Minicurso A-Frame VR para Ensino de Física e Astronomia

João Teles

2025-05-10

Índice

1 Resumo

2 Realidade Virtual em páginas e aplicativos web

- As vantagens, flexibilidade, alcance e potencial de aplicações web baseadas em HTML5/JS/CSS.
- Realidade Virtual em smartphones versus VR headsets. Focaremos no segundo.
- A API (Interface de Programação de Aplicativos) WebXR utilizada pelos headsets VR nas páginas web.
- Exemplos de Plataformas Web que utilizam WebXR:
  - Three.js
  - Babylon.js
  - A-Frame

3 Principais características do A-Frame

- Filosofia
- Objetivos
- Usos

4 Breve introdução ao HTML/Javascript

- Como se estrutura um arquivo HTML/Javascript básico
- Formas de rodar o arquivo HTML:
  - Página local (editor de texto txt). Usaremos apenas para mostrar os passos iniciais do HTML/Javascript.
  - Servidor web (usaremos o Glitch, mas existem inúmeros outros). Será usado para o desenvolvimento de todo o aplicativo deste minicurso.
  - Servidor local (apenas mostraremos, ao final do minicurso, o caminho que estamos adotando em nossos projetos).

5 Aplicativo Sol-Terra-Lua em VR com o A-Frame

Vamos aprender a usar o A-Frame através de um processo mão na massa de construção de um aplicativo VR para visualização da dinâmica do sistema Sol-Terra-Lua (STL).

6 HTML/Javascript básico

Neste capítulo daremos uma noção bastante introdutória de como criar arquivos HTML/Javascript que podem ser carregados diretamente em um *browser* a partir de um arquivo local *txt* ou a partir de uma hospedagem na internet.

6.1 Estrutura básica de uma página HTML

6.1.1 Elementos HTML

Os elementos HTML são incluídos com palavras reservadas na forma de **tags** HTML: `<bla></bla>` ou somente `<blas>`. Exemplos:

- Títulos de seção: `<h1>Título</h1>`
- Parágrafos: `<p>Conteúdo do parágrafo</p>`
- Botões: `<button>Texto do botão</button>`
- Inputs de texto: `<input>`

1. Copie e cole o conteúdo abaixo em um editor de texto para arquivos txt (bloco de notas no Windows, gedit no Linux, etc).

```
<h1>Minha adorável página HTML</h1>

<p>Nesta página estou incluindo somente alguns elementos HTML para testar o seu uso. Alg

<p>Elementos HTML podem ser aninhados uns dentro dos outros, como neste parágrafo, em qu

<p>Clique nesse botão e veja o que acontece: <button>Clique em mim</button> </p>
```

   Em seguida, salve o arquivo com extensão html. Por exemplo: "arquivomeu.html".

2. Abra o arquivo com o navegador de internet (Chrome, Bing, Firefox, Safari, etc.)
3. Edite o arquivo para incluir elementos novos criados por você. Salve o arquivo, recarregue a página e veja que suas alterações foram atualizadas na página.

6.1.2 Adicionando lógica e programação com Javascript

Para que coisas úteis aconteçam na página HTML, precisamos processar informação. Para isso, existe a linguagem Javascript que é incluída com a tag `<script></script>`.

4. O código abaixo foi extraído de Silva (2016). Salve-o em um arquivo HTML e carregue no seu navegador de internet para ver o resultado.

```
<script>
    var a = 2;
    var b = 3;
    var c = a + b;
    document.write("A soma de " + a + " com " + b + " resulta em " + c);
</script>
```

5. Modifique o script anterior para efetuar a multiplicação de dois números. A operação de multiplicação é implementada com o caractere "*" (asterisco).

6.1.3 Interagindo HTML com Javascript

Para fazer com que elementos HTML específicos sejam modificados via código Javascript é necessário adicionar atributos a eles.

