



**GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA
TELECOMUNICACIÓN**

Curso Académico 2020/2021

Trabajo Fin de Grado

**VISUALIZACIÓN DE LA ESTRUCTURA DE
DOCUMENTOS HTML EN REALIDAD VIRTUAL**

Autor : Alberto Sánchez-Seco Úbeda

Tutor : Dr. Jesús M.González Barahona

Trabajo Fin de Grado

VISUALIZACIÓN DE LA ESTRUCTURA DE DOCUMENTOS HTML EN REALIDAD VIRTUAL

Autor : Alberto Sánchez-Seco Úbeda

Tutor : Dr. Jesús M.González-Barahona

La defensa del presente Proyecto Fin de Carrera se realizó el día de
de 2019, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2019

*Dedicado a
mi familia,
por creer en mí siempre*

Agradecimientos

En primer lugar, quiero dar las gracias a mis padres y resto de mi familia, por su apoyo incondicional, por seguirme en cada una de las decisiones que he tomado, no juzgarme con severidad en algunas de ellas, pese a estar equivocado, y por ayudarme a continuar hasta el final cuando las fuerzas escaseaban.

A todos y cada uno que han formado parte de mi etapa universitaria, que son muchos, os estaré eternamente agradecido por los momentos compartidos y la experiencias vividas.

A Pedro, por las incontables horas que he pasado junto a ti en el campus y fuera de él, desde el primer hasta el último año, por ayudarme cada vez que lo necesitaba, no rendirte conmigo nunca y estar ahí siempre en cada momento del camino. Casi todos los recuerdos imborrables que me llevo de esta etapa son junto a ti. Gracias.

A David, por hacerme ver que hay veces que la gente que conoces en la biblioteca de una Universidad puede convertirse en familia. Tú eres prueba de ello, gracias por ser como un hermano para mí desde entonces.

A Álvaro, Rebeca, Ávila y Carlos. Por convertiros en un sostén para mí en los momentos más difíciles de esos años. Por vuestro apoyo y por siempre sacarme una sonrisa cuando lo necesitaba. Sin vosotros, no hubiera llegado hasta aquí y me llevo vuestra amistad incondicional para siempre.

A los profesores que me ayudaron a avanzar en esta etapa, a crecer como persona y motivarme para aprender. En especial, a Jesús Barahona, por su infinita paciencia para hacer el proyecto, por sus consejos, y por siempre tenerme en cuenta. Gracias.

Resumen

Este proyecto consiste en la visualización de un documento HTML en realidad virtual, mediante entidades en 3D.

Todas las páginas web están construidas a partir del lenguaje y estándar denominado HTML. Para poder representar documentos HTML, existe un modelo estándar denominado DOM que proporciona una estructura basada en árbol con la que poder representar de forma jerárquica los elementos que componen la página HTML. Toda esta representación se lleva a cabo en el propio navegador web utilizando las APIs JS del navegador, basadas en el lenguaje JavaScript.

Para la visualización en 3D, se ha utilizado el framework A-Frame, que se trata de un framework web de código abierto para crear experiencias de realidad virtual(VR). Se basa en código HTML, de modo que su éxito se basa en su sencillez, ya que a partir de simples etiquetas, A-Frame se encarga de generar los modelos 3D, la configuración de la realidad virtual y los controles predeterminados.

Se ha utilizado como modelo un documento HTML con distintas etiquetas de varios tipos, y herencia en árbol DOM para poder ver con mayor claridad cada una de las entidades que componen el documento. Todo ello permite una aproximación a visualizar de forma clara los elementos de un HTML, abriendo así camino para poder depurar código HTML en realidad virtual.

Summary

This project consists of the visualization of an HTML document in virtual reality, using 3D entities.

All web pages are built from the language and standard called HTML. To be able to represent HTML documents, there is a standard model called DOM that provides a tree-based structure with which to represent the elements that make up the HTML page in a hierarchical way. All this representation is carried out in the web browser itself using the browser's JS APIs, based on the JavaScript language.

