Desenvolvimento de jogos com HTML5 Canvas e JavaScript

W3Schools: http://www.w3schools.com/tags/ref_canvas.asp

Core HTML5 Canvas: http://corehtml5canvas.com

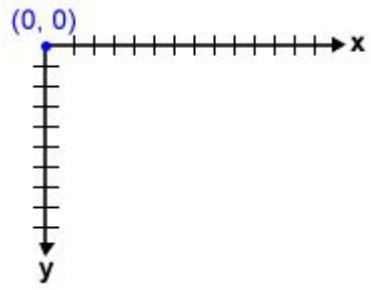
HTML5 2D game development: Introducing Snail Bait: http://goo.gl/Ojvi9T - URL encurtada

Comando do teclado: https://keycode.info/

Ambiente 3D:
http://www.tidbits.com.br/ambientes-3d-usando-html5-e-javascript-com-canvas

OpenGL: https://www.khronos.org/webgl/

Sistemas de coordenadas o Canvas



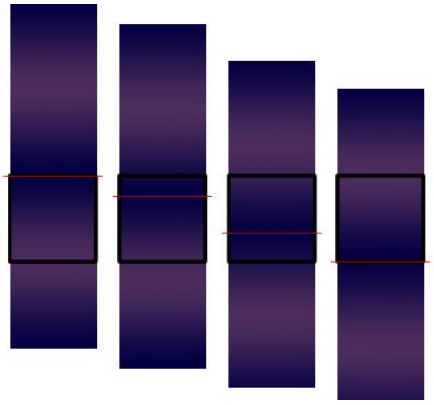
Efeito parallax: http://corehtml5canvas.com/code-live/ch05/example-5.17/example.html

Cenário do jogo de nave (Canvas: 500x500)

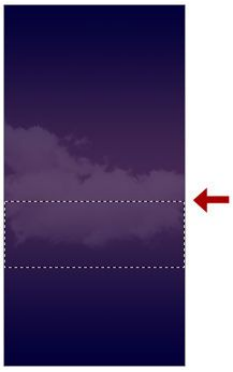
Combinando os elementos do fundo

Os elementos do fundo rolam a velocidades diferentes

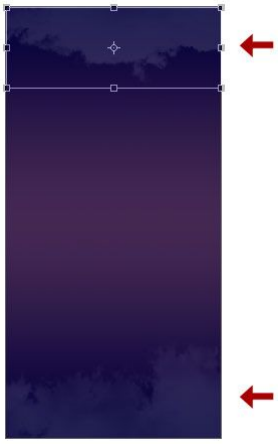
Cada imagem de fundo é desenhada quantas vezes forem necessárias para cobrir a área do jogo. Com imagens maiores que o Canvas, duas vezes são suficientes



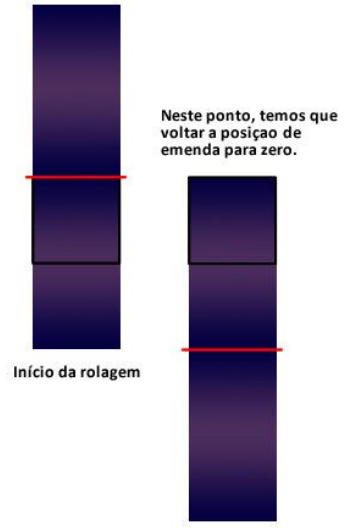
Quebrando uma nuvem pelo centro



Viramos cada metade de cabeça para baixo encaixamos as bordas a imagem



Imagens desceram muito, hora de reposicioná-las



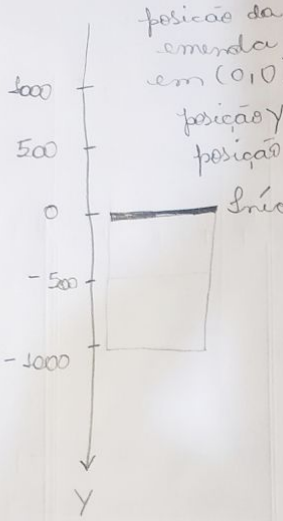
Anotações extras a seguir:

1º cenário

posição da emenda em (0,0)

posiçãoY = 0

posiçãoX = 0

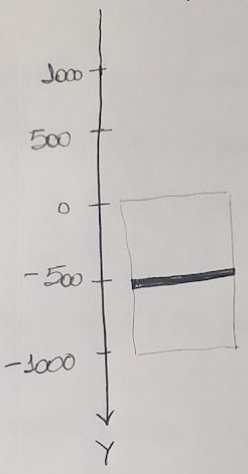


2º cenário

posiçãoY = posiçãoEmenda - altura

= 0 - 500

= - 500

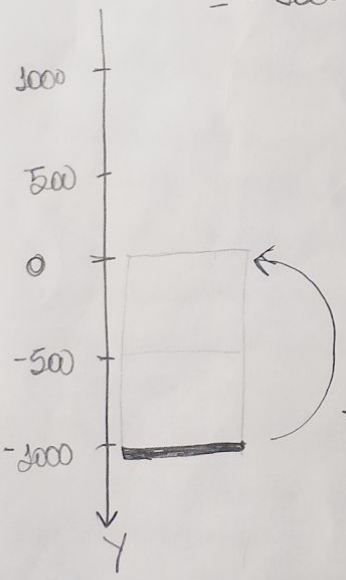


3º cenário

posiçãoY = posiçãoEmenda - altura

= - 500 - 500

= - 1000



Dentro de um loop, vai parecer que está sempre em movimento.

*Quando a posição da emenda > a altura da imagem, a posição da emenda volta a ser zero. Isso fica em loop.

500/2=5

500/2

250

250-5

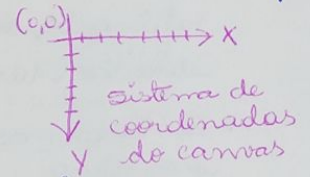
245

1º etapa do trabalho (Projeto orientado a objetos)

Desenvolvimento de jogos com HTML5 Canvas e JavaScript

- Canvas: é uma área retangular em uma página Web onde podemos criar desenhos programaticamente, usando JavaScript.
- JavaScript é a linguagem de programação normal das páginas HTML.


Tela do jogo (500px x 500px)



- Usamos a tag <Canvas> para criar um Canvas em uma página HTML.
- Os atributos width (largura) e height (altura) são obrigatórios, pois são os valores usados na criação da imagem.
- O canvas pode receber dimensões diferentes via CSS, no entanto, seu processamento sempre será feito usando as dimensões informadas na tag. Se as dimensões no CSS forem diferentes, o browser amplia ou reduz a imagem gerada para deixá-lo de acordo com a folha de estilo.
- Para desenhar no canvas, é preciso executar um script após ele ter sido carregado. Neste script, obteremos o contexto gráfico, que é o objeto que realiza de fato as tarefas de desenho no Canvas.
- Referenciamos o canvas e obtemos o contexto gráfico.
- O canvas é referenciado como qualquer elemento em uma página
- O contexto é obtido pelo método 'getContext'.
- Passamos como parâmetro, uma string identificando o contexto desejado
- Sistema de coordenadas do canvas: Para posicionarmos o desenho no canvas, pensamos nele como um enorme conjunto de pontos. Cada ponto possui uma posição horizontal (x) e uma vertical (y)

• Uma vez obtido o contexto gráfico, podemos configurar várias propriedades e chamar nele os métodos de desenho.

→ fillRect(x, y, largura, altura): pinta completamente uma área retangular.
→ strokeRect(x, y, largura, altura): desenha um contorno do retângulo

→ x e y: posição do canto superior esquerdo do retângulo. A partir daí, o retângulo vai para (ou outra imagem) direita (largura) e para baixo.

→ Se trocarmos fillRect por strokeRect, veremos apenas o contorno

→ Podemos configurar propriedades do Contexto:
  * fillStyle: Cor do preenchimento
  * strokeStyle: Cor da linha
  * lineWidht: Espessura da linha em pixels.

• Pathis Conjunto de comandos de desenhos que ficam registrados na memória, aguardando os métodos fill (preencher) e stroke (contornar) serem chamados.

