WebVR con A-Frame

Realidad virtual web

Germán Álvarez
Lead Instructor Web Development

A-Frame es una librería de Javascript sin dependencias, de código abierto, creada para diseñar **experiencias de realidad virtual interactivas e inmersivas en el navegador**.

Compatible con Vive, Rift, Windows Mixed Reality, Daydream, GearVR, Cardboard o Oculus Go, entre otros, y apta para diseñar experiencias AR.

Basado en HTML, proporciona un marco **sistema-entidad-componente** con una estructura declarativa, escalable, reusable e intuitiva.

#INTRODUCCIÓN

Incluir A-Frame a través de archivo local, CDN o NPM:

```
<head>
 <script src="https://aframe.io/releases/0.9.2/aframe.min.js"></script>
</head>
```

Para desarrollo en local es necesario habilitar un servidor sobre el que correr la aplicación.

#INSTALACIÓN

más información

Una facilidad para el desarrollo es el inspector integrado de A-Frame (ctrl + option + i), que permite inspeccionar, manipular y/o exportar la escena a tiempo real con una accesibilidad y usabilidad excelentes.

#INSPECTOR

más información

A-Frame requiere disponer de todas las **entidades** de la escena encapsuladas en el **sistema**, representado por la etiqueta `<a-scene>`

Asimismo, proporciona un juego de entidades por defecto (**primitivas**): box, sphere, plane, cylinder, ring, image, sky, light, camera, cursor...

```
<a-scene>
  <a-box></a-box>
  <a-sphere></a-sphere>
  <a-cylinder></a-cylinder>
  <a-plane></a-plane>
  <a-sky></a-sky>
</a-scene>
```

#PRIMITIVOS

más información

Las entidades primitivas son customizadas mediante **componentes** en forma de atributos, clasificados como *mesh*, *geometry, position, material*, *light...*

```
<a-scene>
  <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
  <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>
  <a-cylinder position="1 0.75 -3" radius="0.5" height="1.5" color="#FFC65D"></a-cylinder>
  <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4" color="#7BC8A4"></a-plane>
  <a-sky color="#ECECEC"></a-sky>
</a-scene>
```

#COMPONENTES

más información

Aunque en realidad, las *entidades primitivas* son entidades edulcoradas sintácticamente:

```
<a-scene>
  <a-box width="3" color="red"></a-box>
</a-scene>
```

Es idéntico a:

```
<a-scene>
  <a-entity geometry="primitive: box; width: 3" material="color: red"></a-entity>
</a-scene>
```

#ENTIDADES

más información

A-Frame está basado en un paradigma Sistema-Entidad-Componente (**ECS**), siendo:

- **Sistemas** - opcionales, proporcionan alcance global, administración y servicios para las clases de componentes: la lógica tras los mismos.
- **Entidades** - objetos contenedores donde integrar componentes, como `<a-entity>`
- **Componentes** - módulos de datos reutilizables que pueden adjuntarse a entidades para proporcionar apariencia, comportamiento y / o funcionalidad, como los atributos incluídos en cada entity: `<a-entity color='red'>`

#ECS

más información

Algunos ejemplos de **entidades** resultantes de combinar **componentes** serían:

- Caja: geometría + posición + material
- Lámpara: geometría + posición + material + luz + sombra
- Señal: geometría + posición + material + texto
- Pelota: geometría + posición + material + físicas + velocidad

```
<a-entity geometry="primitive: sphere; radius: 1.5"
          light="type: point; color: white; intensity: 2"
          material="color: white; shader: flat; src: textures/wood.jpg"
          position="0 0 -5">
</a-entity>
```

#ECS

más información

El componente `animation` permite transicionar **valores** de un componente (posición, rotación…) o de **propiedades** de un componente (color, intensidad…) con aceleraciones, loops, dirección...

```
<a-entity
   geometry="primitive: sphere"
   material="color: #EFEFEF; shader: flat; src: textures/wood.jpg"
   position="-4 0 -5"
   light="type: spot; intensity: 10;"
   animation="property: position; to: -4 -1 -5; dur: 1000; easing: easeInOutQuad; dir: alternate; loop: true;">
</a-entity>
```

#ANIMACIONES

más información

Pudiendo disponer de un componente de animación por entidad:

```
<a-entity
  geometry="primitive: sphere"
  material="color: #EFEFEF; shader: flat; src: textures/wood.jpg"
  position="-4 0 -5"
  animation="property: position; to: 5 2 -10; dur: 3000; easing: easeInOutQuad; dir: alternate; loop: true;">
</a-entity>

<a-entity
  geometry="primitive: sphere"
  material="color: #EFEFEF; shader: flat; src: textures/wood.jpg"
  position="-4 -5 -5"
  animation="property: position; to: -4 2 10; dur: 3000; easing: easeInOutQuad; dir: alternate; loop: true;">
</a-entity>
```