For 3D visualization, the A-Frame framework has been used, which is an open source web framework to create virtual reality (VR) experiences. It is based on HTML code, so its success is based on its simplicity, since from simple labels, A-Frame is responsible for generating the 3D models, the virtual reality settings and the default controls.

I used as a model an HTML document with different tags of various types, and inheritance in the DOM tree to be able to see more clearly each of the entities that make up the document. All this allows an approach to clearly visualize the elements of an HTML, thus opening the way to be able to debug HTML code in virtual reality.

Índice general

| | |
|--|-----------|
| 1. Introducción y Objetivos | 1 |
| 1.1. Contexto | 1 |
| 1.2. Objetivo general | 2 |
| 1.3. Objetivos específicos | 3 |
| 1.4. Estructura de la memoria | 3 |
| 2. Estado del arte | 5 |
| 2.1. HTML | 5 |
| 2.2. JavaScript | 8 |
| 2.3. A-Frame | 10 |
| 2.4. WebXR | 14 |
| 2.5. Otras Tecnologías | 15 |
| 2.5.1. Three.js | 15 |
| 2.5.2. CSS | 16 |
| 2.5.3. html2canvas | 17 |
| 2.5.4. LateX | 18 |
| 2.5.5. GitHub | 19 |
| 2.6. Bootstrap | 19 |
| 3. Desarrollo del Proyecto | 21 |
| 3.1. Introducción | 21 |
| 3.2. Desarrollo e implementación | 22 |
| 3.2.1. Sprint 0 | 22 |
| 3.2.2. Sprint 1 | 25 |

| | | |
|-----------|--------------------------------------|-----------|
| 3.2.3. | Sprint 2 | 28 |
| 3.2.4. | Sprint 3 | 30 |
| 3.2.5. | Sprint 4 | 32 |
| 4. | Resultados | 39 |
| 4.0.1. | Manual de Usuario | 39 |
| 5. | Conclusiones | 51 |
| 5.1. | Consecución de objetivos | 51 |
| 5.2. | Planificación temporal | 52 |
| 5.3. | Aplicación de lo aprendido | 53 |
| 5.4. | Lecciones aprendidas | 54 |
| 5.5. | Trabajos futuros | 55 |
| | Bibliografía | 59 |

Índice de figuras

| | |
|--|----|
| 2.1. Ejemplo Documento HTML | 6 |
| 2.2. Ejemplo Documento HTML en navegador | 6 |
| 2.3. Ejemplo Estructura HTML | 7 |
| 2.4. Acceso DOM a través de JS | 9 |
| 2.5. Ejemplo Documento HTML con A-Frame | 11 |
| 2.6. Ejemplo Escena A-Frame | 11 |
| 2.7. Ejemplo Escena móvil A-Frame | 12 |
| 2.8. Ejemplo Inspector escena A-Frame | 13 |
| 2.9. Ejemplo Entidades A-Frame | 13 |
| 2.10. Ejemplo WebXR | 14 |
| 2.11. Diagrama Three.js | 15 |
| 2.12. Ejemplo CSS | 16 |
| 2.13. Ejemplo CSS | 17 |
| 2.14. Ejemplo uso html2canvas | 17 |
| 2.15. Ejemplo html2canvas | 18 |
| 3.1. Código escena básica | 23 |
| 3.2. Escena básica Sprint 0 | 23 |
| 3.3. Escena básica 2 Sprint 0 | 24 |
| 3.4. Escena básica 3 Sprint 0 | 24 |
| 3.5. Componentes A-Frame | 25 |
| 3.6. Escena Histograma | 26 |
| 3.7. Entidad Cursor | 26 |
| 3.8. Resultado Gráfica | 27 |

