INSTITUTO FEDERAL
SANTA CATARINA

Construindo um Jogo para a Web - *Tetris*

Programação para a Internet

Prof. Vilson Heck Junior

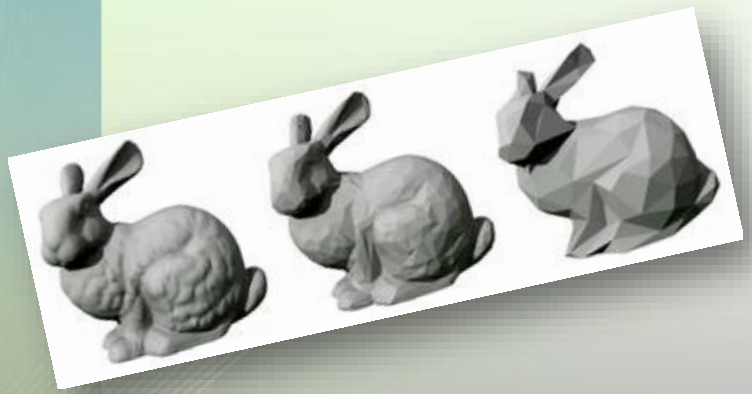
GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA

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

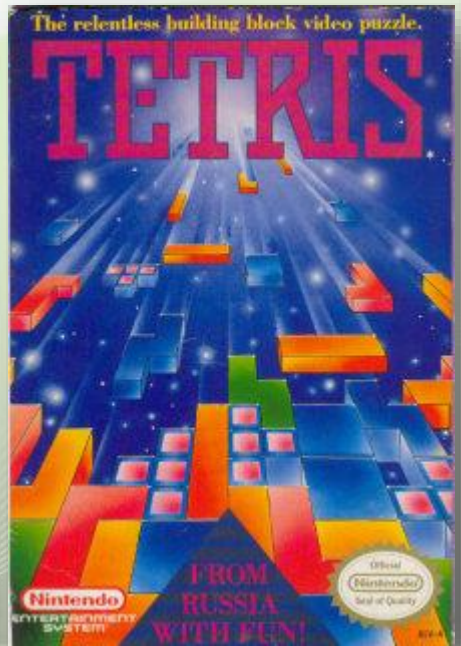
- O som é o elemento responsável por estimular o sentido da audição;
- Não tanto quanto os gráficos, mas os sons são responsáveis por completar uma boa sensação de imersão em jogos e entretenimento;
- Geralmente os sons (músicas ou barulhos) serão escolhidos conforme um determinado contexto ou acontecimento.

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

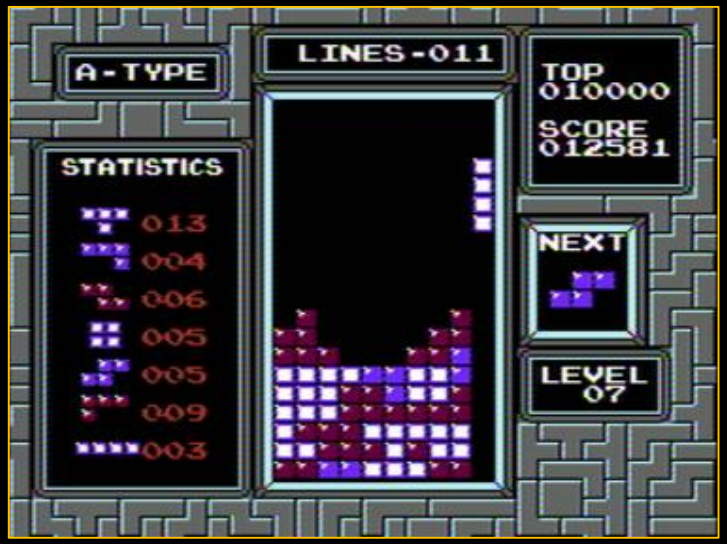
- Tetris:
  - Jogo de encaixe de formas pelo Russo Alexey Pajitnov;
  - Primeira versão foi para o computador soviético DVK-2;



DVK-2 (1984)

IBM PC (1986)

NES (1989)

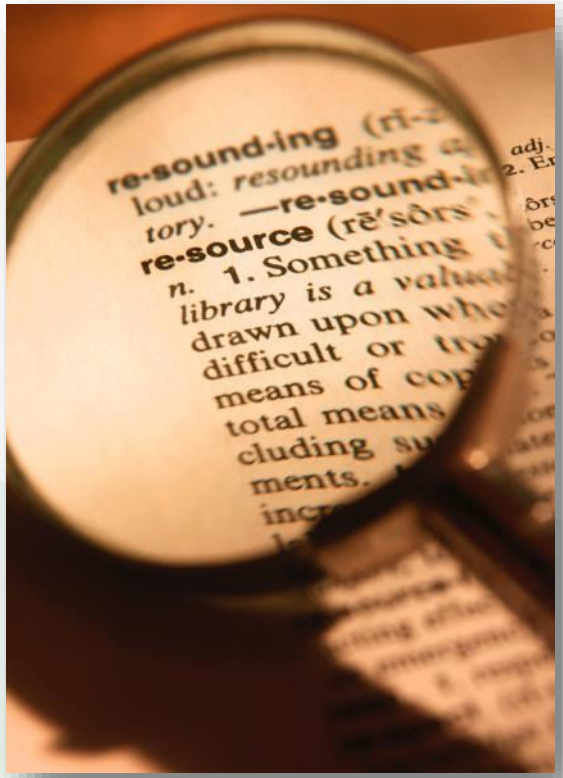


Smartphones (2013)

INSTITUTO FEDERAL
SANTA CATARINA

Nosso Conceito

GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA



Tetris

LISTA DE RECURSOS INICIAIS

Recursos Iniciais

- Pasta: “Tetris”:
  - 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;
  - tetris.js
    - Faremos todo o processamento do jogo, ou seja, daremos vida aos elementos existentes no documento web.
  - Bloco.js
    - Implementaremos uma Classe Bloco para definir o comportamento das peças utilizadas no jogo Tetris;

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 o arquivo de **<script>** Bloco.js dentro do <head>;
- Adicione o arquivo de **<script>** tetris.js ao fim do <body>**;**
  - ***Importante: adicionar Bloco.js antes de tetris.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=340 height=500
  -
  - <button>Iniciar</button>
    - type=“button” onclick=“pausar()” id=“btPausar”
  - <button>Novo Jogo</button>
    - type=“button” onclick=“novoJogo()” id=“btNovo”
  -

estilo.css

```
body {  background-color: black; }
p {     text-align: center; }
#tela {
        background-color: #000000;
        background-image: url('img/bg.png');
        border: solid #FF0000 2px;
}
button {
        background-color: #9dce2c;
        border: none;
        color: white;
        padding: 10px;
        cursor: pointer;
}
button:hover { background-color: #7bac0a; }

button:disabled {
        background-color: #DDDDDD;
        cursor: not-allowed;
}
```