#ANIMACIONES

Es posible acumular diversos componentes de animación sobre una misma entidad, siempre prefijados por `animation__`

```
<a-entity
  geometry="primitive: sphere"
  material="color: #EFEFEF; shader: flat; src: textures/fabric.jpg"
  position="0 5 -10"
  rotation="0 0 0"
  animation__posit="property: position; to: 0 0 -10; dur: 1000;loop: true; dir: alternate"
  animation__rotat="property: rotation; to: 360 0 0; dur: 2000; loop: true; easing: easeInOutQuad; ">
</a-entity>
```

#ANIMACIONES

más información

El componente *light* permite aportar realismo a la escena simulando metalicidad, puntos de luz, reflejo, luz ambiental…:

```
<a-entity geometry="primitive: sphere; radius: 1.5"
  material="color: white; shader: flat; src: textures/marble.jpg" position="5 10 -20"
  animation="property: position; to: 5 2 -20; dur: 3000; easing: easeInOutQuad; dir: alternate; loop: true;"
  light="type:point; color: white; intensity: 1"></a-entity>

<a-entity geometry="primitive: sphere; radius: 1.5"
  material="color: white; shader: flat; src: textures/marble.jpg"  position="-5 10 -20"
  animation="property: position; to: -5 2 -20; dur: 3000; easing: easeInOutQuad; dir: alternate; loop: true; delay: 1000"
  light="type:point; color: white; intensity: 1"></a-entity>

<a-sky height="2048" radius="30" color="black" theta-length="90" width="2048"></a-sky>
<a-plane material="color:#111;" rotation="-90 0 0" height="100" width="100" position="0 0 0"></a-plane>
```

#ILUMINACIÓN

más información

Crear interacción supone analizar cómo A-frame resuelve la **programación basada en eventos**. Es posible producir un evento en la escena de dos formas:

- A través de un componente de tipo cursor:

```
<a-entity cursor="rayOrigin: mouse"></a-entity>
```

- A través de interacción por mirada *(gaze-based interactions)*, propia de los entornos VR que carecen de controladores físicos.

```
<a-camera>
   <a-cursor></a-cursor>
</a-camera>
```

#INTERACCIÓN

más información

Un ejemplo de detección de evento por parte de un objeto sería mediante la propiedad startEvents:

```
<a-entity
  geometry="primitive: sphere"
  material="color: #EFEFEF; shader: flat; src: textures/fabric.jpg"
  position="0 5 -10" rotation="0 0 0"
  animation="property: position; to: 0 0 -10; dur:1000; loop:true; dir: alternate; startEvents: click;">
</a-entity>

<a-entity cursor="rayOrigin: mouse"></a-entity>
```

#INTERACCIÓN

Igualmente válida para los eventos de mirada, que pueden hacer uso del componente *fuse* avitando interacciones no deseadas:

```
<a-entity
  geometry="primitive: sphere"
  material="color: #EFEFEF; shader: flat; src: textures/fabric.jpg"
  position="0 5 -10" rotation="0 0 0"
  animation="property: position; to: 0 0 -10; dur:1000; loop:true; dir: alternate; startEvents: click;">
</a-entity>


 <a-camera>
    <a-cursor fuse="true" fuse-timeout="1500"></a-cursor>
 </a-camera>
```

#INTERACCIÓN

más información

La forma más interesante de crear interacción es la programática, registrando un componente a este efecto:

```
AFRAME.registerComponent('cursor-listener', {
  init: function () {
    this.el.addEventListener('click', () => {this.el.setAttribute('material', 'color', 'red')
    this.el.setAttribute('animation', 'property: rotation; to: 120 120 120; dur: 3000; easing: easeInOutQuad')
    })
}})
```

Garantizando la reusabilidad al integrarla en cualquier entidad:

```
<a-entity id="box" cursor-listener geometry="primitive: box" material="color: blue" position="0 4 -5"></a-entity>
```

#INTERACCIÓN

más información

Además de los componentes estándar, es posible registrar componentes propios en la aplicación o incorporar componentes externos creados por la comunidad en forma de paquetes de NPM indexados y filtrados en el buscador de componentes A-Frame.

También existen infinidad de modelos 3D descargables en plataformas como Sketchfab, Clara.io o Archive3d, entre otros.

#RECURSOS

más información

Gracias por vuestra atención

Germán Álvarez
Lead Instructor Web Development