• A melhor forma é desenhar imagens pré-elaboradas em programas gráficos.
→ Carregar imagem utilizando o método Image() do JS.
→ Após criá-lo, apontamos seu atributo src para o arquivo desejado.
→ Aguarda a imagem ser carregada antes de desenhá-la.
→ O objeto image possui o evento onload, que seria disparado automaticamente pelo browser quando o carregamento estiver completo.
→ Estando carregada, a imagem pode ser desenhada através do método "drawImage" do "context". Este método pode ser chamado de duas formas: → JS

* drawImage(imagem, x, y, largura, altura): desenha a imagem inteira, na posição e tamanho especificados
* drawImage(image, xOrigem, yOrigem, larguraOrigem, alturaOrigem, xDestino, yDestino, larguraDestino, alturaDestino)

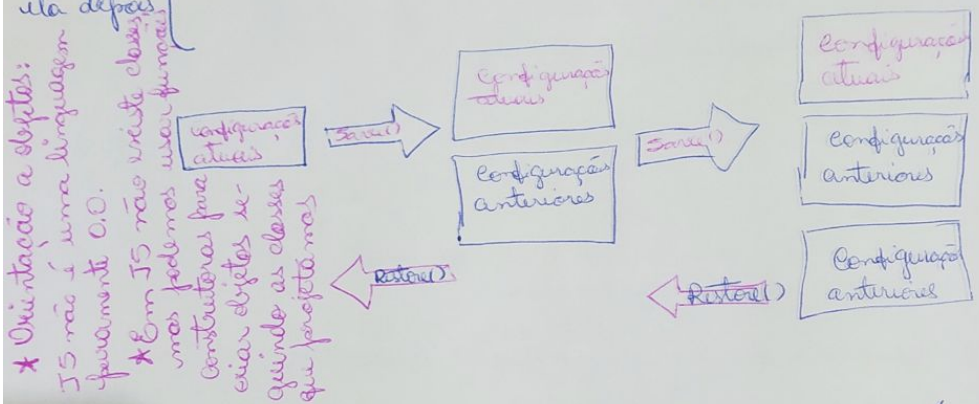
Desenha parte da imagem ←

- Técnica de clipping: Selecionar uma área da imagem original a ser desenhada.
- Recurso importante de context:

Permite guardar configurações e retornar pl ula depois

→ Save: Salva (empilha as configurações)

→ Restore: Retornar as configurações anteriores



* Orientação a objetos: JS não é uma linguagem tecnicamente O.O.
* Em JS não existe classes mas podemos usar funções construtoras para criar objetos semelhantes as classes que conhecemos

- Animação com RequestAnimationFrame: Esse método delega para o browser a tarefa de executar sua animação o mais rápido possível, assim que os recursos do sistema estiverem disponíveis.

→ Solicita a execução de uma função
requestAnimationFrame(minhaFunção)

→ Função de animação

```
function minhaFunção() {
    ...
}
```

→ Solicita o próximo ciclo
requestAnimationFrame(minhaFunção)

- Mexer para direita: Soma um valor a x
- Mexer para a esquerda: Subtrai um valor de x
- Mexer para baixo: soma um valor a y
- Mexer para cima: Subtrai um valor de y
- Mexer na diagonal: alterar valores de x e y para determinada posição.

- Prototype: Objeto associado a uma função construtora. Colocando os métodos neste objeto, todas as instâncias usarão as mesmas cópias de cada método.

Aqui vamos aprender a colocar a função construtora e o prototype de uma classe em um arquivo separado (JS)

- Loop de animação (a classe é responsável por conter o loop)
  - → A animação tem que ser parada e reiniciada em diversos momentos
  - → Vários objetos serão animados em cada ciclo
  - → Objetos podem ser incluídos e excluídos da animação
  - → Os objetos trabalhados dentro de um loop de animação são chamados de Sprites.
    - ↳ Exemplo: Imagem usada no fundo, que pode ser animada.
    - ↳ Eles são criados e eliminados a todo tempo na memória do computador.
  - → A cada ciclo da animação devemos:
    - Limpar tela
    - Atualizar o estado dos sprites
    - Desenhar os sprites
    - Chamar o próximo ciclo da animação