Tetris

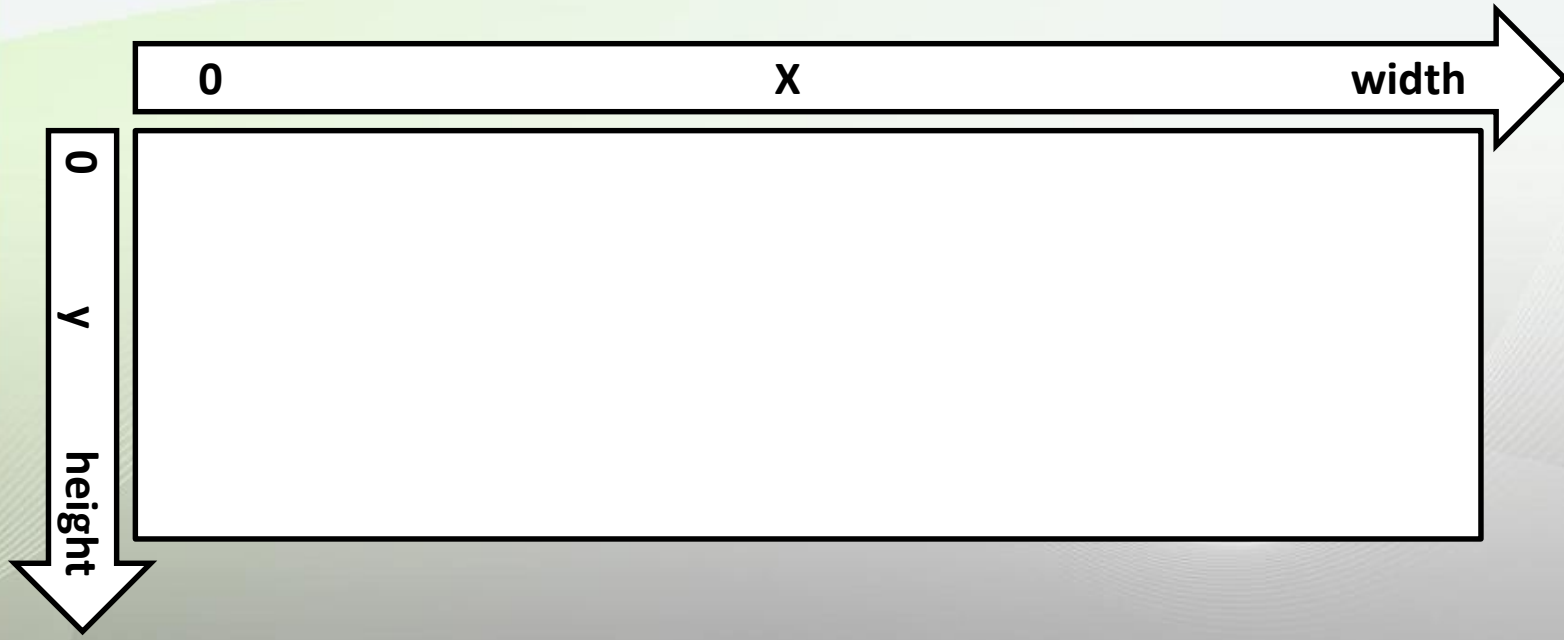
DESENHANDO NO CANVAS

<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;

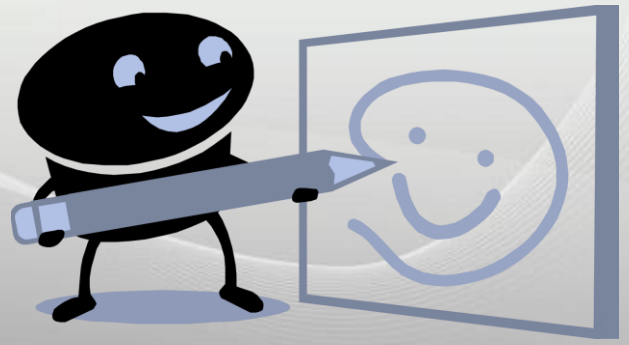


tetris.js

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

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

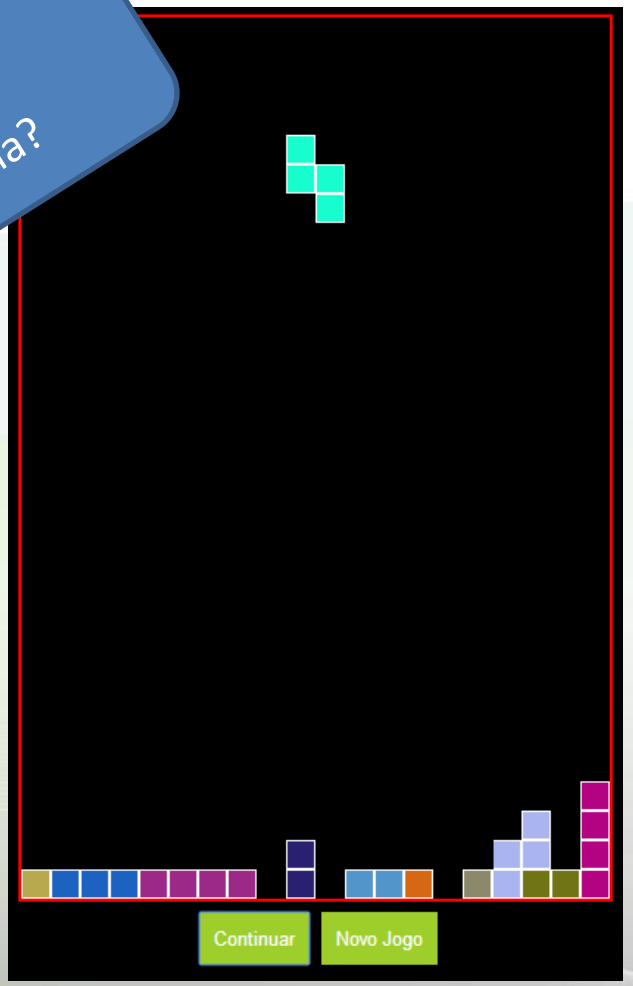
Desenhando

- 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?