6. No código abaixo, um elemento de parágrafo inicialmente vazio recebe o atributo de identificação `id` e o elemento de botão recebe o atributo `onclick`. Dessa forma, quando o botão é clicado, a função `processarClique()` é chamada em que ela atualiza o texto do parágrafo pela sua identificação.

```
<h1>Página dos apertadores de botão</h1>

<p>
    <button onclick="processarClique()">
        Botão incrível
    </button>
</p>

<p id="resultado"><p>

<script>
    var clicadas = 0;

    function processarClique() {
        clicadas = clicadas + 1;
        document.getElementById("resultado").innerHTML = "Você clicou " + clicadas + " v
```

```
    }
</script>
```

7. Salve o código acima em um arquivo .html e abra no seu navegador de internet para ver o resultado.

Veja uma implementação mais precisa do código acima

```
<h1>Página dos apertadores de botão</h1>

<p>
    <button onclick="processarClique()">
        Botão incrível
    </button>
</p>

<p id="resultado"><p>

<script>
    var clicadas = 0;

    function processarClique() {
        clicadas = clicadas + 1;

        if (clicadas > 1) {
            document.getElementById("resultado").innerHTML = "Você clicou " + clicadas + "
        }
        else {
            document.getElementById("resultado").innerHTML = "Você clicou " + clicadas + "
        }
    }
</script>
```

6.2 Estrutura completa de uma página HTML

Uma página HTML completa, em geral, vai ser dividida em três arquivos:

- `index.html`: é o arquivo principal e o primeiro a ser carregado pelo navegador de internet.
- `script.js`: é o arquivo em que a programação Javascript fica melhor separada e organizada.

- `style.css`: é o arquivo de estilos que altera a aparência e às, vezes, a funcionalidade dos elementos HTML.

Em que o arquivo `index.html` terá uma estrutura semelhante a esta:

```
<!DOCTYPE html>

<html>

    <!-- Aqui é incluído o preâmbulo com as configurações da página -->
    <head>

        <!-- Este é o título (opcional) da página -->
        <title>Hello World!</title>

        <!-- Aqui é carregado a arquivo de estilos -->
        <link rel="stylesheet" href="/style.css" />

        <!-- Aqui é carregado o arquivo Javascript -->
        <script src="/script.js" defer></script>

    </head>

    <!-- Aqui é incluído todo o HTML e, caso queira, o Javascript também -->
    <body>

        <h1>Minha página web completa</h1>

        <p>Meu sucinto parágrafo.</p>

        <button>Meu inútil botão</button>

        <!-- Você também pode colocar código Javascript aqui -->
        <script>

            var x = 3;
            var y = 2;
            var z = x + y;

        </script>

    </body>
```

```
</html>
```

Podemos criar arquivos txt para cada um dos três arquivos e abrir o arquivo HTML no navegador de internet como vínhamos fazendo até aqui. Entretanto, a partir de agora começaremos a colocar nossas páginas diretamente na Internet.

Aprender mais sobre HTML/Javascript/CSS

6.3 Disponibilizando páginas web diretamente na Internet com Glitch

Existem diversas plataformas gratuitas que permitem a hospedagem de páginas web. Entre elas, podemos destacar:

- Replit
- CodeSandbox
- GitHub Pages
- Glitch

Neste minicurso usaremos o Glitch.

Glitch é uma plataforma online para desenvolvimento e hospedagem de aplicações web, permitindo edição colaborativa em tempo real diretamente no navegador. Páginas criadas no Glitch são disponibilizadas quase que instantaneamente em um endereço de internet de acesso público.

Vamos começar a partir de um projeto básico que criamos para este minicurso.

8. Acesse: https://glitch.com/edit/#!/fishy-foil-technosaurus

Ao acessar esse projeto, você deverá observar a tela abaixo.