| | |
|---|----|
| 3.9. Ejemplo HTML | 27 |
| 3.10. Ejemplo recursividad DOM | 28 |
| 3.11. Ejemplo HTML Sprint 2 | 28 |
| 3.12. Sistema Coordenadas A-Frame | 29 |
| 3.13. Escena A-Frame | 30 |
| 3.14. Ejemplo HTML Sprint 3 | 31 |
| 3.15. Ecuaciones Coordenadas A-Frame | 31 |
| 3.16. Escena A-Frame Sprint 3 | 32 |
| 3.17. Inspector A-Frame | 32 |
| 3.18. Propiedades caja HTML | 33 |
| 3.19. Ancho y Alto caja HTML | 33 |
| 3.20. Elemento h1 HTML | 34 |
| 3.21. Escena elemento h1 | 35 |
| 3.22. Inspector A-Frame | 35 |
| 3.23. Documento HTML prueba html2canvas | 36 |
| 3.24. Escena A-Frame con html2canvas | 37 |
| 3.25. Inspector A-Frame | 37 |
| 3.26. Documento HTML prueba 2 html2canvas | 37 |
| 3.27. Escena A-Frame con html2canvas e imagen | 38 |
| 3.28. Inspector A-Frame | 38 |
| 4.1. Etiqueta div Banner HTML final | 40 |
| 4.2. Etiqueta div Contenido HTML final | 40 |
| 4.3. Etiqueta div imagen Uni HTML final | 40 |
| 4.4. Representacion cajas HTML demo 01 | 41 |
| 4.5. Documento HTML demo 01 | 41 |
| 4.6. Escena demo 01 | 42 |
| 4.7. Inspector Escena demo 01 | 42 |
| 4.8. Inspector 2 Escena demo 01 | 43 |
| 4.9. Inspector 3 Escena demo 01 | 43 |
| 4.10. Representacion cajas HTML demo 02 | 44 |

| | |
|---|----|
| 4.11. Documento HTML demo 02 | 44 |
| 4.12. Escena demo 02 | 45 |
| 4.13. Escena Inspector 1 demo 02 | 45 |
| 4.14. Escena Inspector 2 demo 02 | 46 |
| 4.15. Escena Inspector 3 demo 02 | 46 |
| 4.16. Representacion cajas HTML demo 03 | 47 |
| 4.17. Documento HTML demo 03 | 47 |
| 4.18. Escena demo 03 | 48 |
| 4.19. Escena Inspector 1 demo 03 | 48 |
| 4.20. Escena Inspector 2 demo 03 | 49 |
| 4.21. Escena Inspector 3 demo 03 | 49 |
| 5.1. Planificación temporal | 53 |
| 5.2. Página Web con Plugin GreaseMonkey | 56 |

Capítulo 1

Introducción y Objetivos

HTML es fundamental para el desarrollo de páginas web, tanto como estándar como lenguaje. Como estándar, dispone de una serie de APIs en el mismo navegador que nos permite, entre otras muchas cosas, acceder y manipular los elementos que lo componen. Además, como lenguaje, es sobre el que está construido las páginas web a las que accedemos en el navegador web.

Una forma más visual y clara de trabajar con documentos HTML es a través de la realidad virtual, utilizando el framework A-Frame, que nos va a permitir visualizar los elementos que componen la estructura del HTML como entidades en 3D, interactuando así en un entorno de realidad virtual.

Este proyecto lo que hace es, partiendo de la estructura en árbol del documento HTML que nos proporciona el DOM, visualizar cada uno de esos elementos como entidades en 3D en un entorno de realidad virtual, de forma que se observa de una manera más clara y visual la estructura jerárquica que compone el HTML.

1.1. Contexto

HTML como lenguaje puede llegar a tener una estructura compleja con numerosas etiquetas y elementos para proporcionar funcionalidad y apariencia a la página. A través del DOM (Document Object Model), nos permite facilitar la representación estructurada del documento. El DOM es la estructura del documento HTML, transformando todas las etiquetas que componen el HTML, anidadas unas dentro de otras, en un árbol de nodos relacionados entre sí. El

DOM es la estructura básica a la hora de programar con documentos HTML, ya que lenguajes como JavaScript nos permite modificar su estructura de forma dinámica, añadiendo etiquetas, eliminando otras,.. a través de un objeto llamado Document, que representa el árbol DOM de la página.

