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

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 son customizadas mediante **componentes** en forma de atributos, clasificados como *mesh*, geométricos, de iluminación y de escena.

```
<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

En realidad, las entidades primitivas son entidades edulcoradas sintácticamente, ya que *todos* los componentes de una escena son **entidades**.

```
<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: glow.jpg"
          position="0 0 -5">
</a-entity>
```

#ECS

más información

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

Las animaciones incluyen aceleraciones, loops, dirección...

```
<a-entity
   geometry="primitive: sphere"
   material="color: #EFEFEF; shader: flat"
   position="-4 0 -5"
   light="type: spot; intensity: 10;"
   animation="property: position; to: -4 -0.2 -5; dur: 1000; easing: easeInOutQuad; dir: alternate; loop: true;">
</a-entity>
```

#ANIMACIONES

más información

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"
   position="-4 0 -5"
   light="type: spot; intensity: 10; color: red;"
   animation__posit="property: position; to: -4 -0.2 -5; dur: 1000;"
   animation__light="property: light; from: red; to: white; dur: 1000; dir: alternate; loop: true;">
</a-entity>
```

#ANIMACIONES

más información

La propiedad `startEvents` permite lanzar una animación a raíz de un evento sobre la entidad:

```
<a-entity
   geometry="primitive: sphere"
   material="color: #EFEFEF; shader: flat"
   position="-4 0 -5"
   light="type: spot; intensity: 10; color: red;"
   animation="property: position; to: -4 -0.2 -5; dur: 1000; startEvents: click;">
</a-entity>
```

#ANIMACIONES

más información

La **programación basada en eventos** es, además, resuelta de dos maneras diferentes:

- A través del componente `event-set`, indicando como propiedad `_event` el nombre del evento.
- Registrando un evento / juego de eventos como un componente, y aplicándolo a cualquier entidad.

#EVENTOS

más información

El componente `event-set` (externo) permite crear interacción asociando eventos a entidades:

```
<script src="https://unpkg.com/aframe-event-set-component@3.0.3/dist/aframe-event-set-component.min.js"></script>
[...]
<a-box
  position="-1 0.5 -3"
  color="#4CC3D9"
  event-set="_event: mouseenter; color: #8FF7FF"></a-box>
```

Pudiendo acumular tantos componentes como se requieran:

```
<a-box
  position="-1 0.5 -3"
  color="#4CC3D9"
  event-set__enter="_event: mouseenter; color: #8FF7FF"
  event-set__leave="_event: mouseleave; color: #4CC3D9"></a-box>
```

#EVENTOS

más información

También es posible **registrar un componente** en forma de juego de eventos:

```
AFRAME.registerComponent('handle-events', {
  init: () => {
     let el = this.el
     el.addEventListener('mouseenter', () => el.setAttribute('color', '#24CAFF'))
     el.addEventListener('click', () => el.setAttribute('scale', {x: 2, y: 1})
   }
  });
```

Pudiendo integrarlos en cualquier entidad:

```
<a-box color="#EF2D5E" position="0 1 -4" handle-events></a-box>
```

#EVENTOS

más información

Para los contextos VR, carentes de ratón, se hace uso de las **interacciones por mirada** (*gaze-based interactions)*.

Estas interacciones son aptas para cualquier dispositivo VR y requieren una entidad de cámara que contenga una entidad de cursor, permitiendo disparar eventos de tipo `mouseenter` y `mouseleave`:

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

#INTERACCIÓN

más información

En estos contextos, carentes de controlador, no existe la opción de disparar un evento de tipo `click`.

Este evento se reproduce a través de una confirmación por permanencia (*fuse*), pudiendo indicar el tiempo de ejecución mediante un componente `fuse-timeout`:

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

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