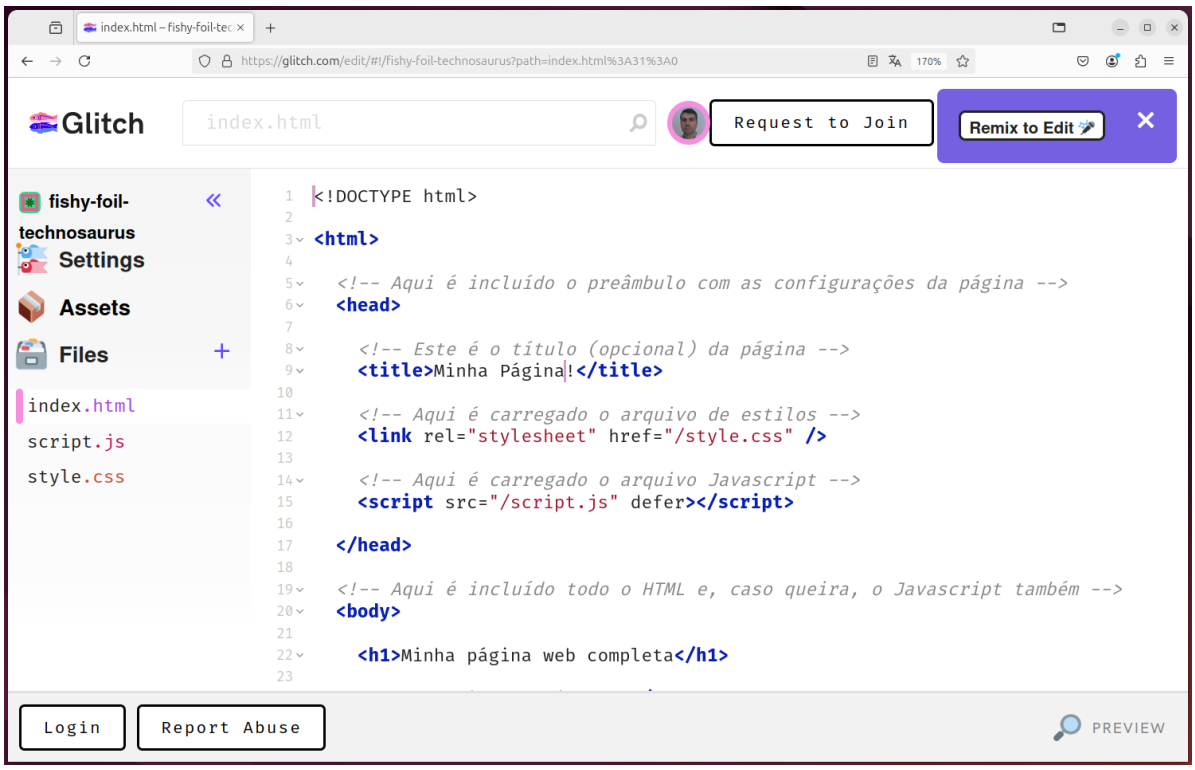

Figura 6.1: Print da tela do projeto fishy-foil-technosaurus no Glitch

A tela apresenta basicamente dois painéis:

- O painel de arquivos e configurações à esquerda.
- O painel de código à direita que mostra o conteúdo do arquivo escolhido no painel de arquivo.

No canto inferior direito há o botão de *preview*, que renderiza (gera/compila) a página index.html que será mostrada para quem acessar a página no endereço público do projeto.

9. Agora, antes de começar a editar e trabalhar em projetos no Glitch, é importante criar uma conta nele, pois isso permitirá acessar os projetos posteriormente.

10. Após criar a conta, acesse novamente a edição do projeto fishy-foil-technosaurus e clique no botão *Remix do Edit* localizado no canto superior direito.

Tarefa

11. A partir do *fishy-foil-technosaurus* remixado, implemente o botão contador de clicadas que fizemos anteriormente no editor de texto.

Clicando em *Settings* no painel de arquivos e configurações você pode alterar o nome do projeto.

Para finalizar, com o botão *Share* você pode compartilhar o seu projeto de três formas diferentes:

- Página web final (*Live site*): https://fishy-foil-technosaurus.glitch.me
- Edição do código [*Code*]: https://glitch.com/edit/#!/fishy-foil-technosaurus
- Apresentação do Projeto [*Project Page*]: https://glitch.com/~fishy-foil-technosaurus

Aprender mais sobre o Glitch

7 O projeto Sol-Terra-Lua

Neste capítulo, aprenderemos o uso básico do A-Frame por meio do desenvolvimento de um projeto para ensino de Astronomia.

No projeto Sol-Terra-Lua (abrev. STL) iremos:

- Criar as esferas solar, terrestre e lunar no espaço tridimensional em realidade aumentada.
- Texturizar a superfície das esferas com os mapas dos respectivos astros.
- Estabelecer os tamanhos e posições dos astros de forma pertinente aos objetivos didáticos de uma atividade educacional.
- Implementar o movimento de rotação de cada astro em torno do próprio eixo.
- Definir o plano orbital da Terra e da Lua.
- Implementar o movimento orbital da Terra e da Lua.
- Dar interatividade aos objetos celestes.

