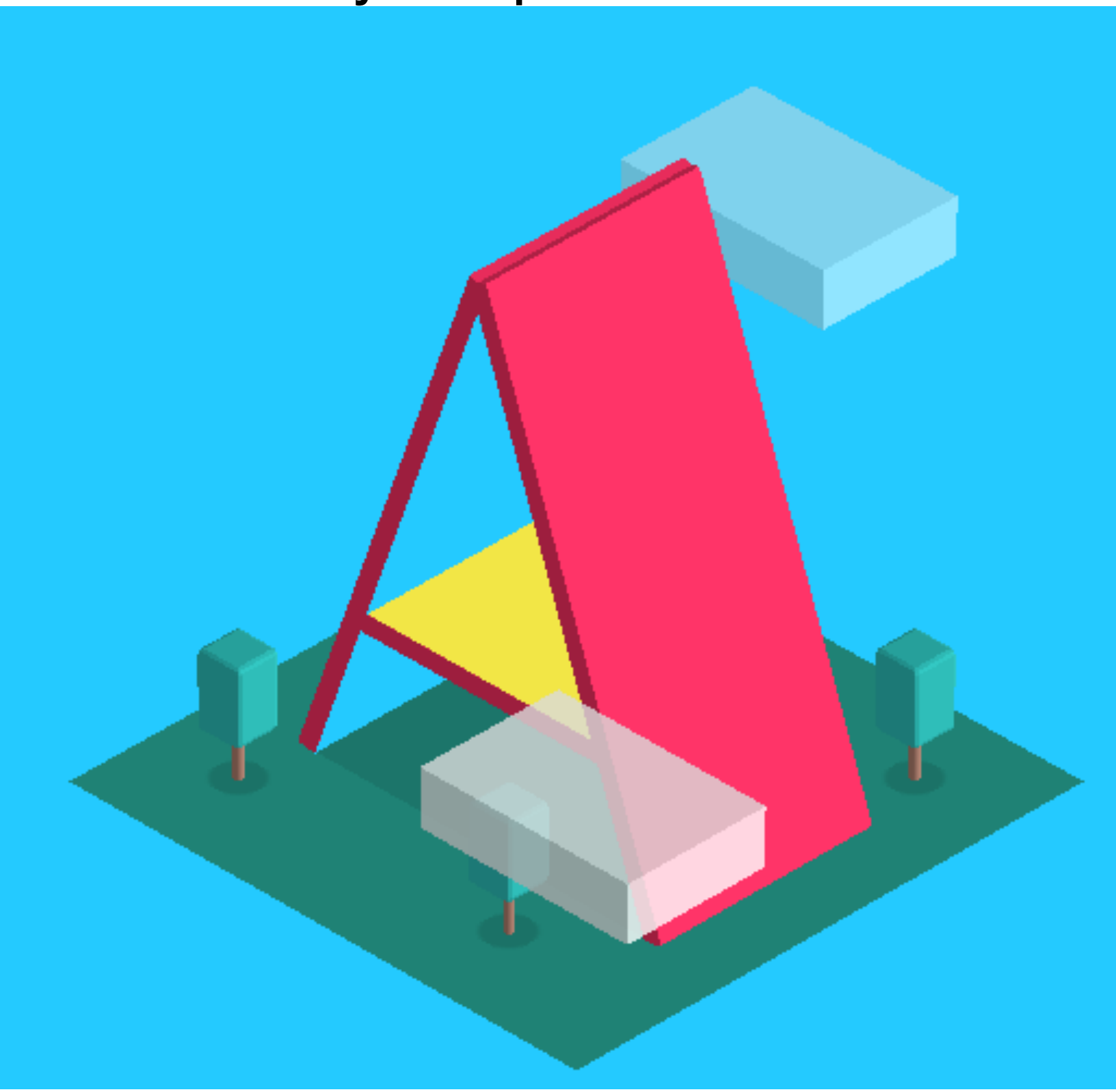
3D y VR para la Web



HTML
JavaScript
A-Frame

3D y VR para la Web

Manual de acompañamiento al taller de 3D y realidad virtual con JavaScript y A-Frame

Hugo Zapata (hugozap@gmail.com)