El W3C DOM estándar forma la base del funcionamiento del DOM en muchos navegadores modernos. Varios navegadores ofrecen extensiones más allá del estándar W3C, hay que ir con extremo cuidado al utilizarlas en la web, ya que los documentos pueden ser consultados por navegadores que tienen DOMs diferentes.

La realidad virtual (RV) es un entorno de escenas y objetos en 3D de apariencia real, generado mediante tecnología informática, que crea en el usuario la sensación de estar inmerso en él, mediante el uso de dispositivos concebidos para conseguir ese efecto, como gafas o casco de realidad virtual. Para la visualización de entornos o escenas en 3D, existe una especificación estándar implementada en JavaScript llamada WebGL, que es soportada por la mayoría de navegadores, de manera que el proyecto puede funcionar en cualquier dispositivo. Además, el uso de APIs de dispositivos para navegadores web como son WebXR nos permite tener acceso a escenas de realidad virtual y realidad aumentada basadas en la web. Este incremento de posibilidades y usos con la RV ha hecho que su popularidad vaya en aumento y sea ya un instrumento con muchos medios hoy en día.

Mediante A-Frame, un framework de código abierto basado en HTML, podemos crear experiencias RV que podemos disfrutar desde nuestro navegador web. Gracias a su compatibilidad con la mayoría de bibliotecas y marcos web existentes, a su herramienta de inspector visual que se puede invocar en el navegador desde cualquier escena y a su versatilidad y facilidad de uso, ha llevado a que A-Frame pueda ser utilizado para numerosos propósitos y aplicaciones.

La idea del proyecto es unir todos estos conceptos, para poder representar ese árbol de nodos de un documento HTML como entidades A-Frame y poder visualizarlo en realidad virtual.

1.2. Objetivo general

El principal objetivo a la hora de llevar a cabo este proyecto es la posibilidad de visualizar el árbol de elementos DOM de un documento HTML, el cuál puede llegar a ser bastante complejo, en un entorno de realidad virtual a través del navegador, de manera clara y atractiva visualmente,

además de promover la realidad virtual como instrumento útil para poder visualizar de una manera clara documentos o entornos complicados.

1.3. Objetivos específicos

- Construir la aplicación en un navegador y utilizando JavaScript, ya que es el lenguaje más utilizado para acceder a los datos y elementos del documento HTML a través del navegador, mediante APIs construidas dentro del navegador para crear dinámicamente contenido HTML, generar escenas en 3D, etc.
- Utilizar el framework A-Frame para generar experiencias en un entorno de realidad virtual.
- Realizar un proyecto multiplataforma capaz de visualizarse en distintos entornos a través del navegador web como ordenadores, teléfonos móviles, etc.
- Ayudar a acercar conceptos como el árbol de elementos de un documento HTML a personas ajenas a él, de una forma atractiva como es la RV.
- Realizar una aplicación intuitiva y atractiva, usando gafas de realidad virtual para navegar por todas las entidades que componen el documento HTML.

1.4. Estructura de la memoria

A continuación se detalla la estructura de la memoria.

- **Capítulo 1: Introducción y Objetivos** En este capítulo se desarrolla brevemente el contexto en el que se ha realizado el proyecto, así como los objetivos que se pretenden alcanzar con la realización del mismo.
- **Capítulo 2: Estado del arte.** En este capítulo se realiza una descripción de alto nivel de las diferentes tecnologías utilizadas en la realización del proyecto.
- **Capítulo 3: Desarrollo del Proyecto.** En este capítulo se describe el desarrollo del proyecto incluyendo su arquitectura y modelo de datos.