Formas

- Encontramos: Diversas peças formadas por pequenos quadrados;
- Iremos criar matrizes de valores booleanos para armazenar o formato das peças utilizadas. Ex. da peça "L":

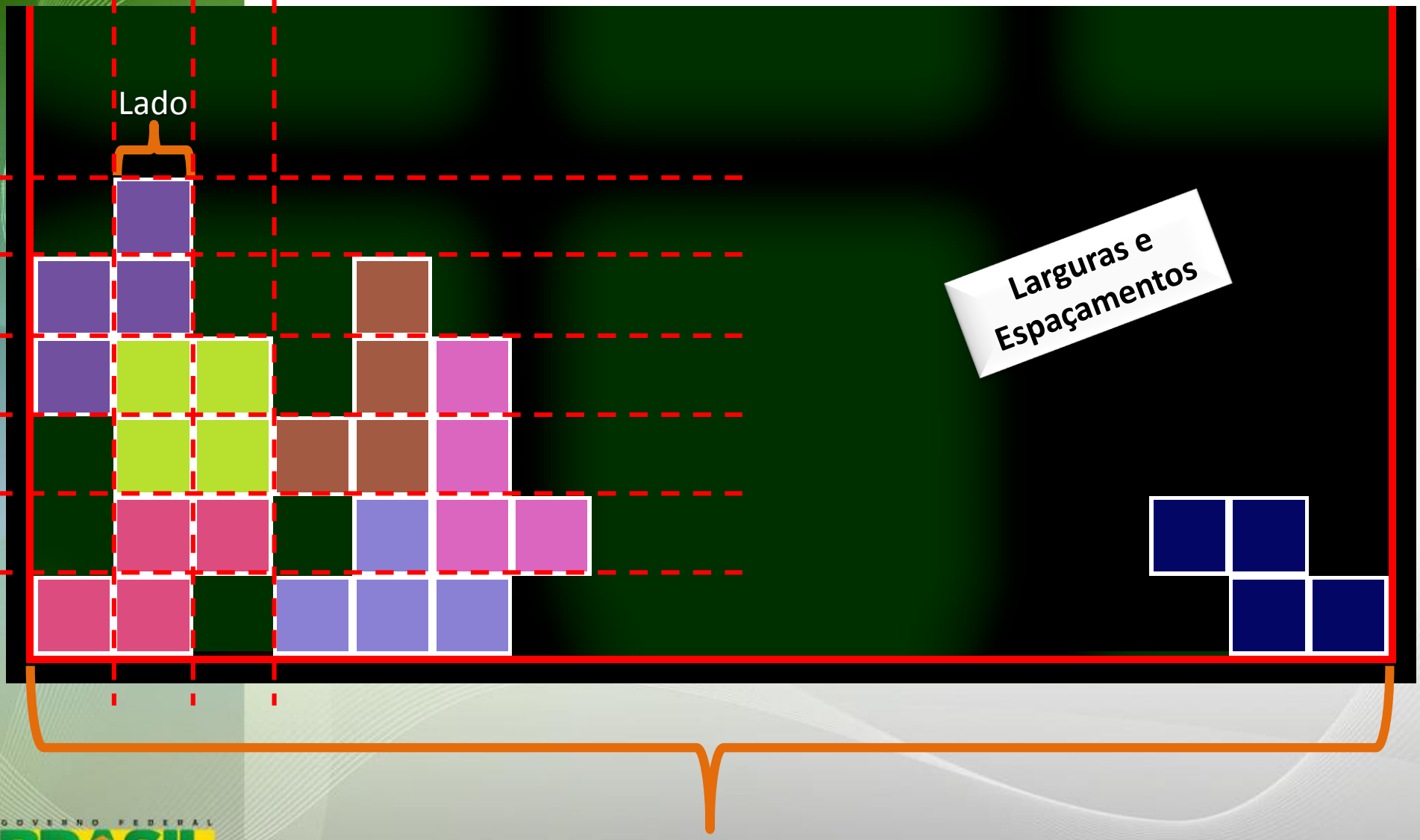
F	V	F
F	V	F
F	V	V

pedras.js

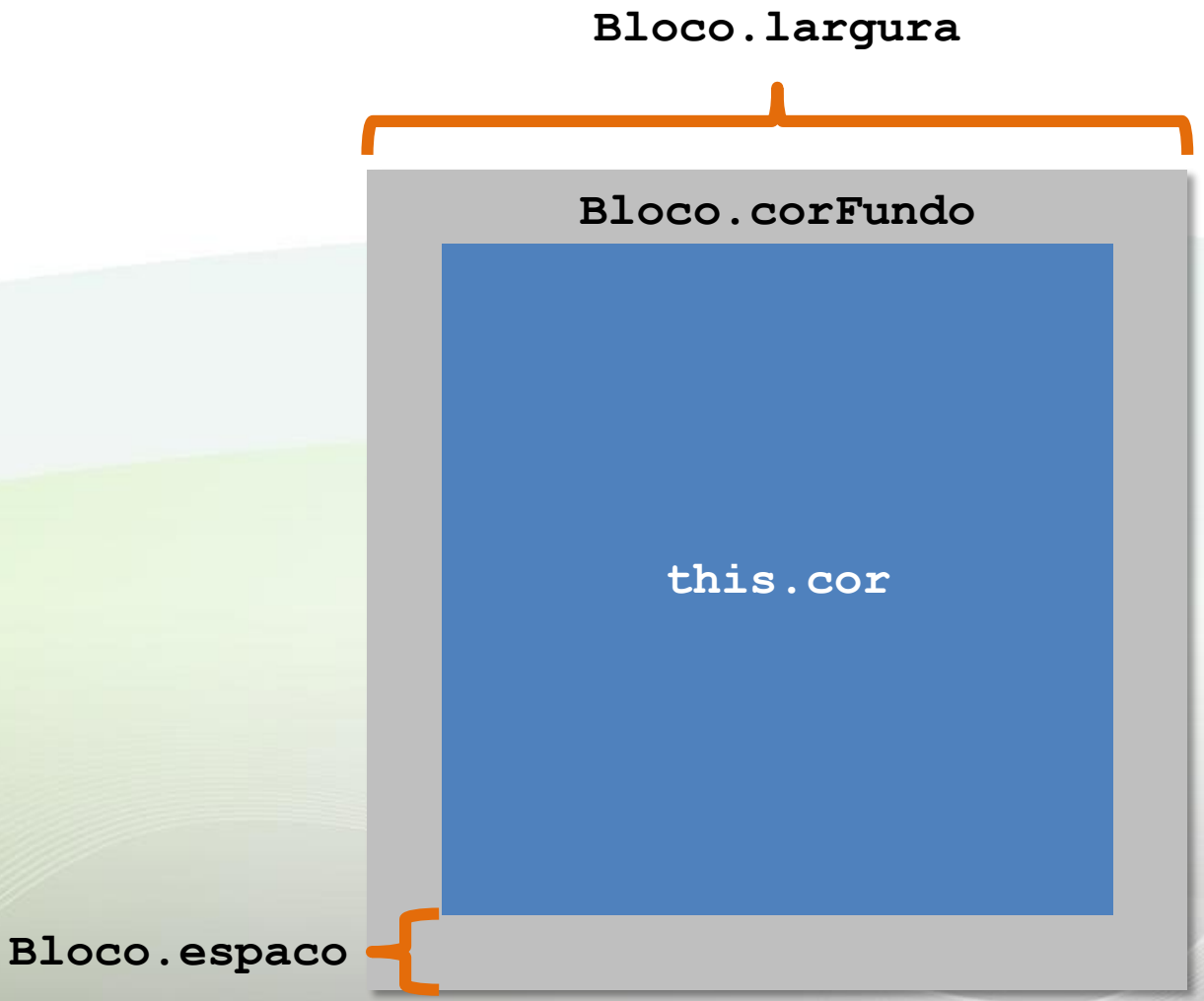
```
//Atalhos para facilitar
   escrita das matrizes
var X = true;
var l = false;

var Lesq = [
//Posição 1
[[l,X,l],
 [l,X,l],
 [l,X,X]],
//Posição 2
[[X,X,X],
 [X,l,l]],
//Posição 3
[[l,X,X],
 [l,l,X],
 [l,l,X]],
//Posição 4
[[l,l,X],
 [X,X,X]]
];

// ... Baixar arquivo...

//Cria um array para indexar
   de forma aleatória uma
   peça
var todasPedras = [quadrado,
   linha, Lesq, Ldir, Te, S1,
   S2];
```