Plataforma Web

A-Frame

Librería creada por Mozilla

https://aframe.io/

HTML

Declaración de escena

JS

JavaScript

Comportamiento (Lógica)

WEBVR

Soporte Realidad Virtual

Configuración Inicial

Se puede crear un nuevo proyecto con unas pocas líneas. (Archivo index.html)

```
<html>
 <head>
  <script src="https://aframe.io/releases/0.5.0/aframe.min.js"></script>
 </head>
 <body>
  <a-scene>
   <a-box depth="2" height="2" width="2" rotation="14 15 15" color="red"></a-box>
   <a-entity camera look-controls wasd-controls scale="1 1 1" position="0 1 15"></a-entity>
  </a-scene>
 </body>
</html>
```

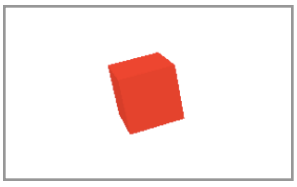
También se puede descargar el archivo y guardarlo en la misma carpeta. En ese caso solo se pone el nombre del archivo.

1. Referencia a la librería a-frame
2. Declaración de la escena (a-scene) que contiene todos los elementos 3D
3. Declaración de un elemento (Caja)

    - ancho (width) = 2
    - alto (height) = 2
    - profundidad (depth) = 2

4. Declaración de la **cámara** y su posición inicial

Si todo funcionó bien deberías ver algo como esto:



Nota: Todas las páginas Web tienen estos elementos.

html: contenedor principal de la página
head: sección donde se declaran las librerías necesarias.
body: contenido de la página

Inspector

El inspector de a-frame es una herramienta útil
para examinar las propiedades de los elementos

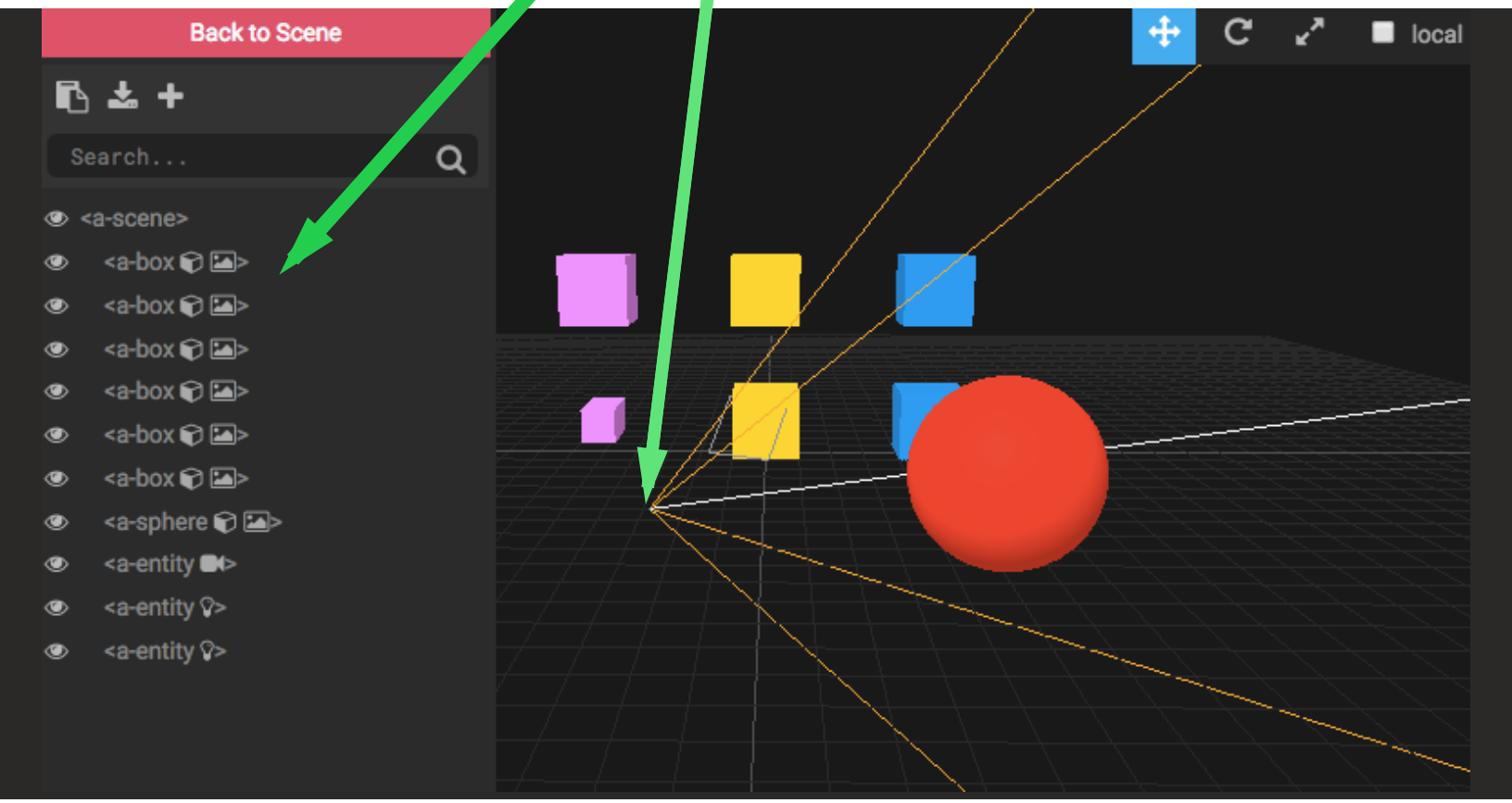
Abrir el inspector:

Presionar la combinación

Control+Alt+i

Elementos de la escena

Cámara



Algunas primitivas de A-frame

Caja: (a-box)

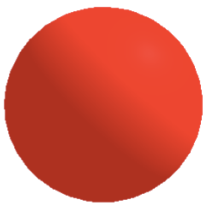
a-box

```
<!-- Caja básica. -->
<a-box color="tomato" depth="2" height="4" width="0.5"></a-box>
```

Algunos atributos

Atributo	Valor por defecto	
color	#FFF	
height	1	Alto
width	1	Ancho
depth	1	Profundidad
src	None	Referencia al material (textura) a aplicar

Esfera (a-sphere)



a-sphere

```
<!-- Esfera básica. -->
<a-sphere color="yellow" radius="5"></a-sphere>
```

Atributo	Valor por defecto	
color	#FFF	
radius	1	
src	None	Referencia al material (textura) a aplicar

Plano (a-plane)

a-plane

```
<!-- Plano básico. -->
 <a-plane color="#50E3C2" height="20" width="20"></a-plane>
```

Los planos son superficies sin profundidad. Se pueden usar como “piso” de la escena. Al igual que las otras figuras, pueden tener texturas con el atributo **src**.

```
<!-- Plano con textura paralela al piso. -->
 <a-plane src="#ground" height="100" width="100" rotation="-90 0 0"></a-plane>
```

Para que la textura sea visible debe estar declarada en la sección de recursos (assets)

```
<a-assets>
    <img id="ground" src="ground.jpg">
  </a-assets>
```

Otras primitivas:

Cono

```
<a-cone  radius-bottom="2" radius-top="0.5"
color=“#4990E2” position="-5 1 0"></a-cone>
```

Anillo

```
<a-ring color="teal" radius-inner="1" radius-
outer="2" position="0 10 0"></a-ring>
```

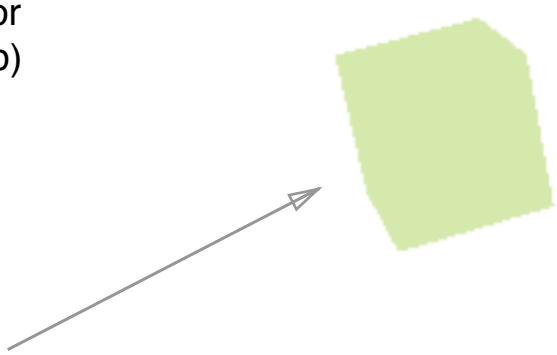
“Torus” (Dona)

```
<a-torus color="#43A367" arc="270" radius="5"
radius-tubular="0.1" position="5 10 0"></a-torus>
```

Agregando Luces a nuestra escena.

