "snd/cp.ogg" type="audio/ogg">
</audio>
<audio controls id= "comer2">
    <source src= "snd/br.mp_" type="audio/mpeg">
    <source src= "snd/br.ogg" type="audio/ogg">
</audio>
<audio controls id="gameover">
    <source src= "snd/ha.mp_" type="audio/mpeg">
    <source src= "snd/ha.ogg" type="audio/ogg">
</audio>
```

script.js

```
//Referências dos objetos gráficos
var canvas = document.getElementById("tela");
var context = canvas.getContext("2d");
var btPausa = document.getElementById("btPausa");
var sndcomer1 = document.getElementById("comer1");
var sndcomer2 = document.getElementById(“comer2");
var sndgameover = document.getElementById("gameover");
```

- **Prática**: Insira os comandos de reprodução nos locais apropriados:
  - ~~sndcomer1.play();~~
  - ~~sndcomer2.play();~~
  - sndgameover.play();



Snake

COMENDO!

Comendo

- Cada vez que a cabeça da snake encontrar uma fruta, a fruta será comida!
  - O ato de comer a fruta deve fazer com que aquela fruta desapareça e apareça uma nova fruta em outro lugar;
    - O lugar de surgimento da nova fruta deve ser aleatório
    - A fruta não pode aparecer em nenhuma posição já ocupada por alguma parte do corpo da snake;
  - Ao comer cada fruta, o corpo da snake deve crescer em um nodo.

Comendo

- Ao comer a fruta, deverá ser emito apenas um dos dois sons de comer:
  - comer1
  - comer2
- O algoritmo deverá sortear de forma aleatória, dando preferência para o som “comer1”.

script.js

```
function sndComer() { //Reproduzir som aleatório de comer
   if (Math.random() < 0.8)
      sndcomer1.play();
   else
      sndcomer2.play();
}

function novaPosFruta() { //Determinar uma nova posição para a fruta
   do {
      xfruta = Math.floor(Math.random() * nx);
      yfruta = Math.floor(Math.random() * ny);
   } while (colisaoFruta() == true);
}

function colisaoFruta() { //Verificar se posição da fruta colide com corpo da snake
   for (i = 0; i < nodos.length; i++) {
      if ((xfruta == nodos[i].x) && (yfruta == nodos[i].y))
         return true;
   }
   return false;
}
```

script.js

```
function detectarColisoes() {
   ...
   //Comer a fruta
   if ((nodos[0].x == xfruta) && (nodos[0].y == yfruta)) {
      sndComer();
      var ultimo = nodos.length - 1;
      nodos.push(new Nodo(nodos[ultimo].x, nodos[ultimo].y, nodos[ultimo].direc));
      var novoultimo = ultimo + 1;
      switch (nodos[ultimo].direc) {
         case dbaixo:
            nodos[novoultimo].y -= 1;
            break;
         case ddireita:
            nodos[novoultimo].x -= 1;
            break;
         case dcima:
            nodos[novoultimo].y += 1;
            break;
         case desquerda:
            nodos[novoultimo].x += 1;
            break;
      }
      novaPosFruta();
   }
}
```

Snake

ANIMANDO AS COISAS

Animações

- O movimento da *Snake* já é uma animação bem simples. Animações nada mais são do que mudanças constante na cena, representando alguma coisa;
- Animações podem ser definidas de formas simples, como uma sequência de imagens que são alternadas, ou de formas mais complexas, através da reconstrução de uma geometria ao longo do tempo.

Animações

- Em nosso jogo, iremos inserir uma animação muito simples, apenas a fruta vermelha irá girar em torno de seu centro;
- Para isto, iremos redesenhar ela, não completamente circular, pois assim podemos observar alterações na imagem;

script.js

```
//Informações sobre estado atual do jogo
var rotacao = 0;
```

Inserir no Inicio

```
//Desenhar a fruta
context.fillStyle = "#FF0000";
xi = distancia + (xfruta * (largura + distancia)) + Math.floor(largura / 2);
yi = distancia + (yfruta * (largura + distancia)) + Math.floor(largura / 2);
rotacao += Math.PI * 0.1;
if (rotacao > Math.PI * 2)
    rotacao -= Math.PI * 2;
var r = rotacao + (Math.PI * 1.5);
context.beginPath();
context.arc(xi, yi, distancia, r, rotacao, true);
context.closePath();
context.fill();
```

Alterar em desenhar()

Trabalho

1. Customize cores e outras configurações do arquivo de estilo;
2. Customize cores, tamanhos e disposição dos objetos do jogo (dentro do *Javascript*). Utilize gradientes ou imagens;
3. Complete o HTML informando o nome da disciplina, o nome do instituto e o seu nome, dispondo os elementos com layouts CSS;
4. Existe um **bug**: quando a Snake está indo em uma determinada direção, caso seja pressionada uma tecla para ela ir ao sentido oposto, ela morre. Corrija este bug.
5. Outro **bug**: quando o jogo não esta em execução, teclas pressionadas são enfileiradas. Ao retomar o jogo a snake irá realizar movimentos inesperados! Corrija este bug.
6. Adicione novas características e funcionalidades:
    1. Crie um placar com pontuação;
    2. Crie uma indicação visual dentro do Canvas de fim de jogo;
    3. Crie frutas especiais, que aparecem em momentos aleatórios, valem mais pontos, mas permanecem por pouco tempo disponíveis;
    4. Adicione teclas de atalho para “Pausa” e para “Novo Jogo”;
    5. Ao comer cada fruta, aumente um pouco a velocidade;
    6. Crie outros elementos a seu critério;
7. Entregue os arquivos por e-mail ao Professor.