7.1 Elementos HTML do A-Frame

Todo o desenvolvimento do projeto será realizado no Glitch.

1. Faça o login na sua conta do Glitch e remixe o projeto aframe-minicurso-vr

Repare que o projeto está bastante vazio, com apenas o arquivo `index.html` e sem qualquer código Javascript explícito. Entretanto, temos dentro do `head` a inclusão do arquivo online:

`https://aframe.io/releases/1.6.0/aframe.min.js`.

É esse arquivo, acessado remotamente, que disponibilizará o núcleo de funcionalidades do A-Frame.

1. O elemento HTML básico e obrigatório do A-Frame é o *a-scene*, o qual temos que incluir dentro do elemento *body*:

```
<body>
    <a-scene>
    </a-scene>
</body>
```

Dentro do *a-scene* todos os elementos espaciais do projeto serão incluídos. Esses elementos recebem o nome de *a-entity* (entidades). Cada entidade, por sua vez, terá uma série de propriedades, chamadas de *components* (componentes). Por fim, cada componente terá atributos que definirão as aparências e funcionalidades da entidade.

Esse tipo de estruturação segue o modelo ECS (Entity-Component-System) que é alternativo ao modelo de classes e objetos. Ele se opõe ao sistema de hierarquização da programação orientada a objetos ao permitir maior modularização e inter-operacionalização dos componentes que podem ser aplicados diretamente em diferentes entidades.

Eis um exemplo de uma estrutura ECS no A-Frame:

a-scene:

- **a-entity** `id = "carro"`:
  - `rodas = "quantidade : 4; aro: 20"`
  - `material = "cor: azul; metalicidade: 0.5"`
- **a-entity** `id = "moto"`:
  - `rodas = "quantidade : 2; aro: 40"`
  - `material = "cor: vermelho; metalicidade: 0.8"`

Figura 7.1: Ilustração das *Entidades* carro e moto com seus *componentes* e *atributos* específicos (feita com DALL-E)

Nesse exemplo, *carro* e *moto* são ***entidades*** distintas, mas *rodas* e *material* são ***componentes*** que podem ser compartilhados pelas diferentes entidades. Esses componentes irão se diferenciar por possuírem valores diferentes para os seus ***atributos*** (e.g. rodas com `quantidade : 2` ou `quantidade : 4`) e por estarem atrelados a diferentes entidades. Uma das vantages dessa abordagem ECS é que eu preciso programar o componente *rodas* somente uma vez e ele pode

ser usado em diferentes veículos.

Entidades podem ser aninhadas. Por exemplo, ao invés de tratar as rodas como sendo componentes, poderíamos considerar mais apropriado implementá-las na forma de uma entidade:

a-scene:

- **a-entity** `id = "fusca"`:
  - **a-entity** `id = "rodas_do_fusca"`:
    - `material = "cor: preto; metalicidade: 0.1"`
    - `dimensoes = "aro: 20; largura: 10; quantidade: 4"`
- **a-entity** `id = "moto"`:
  - `rodas = "quantidade : 2; aro: 40"`
  - `material = "cor: vermelho; metalicidade: 0.8"`

Vamos então criar nossa primeira entidade: o planeta Terra.

2. Coloque a entidade abaixo dentro do elemento a-scene:

```
<a-entity id="terra" geometry="primitive: sphere" material="color: blue"></a-entity>
```

Se fizermos o preview lateral para tentarmos observar o resultado, provavelmente não veremos nada. Isso ocorre, pois não estamos usando os óculos de realidade virtual. A melhor forma de interagir com a página gerada na tela do computador é escolhendo a opção *Preview in a new window*. Na página que foi aberta, devemos entrar no modo de inspeção (*Visual inspector*) ao digitar a combinação de teclas `<ctrl>` + `<alt>` + `i` ou `<ctrl>` + `<option>` + `i`.

Cada botão do mouse quando mantido apertado permite rotacionar, transladar e aproximar/afastar a visão da cena. Ao clicar em uma entidade é possível também aferir e alterar os valores dos seus atributos. É possível até mesmo criar entidades novas e atribuir componentes no *Visual inspector*.