Luz Ambiente

Afecta todos los materiales. El color de la caja era blanco pero se ve del mismo color de la luz ambiente verde claro -> (#cfeeab)

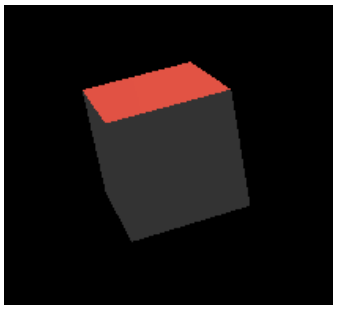


```
<a-entity light="type: ambient; color: #cfeeab"></a-entity>
```

Luz Direccional (Como el sol)

Luz que viene de una dirección. La propiedad `position` indica el vector de donde proviene la luz.

En el ejemplo el color de la luz es roja.



La dirección se declara como un hijo (también es entity).

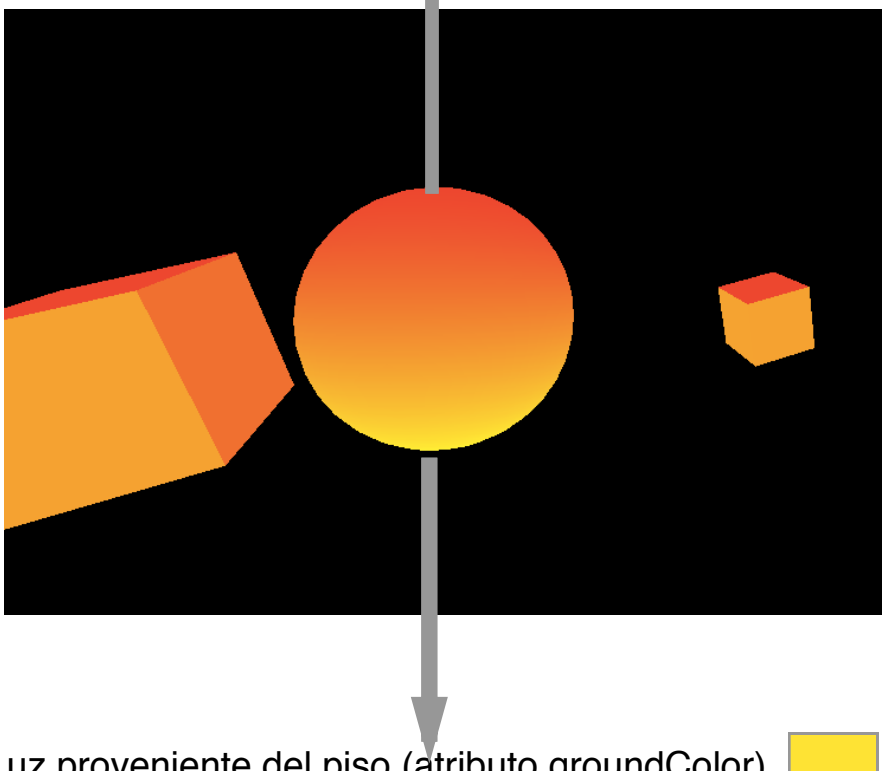
```
<a-entity
    light="type: directional; color: red; intensity:0.8"
    target="#direccionLuz" position="1 10 0">

    <a-entity id="direccionLuz" position="0 0 -1"></a-entity>
</a-entity>
```

Agregando Luces a nuestra escena.

Luz Hemisferio.

1) Luz de la parte de arriba (atributo color)

Este tipo de luz permite asignar 2 colores:

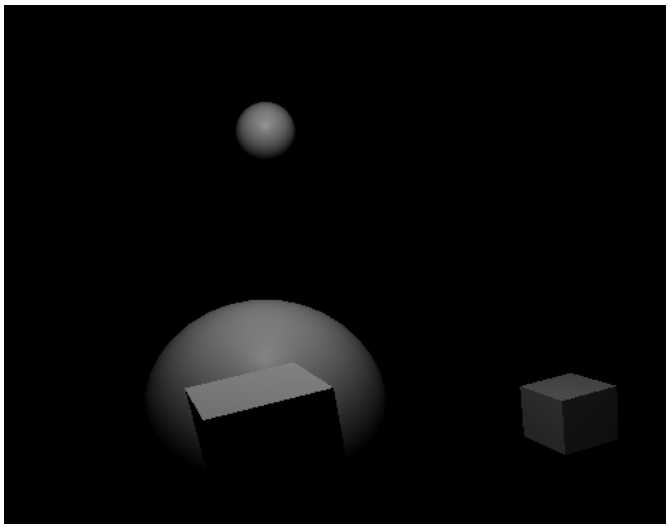


```
<a-entity light="type: hemisphere; color: red; groundColor: yellow; intensity: 1" position="0 10 0">
</a-entity>
```