- **Capítulo 4: Resultados.** En este capítulo se describen y analizan los resultados obtenidos.
- **Capítulo 5: Conclusiones.** En este capítulo se indican las conclusiones a las que se ha llegado una vez finalizado el proyecto.

Capítulo 2

Estado del arte

El proyecto se basa fundamentalmente en documentos HTML sobre el que construir la estructura de visualización, el lenguaje JavaScript para manejar y modificar dinámicamente los elementos del documento y el framework A-Frame, que nos permite generar escenas y experiencias en realidad virtual. También son remarcables librerías o tecnologías fundamentales para la visualización en 3D como son Three.js o WebXR, además de CSS para la apariencia del HTML. A continuación se detalla más en profundidad las tecnologías utilizadas en el desarrollo del proyecto.

2.1. HTML

HTML¹, siglas en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), es un estándar y lenguaje de marcado que se utiliza para el desarrollo de páginas web. Fue creado por Tim Berners-Lee entre finales de los años 80 y principios de los 90.

Como lenguaje, se trata de un lenguaje de marcas, para indicar la estructura del documento que utilizan etiquetas y es el componente más básico de la Web, definiendo el significado y estructura del contenido Web.

A continuación se muestra un ejemplo muy sencillo de un documento HTML con un par de etiquetas que muestra un título y un pequeño párrafo, y el resultado en el navegador.

¹<https://developer.mozilla.org/es/docs/Web/HTML>

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Ejemplo Documento HTML</title>
</head>
<body>

  <h1> Esto es un ejemplo de título</h1>

  <div>
    <!-- Ejemplo de párrafo-->
    <p> Al día siguiente volvió el principito. -Hubiese sido mejor venir a la misma hora -dijo el zorro-. i vienes, por ejemplo, a las cuatro de la tarde, comenzaré a ser feliz desde las tres. Cuanto más avance la hora, más feliz me sentiré. A las cuatro me sentiré agitado e inquieto; ¡descubriré el precio de la felicidad! Pero si vienes a cualquier hora, nunca sabré a qué hora preparar mi corazón... Los ritos son necesarios.</p>
  </div>