Baixar arquivo e adicionar script dentro da seção head

Grade



Bloco – conjunto de pedras



Bloco.js

```
var Bloco = function(xi, yi, cori) {

}; //Fim parte dinâmica de Bloco

//Atributos estáticos
Bloco.corFundo = "#FFFFFF";
Bloco.espaco = 1;
Bloco.largura = 20;
//Método estático
Bloco.desenhar = function(context, x, y, corFrente) {
        context.fillStyle = Bloco.corFundo;
        var x1 = x * Bloco.largura;
        var y1 = y * Bloco.largura;
        context.fillRect(x1, y1, Bloco.largura, Bloco.largura);
        context.fillStyle = corFrente;
        x1 = x1 + Bloco.espaco;
        y1 = y1 + Bloco.espaco;
        context.fillRect(x1, y1, Bloco.largura - Bloco.espaco*2,
                                 Bloco.largura - Bloco.espaco*2);
};
```

tetris.js

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

//Variáveis e Configurações do jogo
var nx = Math.floor(canvas.width / Bloco.largura);
var ny = Math.floor(canvas.height / Bloco.largura);
var tabuleiro = null;
var peca;
var pausa = true;
```

tetris.js

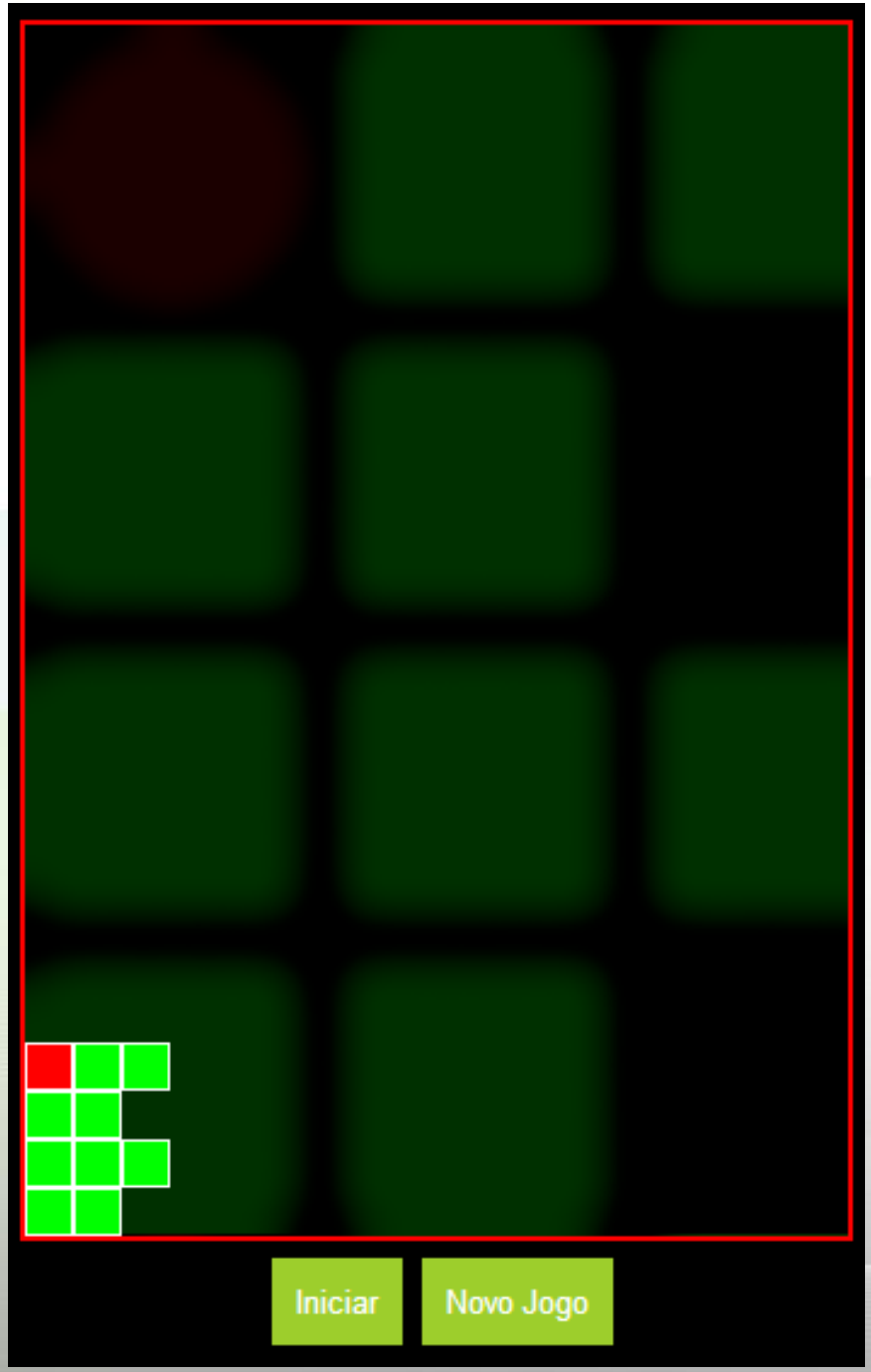
```
function novoJogo() {
        if (!pausa) {
                //pausar();
        }
        //Criar o tabuleiro vazio
        tabuleiro = new Array(nx);
        for (x = 0; x < nx; x++) {
                tabuleiro[x] = new Array(ny);
                for (y = 0; y < ny; y++) {
                        tabuleiro[x][y] = null;
                }
        }
        //novaPeca();
        //criarIF();
        //desenharTudo();
        btNovo.disabled = false;
        btPausar.disabled = false;
        btPausar.innerHTML = "Iniciar";
}
```

tetris.js

```
function novoJogo() {
        ...
}
function criarIF() {
        tabuleiro[0][ny-4] = "#FF0000";
        tabuleiro[0][ny-3] = "#00FF00";
        tabuleiro[0][ny-2] = "#00FF00";
        tabuleiro[0][ny-1] = "#00FF00";
        tabuleiro[1][ny-4] = "#00FF00";
        tabuleiro[1][ny-3] = "#00FF00";
        tabuleiro[1][ny-2] = "#00FF00";
        tabuleiro[1][ny-1] = "#00FF00";
        tabuleiro[2][ny-4] = "#00FF00";
        tabuleiro[2][ny-2] = "#00FF00";
}
novoJogo(); //Fora do método
```

tetris.js