2) Luz proveniente del piso (atributo groundColor)

Luz tipo Punto.

Similar a un bombillo.
Tiene posición X,Y,Z



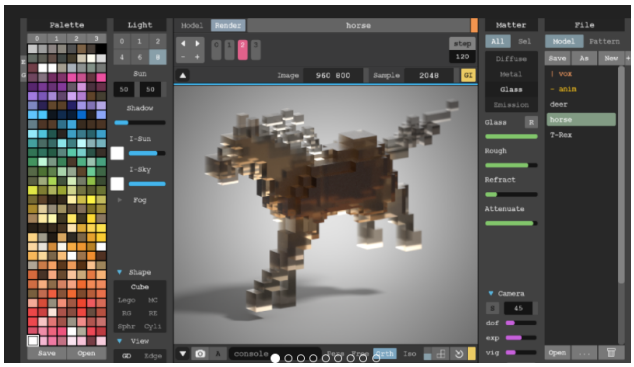
```
<a-entity
light="type: point; intensity: 0.75; distance: 50; decay: 2"
        position="0 10 10">
</a-entity>
```

Agregando Modelos 3D a nuestro Proyecto.

Existen diferentes formatos 3D soportados. Para cada formato puede cambiar el método para importarlo

Archivos .PLY

Se pueden crear con el software **MagicaVoxel** (gratuito).
- Son de muy buena calidad
- Cuidado con incluir muchos (puede afectar rendimiento)



Archivos (DAE , OBJ)

- No contienen información de sombras
- Menor tamaño
- Más rápidos que PLY (depende del número de caras)
- No se ven tan bien como los PLY.

```
  <a-assets>
    <a-asset-item id="house" src="scene_house7.bake.ply"></a-asset-item>
    <a-asset-item id="house2" src="obj_house2a.dae"></a-asset-item>
  </a-assets>
  <a-entity ply-model="src: #house"
   position="-10 0 0"
   rotation="-90 0 0"
   scale="0.1 0.1 0.1">
  </a-entity>
  <a-collada-model src="#house2" position="0 0 0"></a-collada-model>
```

house es un archivo .ply (exportado por MagicVoxel) Incluye sombras.
Necesita ser rotado 90 grados sobre el eje X

Uso de JavaScript para manipular nuestra escena

Si deseamos usar JavaScript para modificar nuestra escena, se recomienda crear componentes de a-frame.
A continuación vamos a crear un componente básico como ejemplo.

Plantilla básica de un componente a-frame

```
AFRAME.registerComponent('rotador-colores', {
  schema: {
      //Aquí van los parámetros de componente
  },
  init: function () {
      //Aquí va la lógica del componente
      //Esta función se llama al iniciar el componente
  }
});
```