</body>
</html>

```

Figura 2.1: Ejemplo Documento HTML



Figura 2.2: Ejemplo Documento HTML en navegador

Este ejemplo simple está escrito en HTML5, que es la última versión de HTML, y que empieza con la siguiente línea: <!DOCTYPE html>. Esta versión de HTML introdujo algunas novedades como lenguaje respecto a su ya obsoleta predecesora HTML4, como el poder dar cobertura a la reproducción de contenido multimedia y la posibilidad de añadir archivos multimedia a la web, nuevas etiquetas para crear animaciones en 2D e introduce la etiqueta de <canvas>, que se usa para dibujar escenas gráficas usando JavaScript, almacenamiento local en el lado del cliente, además de otras muchas mejoras y avances. En HTML5 se concibe la página ya no como un sitio estático, sino como algo dinámico e interactivo.

Como estándar, HTML5 se estableció definitivamente en 2016 y es el estándar con el que están programadas todas las páginas web. Su especificación define también cómo sus etiquetas interactúan con JavaScript, a través del Modelo de Objetos de Documentos (DOM). Este

estándar introduce varias APIs como por ejemplo la API Canvas, que nos proporciona un área en el documento para poder dibujar cualquier cosa; Web Workers, para ejecutar scripts en segundo plano (background), sin ser interrumpidos por eventos producidos por interacciones del usuario con la interfaz; o APIs de navegador para manipular los elementos del DOM y que son imprescindibles para el desarrollo de este proyecto. Algunas de estas APIs se definen en el apartado siguiente, centrado en JavaScript. Como se ha comentado en el Contexto, lo interesante del documento HTML es poder transformar las etiquetas en un árbol de nodos para poder modificar su estructura de forma dinámica a través de JavaScript.

La estructura de etiquetas del HTML mostrado anteriormente sería la siguiente:

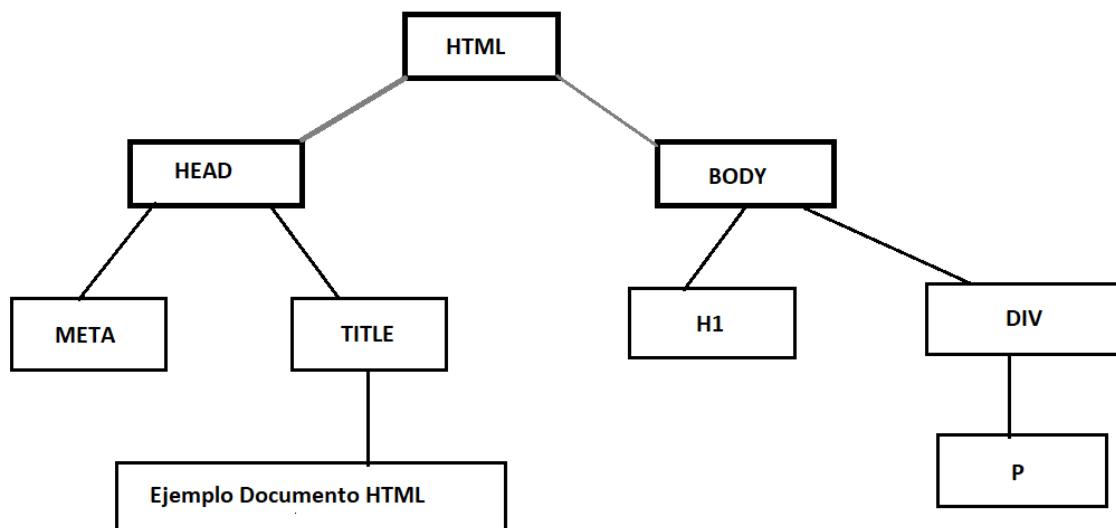


Figura 2.3: Ejemplo Estructura HTML

Es imposible separar del HTML el concepto del Modelo de Objetos de Documentos (DOM), que es la API para documentos HTML que define la estructura lógica de los mismos y el modo en el que se accede y manipula. Con el DOM, podemos construir documentos, navegar por su estructura, y añadir, modificar, o eliminar elementos y contenido.

En el DOM, los documentos tienen una estructura lógica que es muy parecida a un árbol. Cada documento contiene un nodo `doctype` que sirve como raíz para el documento y del que derivan todos los demás. Como se observa en la imagen anterior, el nodo raíz es un nodo de tipo *documento HTML*, y a partir de él existen dos nodos en el mismo nivel formados por las etiquetas *head* y *body*. El uso de lenguajes como JavaScript, que se explica a continuación, nos

va a permitir acceder y manipular cada uno de los elementos que componen el DOM.

2.2. JavaScript

JavaScript² es un lenguaje de programación creado por Brendan Eich, un trabajador de Netscape, en el año 1995. Se trata de un lenguaje que permite añadir interactividad en páginas web pudiendo ejecutarse en el navegador del usuario. La mayoría de navegadores existentes son compatibles con JavaScript. Se trata de un lenguaje interpretado y orientado a objetos. Es también un lenguaje débilmente tipado, lo que significa que las variables no tienen que tener un tipo especificado.

Usando el DOM como modelo de objeto del documento HTML nos va a permitir manipular este mismo documento con JavaScript a través de objetos, como el objeto *document*, que representa al documento mismo. Es decir, el contenido de la página es almacenado como un árbol de nodos a través del DOM y el acceso y manipulación de los mismos se hace a través de JavaScript. Para acceder al DOM simplemente se crea un <script> en el propio HTML o fuera de él donde se escribe el código JS, accediendo así a los elementos *document* o *window*. A continuación se muestra un sencillo ejemplo que se ejecuta cuando el documento se está cargando, además de crear un nuevo elemento H1, ponerle texto y agregar dicho elemento al árbol de nodos.

²<https://developer.mozilla.org/es/docs/Web/JavaScript>