```
function criarIF() {
}
function desenharTudo() {
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        for (x = 0; x < nx; x++) {
                for (y = 0; y < ny; y++) {
                        if (tabuleiro[x][y] != null) {
                                Bloco.desenhar(ctx, x, y,
                                                tabuleiro[x][y]);
                        }
                }
        }
        //peca.desenharBloco(ctx);
}
novoJogo(); //Fora do método
```



Bloco.js

```
var Bloco = function(xi, yi, cori) {

        //Atributos dinâmicos
        this.x = xi;
        this.y = yi;
        this.estadoAtual = 0;
        this.mapa = todasPedras[Math.round(Math.random() *
                                    (todasPedras.length-1))];
        this.maxEstados = this.mapa.length;
        this.cor = cori;

}; //Fim parte dinâmica de Bloco
//Atributos estáticos
        ...
```

Bloco.js (métodos dinâmicos)

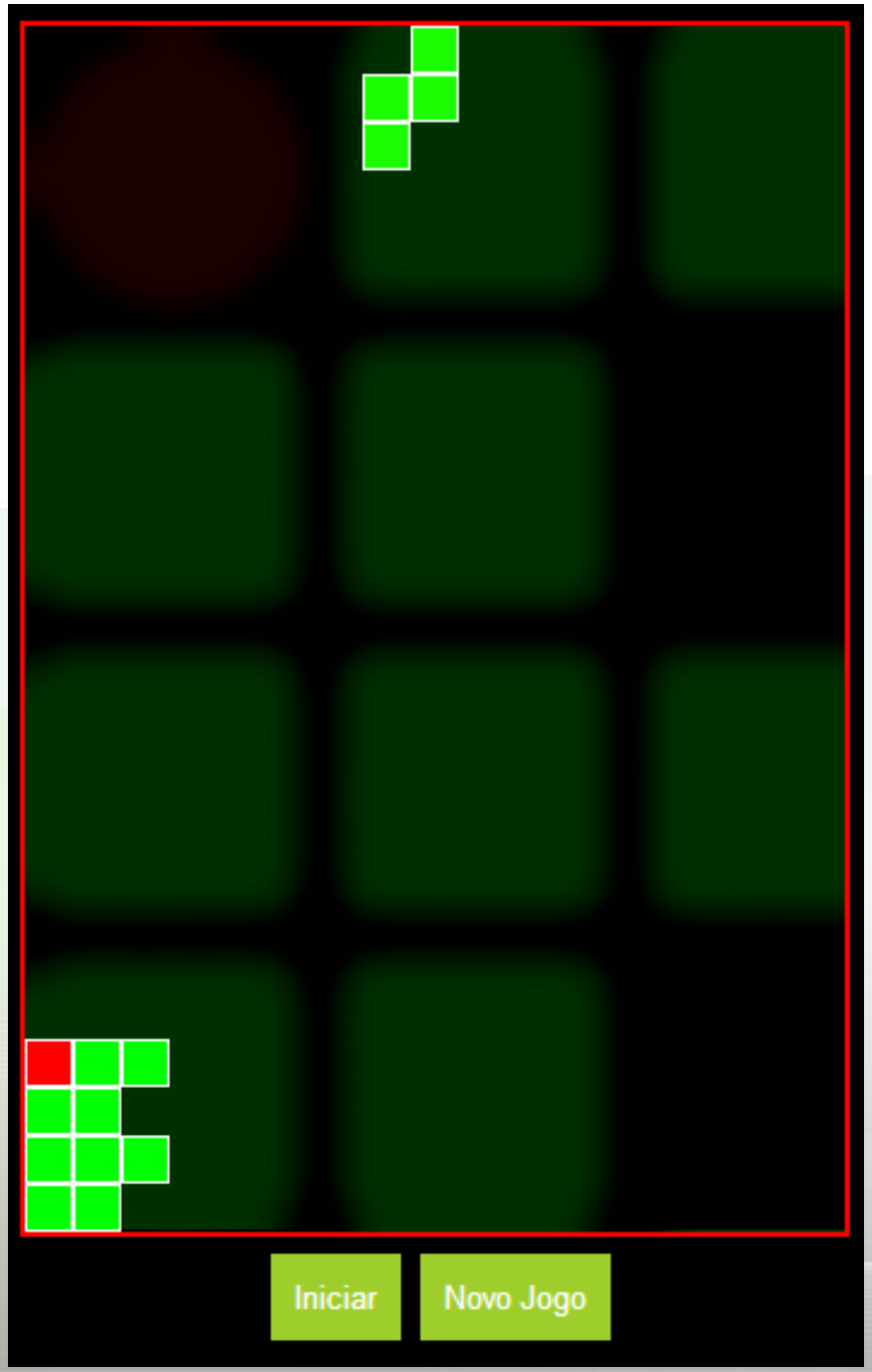
```
//Atributos dinâmicos
...
//Métodos dinâmicos
this.desenharBloco = function(context) {
        for (i = 0; i < this.mapa[this.estadoAtual].length; i++) {
                for (j = 0; j < this.mapa[this.estadoAtual][i].length;
                                                        j++) {
                        if (this.mapa[this.estadoAtual][i][j] == true) {
                                Bloco.desenhar(context, this.x + i,
                                                this.y + j, this.cor);
                        }
                }
        }
};

}; //Fim parte dinâmica de Bloco
```

tetris.js

```
function criarIF() {
        ...
}


function novaPeca() {
        peca = new Bloco(Math.floor(nx/2-1), 0, novaCor() );
}