Voltando à Terra, vamos "dar um talento" em nosso *pale blue dot*.[1] Para isso, vamos usar uma imagem planificada da superfície da Terra e mapeá-la na esfera. Por sorte, o A-Frame faz isso muito diretamente para a gente. Vejamos como.

[1] Expressão cunhada pelo astrônomo e divulgador científico Carl Sagan ao descrever a imagem da Terra capturada pela sonda Voyager 1, a uma grande distância. No retrato, o planeta aparece como um minúsculo ponto azul. Sagan utilizou essa imagem para refletir sobre a humildade e a responsabilidade da humanidade em relação ao único lar conhecido.

7.2 Carregando arquivos externos em nosso projeto STL

O A-Frame possui um elemento chamado `<a-asset>` para pré-carregar todos os recursos (imagens, áudios, vídeos, etc) que sejam usados na página. Existem duas formas para fazer isso:

- Carregando o recurso dentro do Glitch via *Files* no menu lateral de arquivos.
- Indicando o endereço de internet (URL) do recurso caso ele esteja disponível.

No momento, vamos usar a segunda opção.

3. Clique com o botão direito do mouse sobre o mapa da Terra abaixo e copie o seu link.

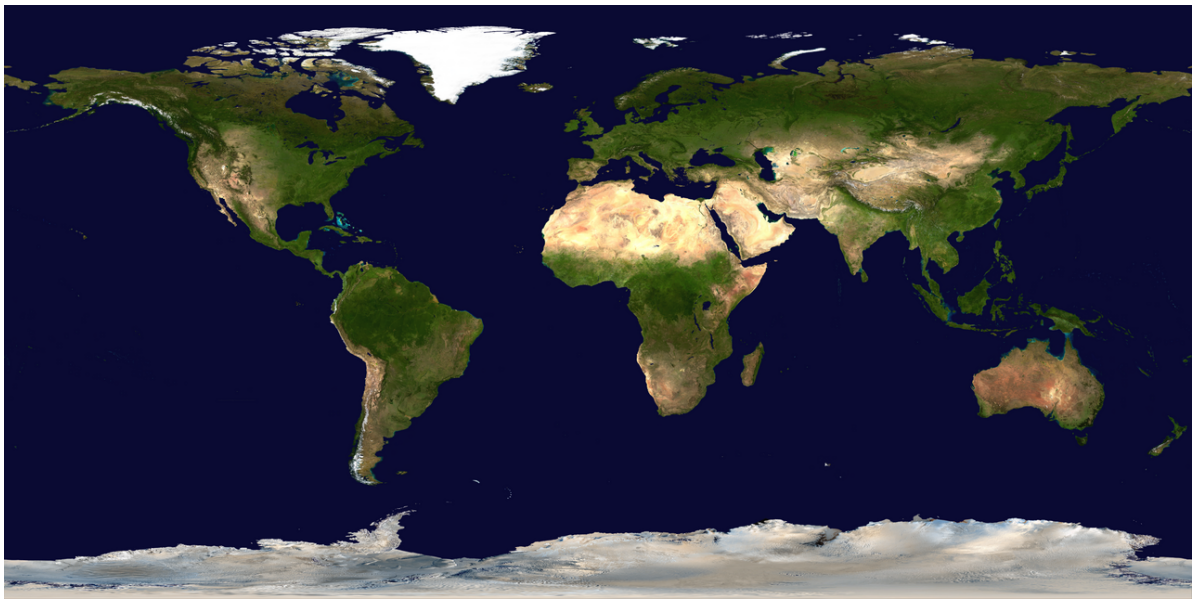

Figura 7.2: https://en.wikipedia.org/wiki/World_map#/media/File:Blue_Marble_2002.png

4. Dentro do `<a-scene>` inclua o elemento:

```
<a-assets>
    <img id="mapa_terra" src="substitua_isto_pelo_endereço_da_imagem">
</a-assets>
```

5. Em seguida substitua o atributo *color* por *src* no componente *material* da entidade `terra`, usando como valor a `id` do mapa da Terra:

```
<a-entity id="terra"
    geometry="primitive: sphere"
    material="src: #mapa_terra">
</a-entity>
```

8 Referências

Silva, Nelson Canzian da. 2016. *Física com JavaScript*. Depto de Física - UFSC. https://canzian.prof.ufsc.br/fisicacomjavascript/fjs-00-tudo.pdf.