Ejemplo de un componente que cambia de colores

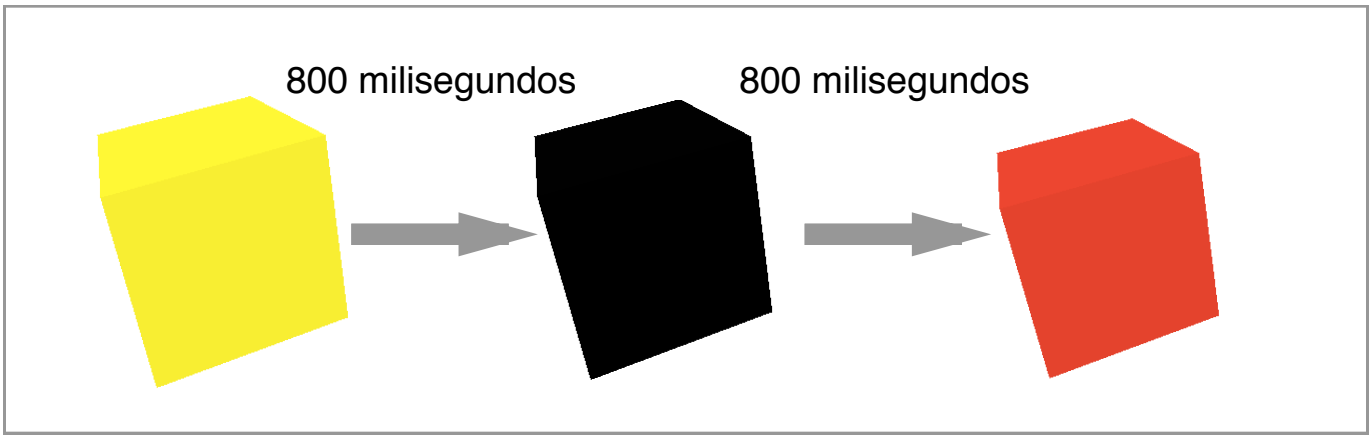
```
AFRAME.registerComponent('rotador-colores', {
  schema: {
      colores: {type:'array', default:['yellow','red']},
      duracion: {type:'number', default:1000}
  },
  init: function () {
      this.indice = 0
    setInterval( ()=>{
            this.el.setAttribute('color', this.data.colores[this.indice])
            this.indice++;

            //Si llegamos al final , volver a 0
            if(this.indice >= this.data.colores.length) {
                this.indice = 0;
            }
    }, this.data.duracion )
  }
});
```

Usando nuestro componente

En nuestra escena podemos aplicar el componente a alguno de los elementos
Aquí lo vamos a aplicar a una caja (a-box)

```
<a-box rotador-colores="colores:yellow,black,red;duracion:800"
color="#6885d0" depth="2" height="2" width="2" rotation="14 15
15"></a-box>
```

Al refrescar nuestra escena la caja cambiará de color cada 800 milisegundos



Animaciones (Usando componente básico a-animation)

Con las animaciones se puede animar el cambio de un valor desde un valor inicial (from) hasta un valor final (to)
A continuación un ejemplo de la animación de la intensidad de la luz desde su valor original (1) hasta el valor final (3)

a-animation

```
<a-light intensity="1">
  <a-animation attribute="intensity" to="3"></a-animation>
</a-light>
```

Duración y repetición:

A continuación se configura la animación para que nunca termine (repeat="indefinite")

```
<a-sphere radius="2" color="red" position="5 1 0">
      <a-animation attribute="scale"
          dur="1000"
          fill="both"
          to="2 2 2"
          repeat="indefinite"></a-animation>
   </a-sphere>
```

La mayoría de propiedades numéricas se pueden animar
- posición, rotación, escala
También es posible animar propiedades booleanas (true/false) así como transiciones de colores

Animaciones (Usando un componente especializado)

Existe un componente más poderoso para realizar animaciones. No es parte de la librería pero se puede descargar aquí:

https://github.com/ngokevin/kframe/tree/master/components/animation/

Animación con componente kframe-animation

```
<a-cylinder color="#F55" radius="0.1"
            animation="property: color; dir: alternate; dur: 1000;
                       easing: easeInSine; loop: true; to: #5F5"
            animation__scale="property: scale; dir: alternate; dur: 200;
                       easing: easeInSine; loop: true; to: 1.2 1 1.2"
            animation__yoyo="property: position; dir: alternate; dur: 1000;
                             easing: easeInSine; loop: true; to: 0 2 0">
  </a-cylinder>
```

La manera de animar con este componente es diferente. Agregamos atributos “animation” y las propiedades de la animación se declaran de manera similar a estilos CSS

"property: color; dir: alternate; dur: 1000;
easing: easeInSine; loop: true; to: #5F5"

Este valor de animación nos dice lo siguiente:
- La propiedad a animar es ‘color’
- La dirección es ‘alterna’ (inicio-fin-inicio-fin…)
- Es infinita: loop: true
- El valor final es el color #5F5