novoJogo(); //Fora do método
```

tetris.js

```
function novaPeca() {
        ...
}
function novaCor() {
        var r,g,b,dif;
        do {
                r = Math.round(255 * Math.random());
                g = Math.round(255 * Math.random());
                b = Math.round(255 * Math.random());
                dif = Math.abs(r - g) + Math.abs(g - b) + Math.abs(r - b);
        } while (dif < 60);
        var sr = r.toString(16);
        if (sr.length < 2) { sr = "0" + sr;   }
        var sg = g.toString(16);
        if (sg.length < 2) { sg = "0" + sg;   }
        var sb = b.toString(16);
        if (sb.length < 2) { sb = "0" + sb;   }
        return ("#" + sr + sg + sb);
}
novoJogo(); //Fora do método
```


Iniciar
Novo Jogo

O que mais falta?

- Movimentos?
- Sons?
- O que mais?

Tetris

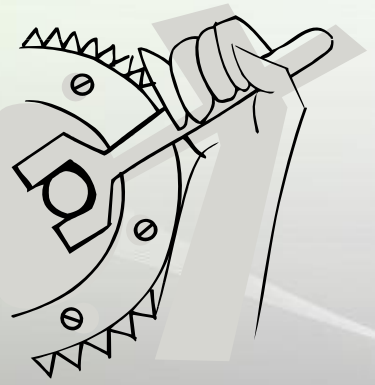
COLOCANDO VIDA

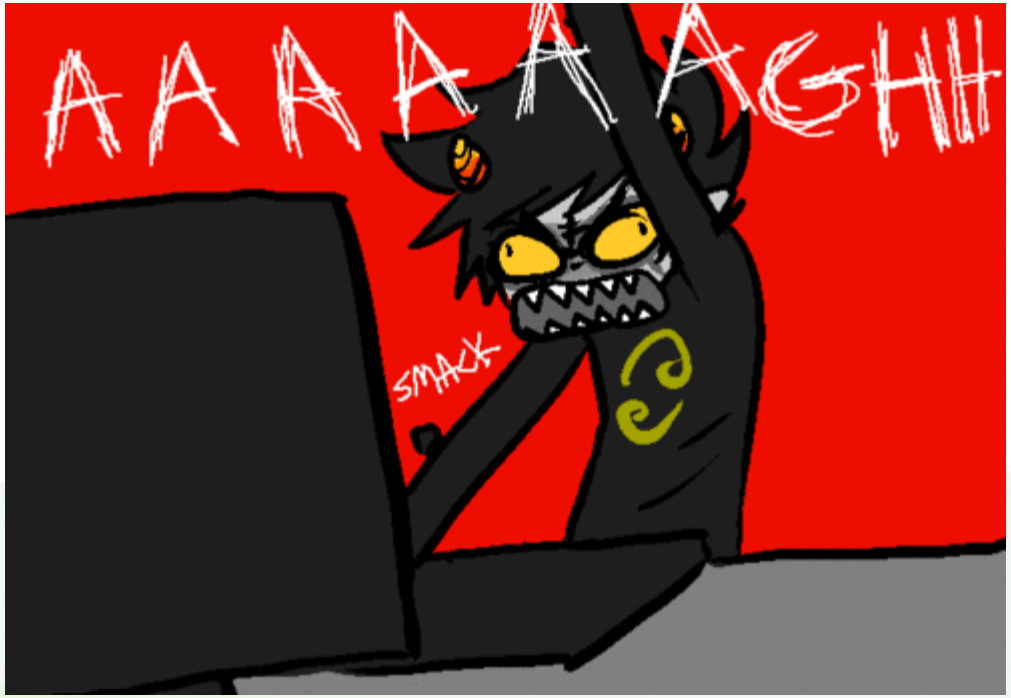
O que precisamos?

- Fazer as pedras se movimentar:
  - Com qual intervalo de tempo?
  - Para qual direção?
- E quando a pedra bater em uma Parede?
- E quando a pedra bater em outras pedras?
- E quando uma linha for preenchida?
- E quando o usuário pressionar alguma Seta?

Movimentação das Pedras

- As pedras poderão ser movidas por dois motivos:
  - Pressionar de uma tecla pelo jogador;
  - Avanço natural associado ao tempo;





Tetris

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 irá intervir basicamente através de comandos!
  - Comandos são captados através de eventos.

Eventos!

- Nosso document possui 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;

tetris.js

```
//Variáveis e Configurações do jogo

//Eventos de tecla para método onKD
document.onkeydown = onKD;

function onKD(evt) {
        if (evt.keyCode == 38) {
                //peca.girarEsquerda(tabuleiro);
        }
        //var tq = peca.encostado(tabuleiro);
        //var aa = tq & 2;
        //var bb = tq & 4;
        //if (evt.keyCode == 39 && bb == 0) peca.x++;
        //if (evt.keyCode == 37 && aa == 0) peca.x--;
        //if (evt.keyCode == 40) avancarJogo();
        desenharTudo();
}

novoJogo();
```

Bloco.js

```
//Métodos dinâmicos
        ...

this.girarEsquerda = function(tabul) {
        do {
                this.estadoAtual--;
                if (this.estadoAtual < 0) {
                        this.estadoAtual = this.maxEstados - 1;
                }
        } while (this.detectarColisao(tabul) == true);
};
```

Bloco.js

```
//Métodos dinâmicos
        ...

this.detectarColisao = function(tabul) {
   for (i = 0; i < this.mapa[this.estadoAtual].length; i++) {
      for (j = 0; j < this.mapa[this.estadoAtual][i].length; j++) {
         if (this.mapa[this.estadoAtual][i][j] == true) {
            if (!this.posicaoValida(this.x+i, this.y+j, tabul)) {
               return true;
            }
            if (tabul[this.x + i][this.y + j] != null) {
               return true;
            }
         }
      }
   }
   return false;
};
```

Bloco.js

```
//Métodos dinâmicos
        ...

this.posicaoValida = function(x, y, tabul) {
   var xv = (x >= 0) && (x < tabul.length);
   var yv = false;
   if (xv) {
      yv = (y >= 0) && (y < tabul[x].length);
   }
   return yv;
};
```

tetris.js

```
//Variáveis e Configurações do jogo

//Eventos de tecla para método onKD
document.onkeydown = onKD;

function onKD(evt) {
        if (evt.keyCode == 38) {
                peca.girarEsquerda(tabuleiro);
        }
        //var tq = peca.encostado(tabuleiro);
        //var aa = tq & 2;
        //var bb = tq & 4;
        //if (evt.keyCode == 39 && bb == 0) peca.x++;
        //if (evt.keyCode == 37 && aa == 0) peca.x--;
        //if (evt.keyCode == 40) avancarJogo();
        desenharTudo();
}

novoJogo();
```

Bloco.js (1/3)

```
//RETORNOS para  encostado:
//Sem encosto - 0; Abaixo = 1; Esquerda = 2; Direita = 4;
this.encostado = function(tabul) {
   var toque = 0;
   for (i = 0; i < this.mapa[this.estadoAtual].length; i++) {
      for (j = this.mapa[this.estadoAtual][i].length - 1; j >= 0; j--) {
         if (this.mapa[this.estadoAtual][i][j] == true) {
            //Encostado no fundo
            if (this.y + j == tabul[0].length - 1) {
               toque = toque | 1;
            } else {
               //Encostado em outro bloco
               if (this.posicaoValida(this.x + i, this.y + i, tabul)) {
                  if (tabul[this.x + i][this.y + j + 1] != null) {
                     toque = toque | 1;
                  }
               }
            }                                          ...
```

Bloco.js (2/3)

```
...

            //Encostado a esquerda
            if (this.x + i - 1 < 0) {
                toque = toque | 2;
            } else {
                //Encostado em outro bloco
                if (tabul[this.x + i - 1][this.y + j] != null) {
                    toque = toque | 2;
                }
            }

...
```

Bloco.js (3/3)

```
...

            //Encostado a direita
            if (this.x + i + 1 >= tabul.length) {
               toque = toque | 4;
            } else {
               //Encostado em outro bloco
               if (tabul[this.x + i + 1][this.y + j] != null) {
                  toque = toque | 4;
               }
            }
         } //Fim do if (this.mapa
      } //Fim do for (j
   } //Fim do for (i
   return toque;
};
```

tetris.js

```
//Variáveis e Configurações do jogo

//Eventos de tecla para método onKD
document.onkeydown = onKD;

function onKD(evt) {
        if (evt.keyCode == 38) {
                peca.girarEsquerda(tabuleiro);
        }
        var tq = peca.encostado(tabuleiro);
        var aa = tq & 2;
        var bb = tq & 4;
        if (evt.keyCode == 39 && bb == 0) peca.x++;
        if (evt.keyCode == 37 && aa == 0) peca.x--;
        //if (evt.keyCode == 40) avancarJogo();
        desenharTudo();
}

novoJogo();
```

tetris.js

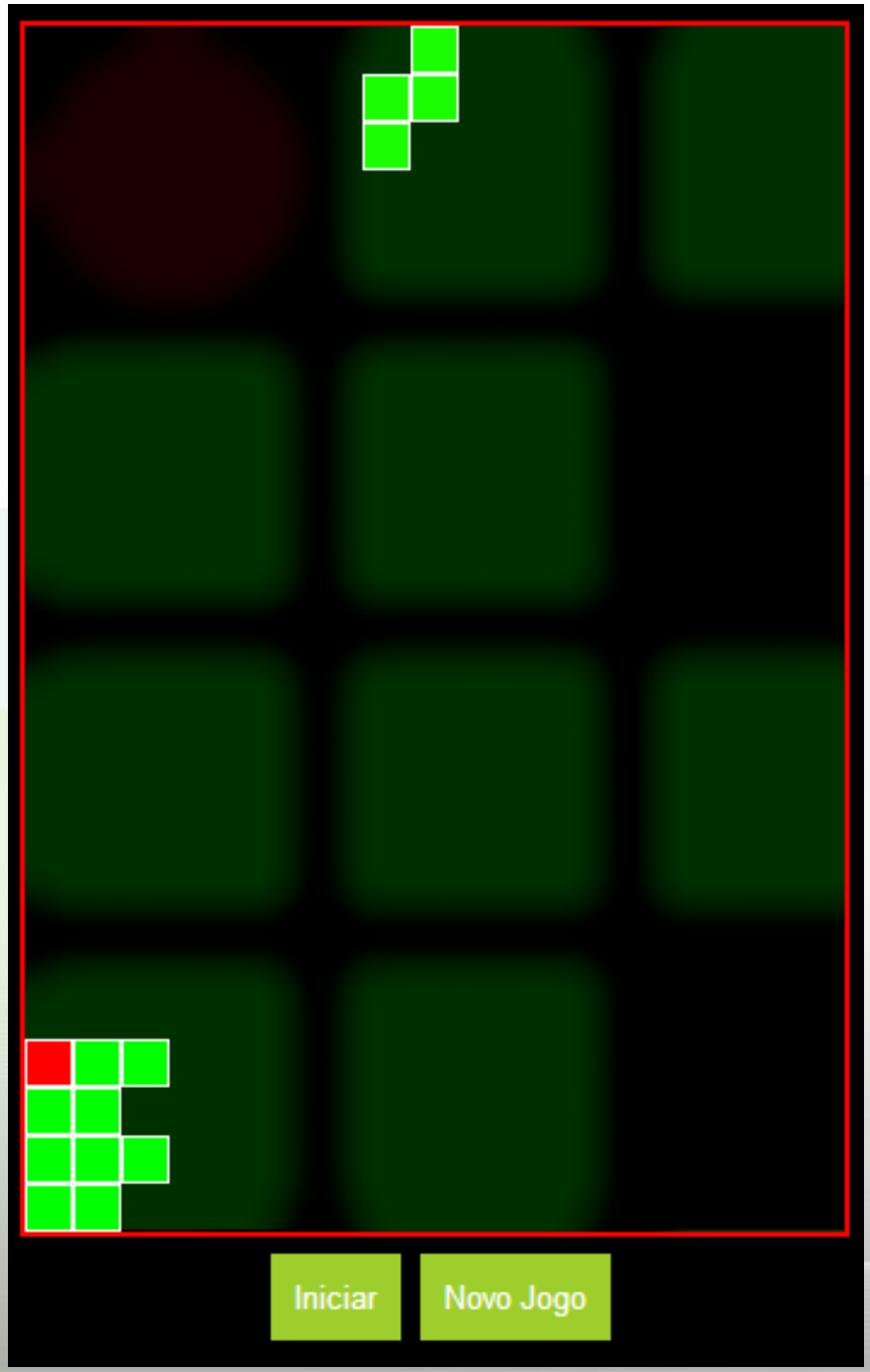
```
...

function avancarJogo() {
        var encostadoFundo = peca.encostado(tabuleiro) & 1;
        if (encostadoFundo == 1) {
                peca.finalizar(tabuleiro);
                novaPeca();
        } else {
                peca.y++;
        }
}

novoJogo();
```

Bloco.js

```
//Métodos dinâmicos
...

this.finalizar = function(tabul) {
   for (i = 0; i < this.mapa[this.estadoAtual].length; i++) {
      for (j = 0; j < this.mapa[this.estadoAtual][i].length; j++) {
         if (this.mapa[this.estadoAtual][i][j] == true) {
            tabul[this.x + i][this.y + j] = this.cor;
          }
      }
   }
};
```

INSTITUTO FEDERAL
SANTA CATARINA

Testar!



GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA

Tetris

MOVIMENTAÇÃO DAS PEDRAS

Movendo com Tempo

- 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 pedra será uma linha;
- Agora precisamos determinar o *intervalo de tempo* que nosso jogo ira usar para fazer cada movimento das pedras;
- Como nosso jogo gira em torno principalmente das pedras, 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;

tetris.js

```
...

var relogio = null;

function pausar() {
        pausa = !pausa;
        if (pausa == false) {
                relogio = setInterval("loopPrincipal()", 400);
                btPausar.innerHTML = "Pausar";
        } else {
                btPausar.innerHTML = "Continuar";
                clearInterval(relogio);
        }
}

novoJogo();
```

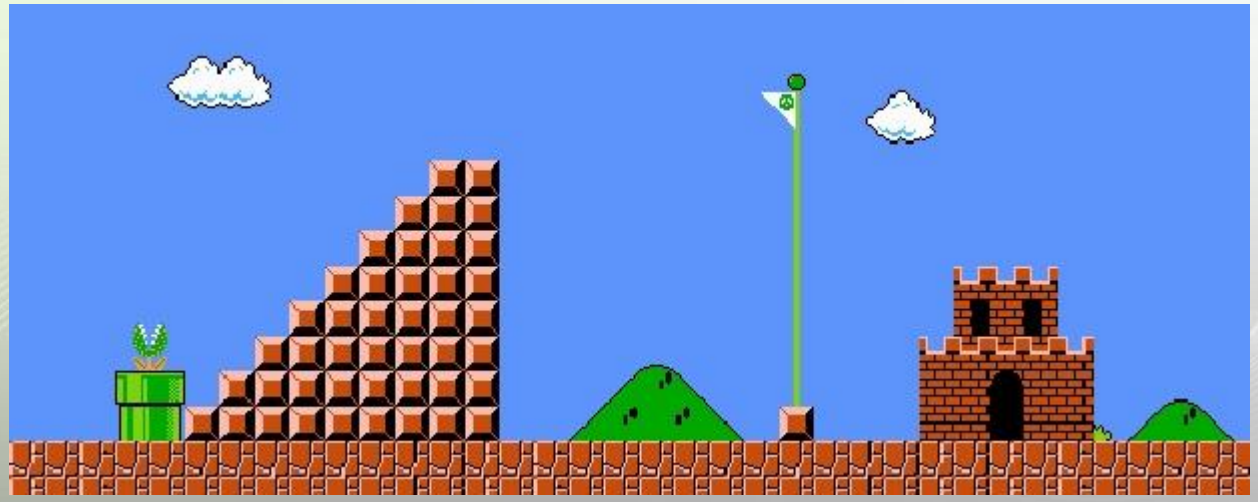
tetris.js

```
...


function loopPrincipal() {
        avancarJogo();
        desenharTudo();
}

novoJogo();
```

Testar

Alterações Restantes

- O que falta alterar?
  - Detecção de linha completa?
  - Fim de jogo?
  - Sons?



tetris.js

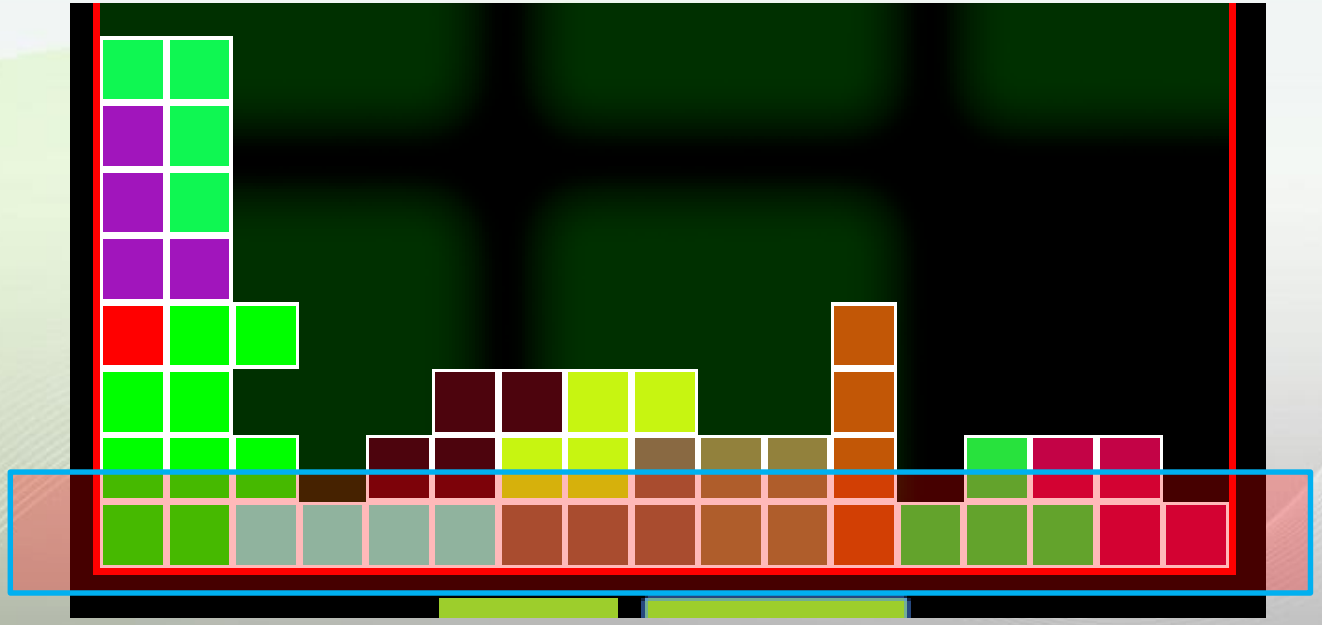
```
...
function loopPrincipal() {
        avancarJogo();
        desenharTudo();
        detectaGameOver();
}

function detectaGameOver() {
        if( peca.detectarColisao(tabuleiro) == true) {
                btPausar.disabled = true;
                clearInterval(relogio);
                alert('Game Over!');
        }
}

novoJogo();
```

Tetris

DETECTANDO LINHAS COMPLETAS

Detectando Linhas

- O principal objetivo do usuário e preencher linhas totalmente, de forma a eliminá-la:
  - Detectar quais linhas estão completas;
  - Eliminar as linhas completas:
    - Mover as demais linhas para baixo!



tetris.js

```
...


function loopPrincipal() {
        avancarJogo();
        verificaLinhas(tabuleiro);
        desenharTudo();
        detectaGameOver();
}


...
```

tetris.js

```
function verificaLinhas(tabul) {
  for (j = tabul[i].length - 1; j > 0; j--) {
    var completa = true;
    for (i = 0; i < tabul.length; i++) {
      if (tabul[i][j] == null) {
        completa = false;
        break;
      }
    }
    if (completa) {
      for (j2 = j; j2 > 0; j2--) {
        for (i = 0; i < tabul.length; i++) {
          tabul[i][j2] = tabul[i][j2-1];
        }
      }
    }
  }
}
```

Tetris

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**:
  - snd_gameover.mp_ e snd_gameover.ogg;
  - snd_gira.mp_ e snd_gira.ogg; e
  - snd_linha.mp_ e snd_linha.ogg.
  - snd_toque.mp_ e snd_toque.ogg.

<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

```
<audio controls id= "sndToque">
  <source src= "snd/snd_toque.mp_" type="audio/mpeg">
  <source src= "snd/snd_toque.ogg" type="audio/ogg">
</audio>
<audio controls id= "sndLinha">
  <source src= "snd/snd_linha.mp_" type="audio/mpeg">
  <source src= "snd/snd_linha.ogg" type="audio/ogg">
</audio>
<audio controls id="sndGameOver">
  <source src= "snd/snd_gameover.mp_" type="audio/mpeg">
  <source src= "snd/snd_gameover.ogg" type="audio/ogg">
</audio>
<audio controls id="sndGira">
  <source src= "snd/snd_gira.mp_" type="audio/mpeg">
  <source src= "snd/snd_gira.ogg" type="audio/ogg">
</audio>

<script src="tetris.js"></script>
```

tetris.js

```
//Recuperando referência dos objetos no documento
var canvas = document.getElementById("tela");
var ctx = canvas.getContext("2d");
var btPausar = document.getElementById("btPausar");
var btNovo = document.getElementById("btNovo");
var sndToque = document.getElementById("sndToque");
var sndLinha = document.getElementById("sndLinha");
var sndGameOver = document.getElementById("sndGameOver");
var sndGira = document.getElementById("sndGira");
```

Prática: Insira os comandos de reprodução nos locais apropriados:

Ex.: sndToque.play();

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 e/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 o jogo não esta em execução, teclas pressionadas são processadas. Corrija este bug.
5. 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 uma exibição da próxima "peça" que irá entrar no jogo;
    4. Adicione teclas de atalho para "Pausa" e para "Novo Jogo";
    5. Substitua as setas por outras teclas de atalho (problema de rolagem);
    6. Ao passar do tempo, aumente velocidade gradualmente;
6. Adicione outros elementos a seu critério;
7. Entregue os arquivos por e-mail ao Professor.