```

<html>
  <head>
    <script>
      // ejecuta esta función cuando se cargue el documento
      window.onload = function() {

        // crea dinámicamente un par de elementos HTML en una página vacía
        var heading = document.createElement("h1");
        var heading_text = document.createTextNode("el texto que desee");
        heading.appendChild(heading_text);
        document.body.appendChild(heading);
      }
    </script>
  </head>
  <body>
    </body>
  </html>

```

Figura 2.4: Acceso DOM a través de JS

Por otro lado, JavaScript tiene, sobre todo del lado del Cliente, muchas APIs disponibles. Estas APIs son interfaces de programación que permiten crear funcionalidades complejas de una manera simple. Cabe destacar las APIs de Navegador, integradas en el propio navegador web, y que nos permite hacer cosas complejas y útiles con los datos del navegador. Entre las APIs de navegador más destacadas están:

- **APIs para manipular documentos** : Un ejemplo es el API DOM para manipular HTML y CSS.
- **APIs que obtienen datos del servidor**: Estas APIs se usan sobre todo para actualizar pequeñas secciones de la página web.
- **APIs para dibujar y manipular graficos**: Están soportadas por la mayoría de navegadores. Las más populares son Canvas y WebGL, y permiten actualizar y manipular la información contenida en un <canvas>HTML, para crear escenas en 2D o 3D.
- **APIs de Audio o Video**: APIs para manipular elementos multimedia como *HTMLMediaElement* o *WebRTC*.

En definitiva, la capacidad de interactuar mediante JavaScript con el navegador o la página web son básicamente APIs, conformadas por objetos con una serie de métodos y propiedades

que son accesibles desde nuestro programa para obtener información de ellos o manipularlos. Por ejemplo, el objeto *Document* implementa el DOM, que no es más que una API para interactuar con todos los elementos del documento HTML.

2.3. A-Frame

A-Frame³ es un framework web de código abierto para construir experiencias en realidad virtual. Está basado en código HTML, lo que hace más fácil su uso. Originalmente fue concebido dentro de Mozilla y ahora es mantenido por los co-creadores de A-Frame dentro de Supermedium, un navegador específico para realidades virtuales.

A-Frame apoya la mayoría de sistemas de realidad virtual como son Vive, Rift, Windows Mixed Reality, Daydream, GearVR, Cardboard, Oculus Go, .. e incluso puede ser utilizado para el uso de realidad aumentada.

A continuación se muestra un documento HTML donde se inserta una escena básica con A-Frame. Para construir una escena basta con añadir en la etiqueta `<script>` la versión instalable de A-Frame disponible desde su propia página web, y a partir de ahí insertar dentro de la etiqueta `<a-scene>` los elementos de la escena que se quiere introducir. En este caso, una pequeña caja, un cilindro, una esfera, un plano y un color de fondo.

³<https://aframe.io/>

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Ejemplo Documento HTML y A-Frame</title>

    <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
  </head>
  <body>
    <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>
  </body>
</html>
```

Figura 2.5: Ejemplo Documento HTML con A-Frame

El resultado en el navegador es el siguiente:

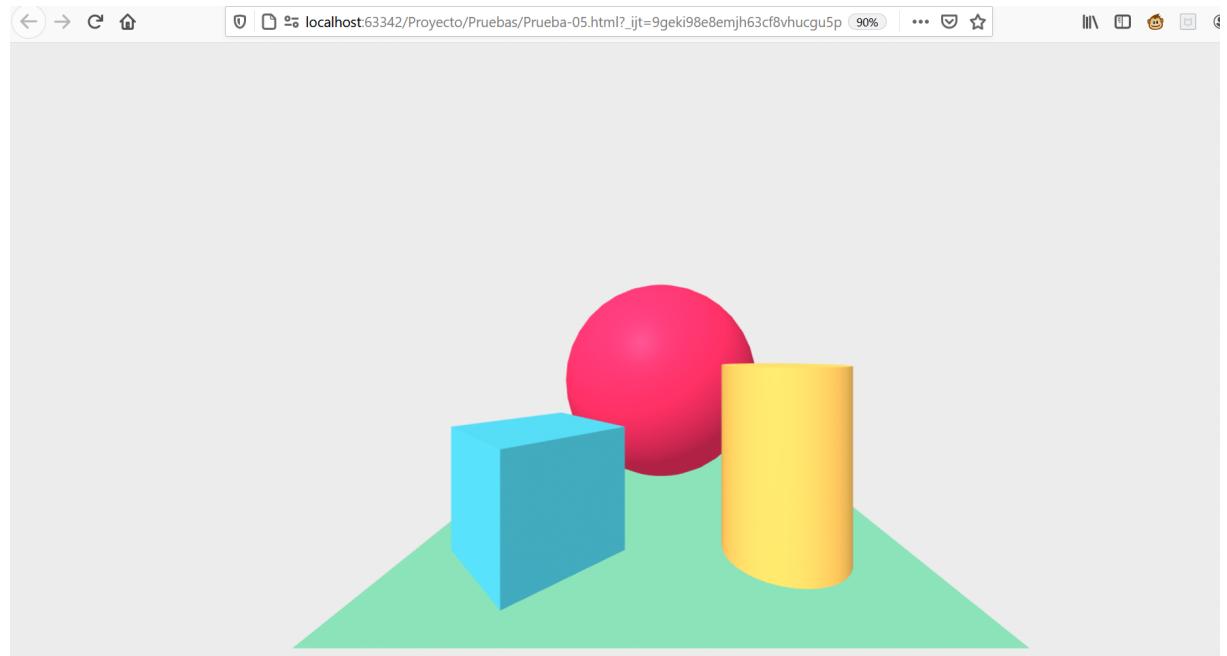


Figura 2.6: Ejemplo Escena A-Frame

A-Frame es soportado por la mayoría de navegadores Web existentes en la actualidad, y funciona igualmente en otros dispositivos como son tablets o teléfonos móviles con orientación de movimiento para crear una total sensación inmersiva de realidad virtual.

Un ejemplo de la misma escena de antes vista a través del teléfono móvil se muestra a continuación. Con el uso de dispositivos de realidad virtual nos permite crear la sensación de estar dentro de la escena a través de los datos de posición, orientación y movimiento.

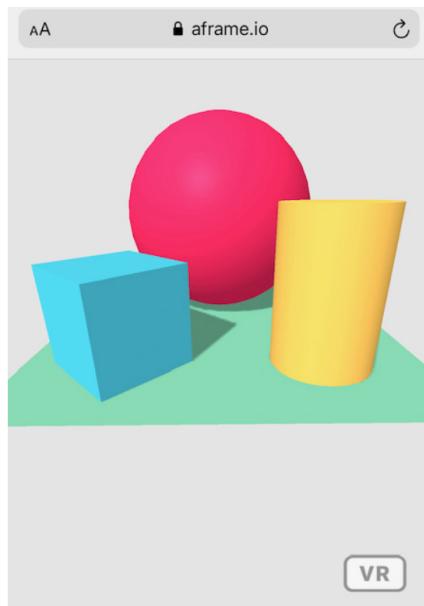


Figura 2.7: Ejemplo Escena móvil A-Frame

Además de escenas con entidades geométricas como las mostradas, A-Frame nos permite realizar múltiples cosas como modelos en 3D propios o algunos de los gratuitos que existen ya desarrollados en código abierto, escenas interactivas basadas en videojuegos, animación en realidad virtual, etc. Dispone de su propio inspector, para ver y crear gráficamente escenas en A-Frame, similar a un inspector de páginas web. Este inspector nos permite tener distintas perspectivas de la escena, acceder y modificar alguna de las propiedades de los componentes de las distintas entidades, mover los objetos por la escena, etc. La apariencia del inspector de una escena básica A-Frame es la siguiente:

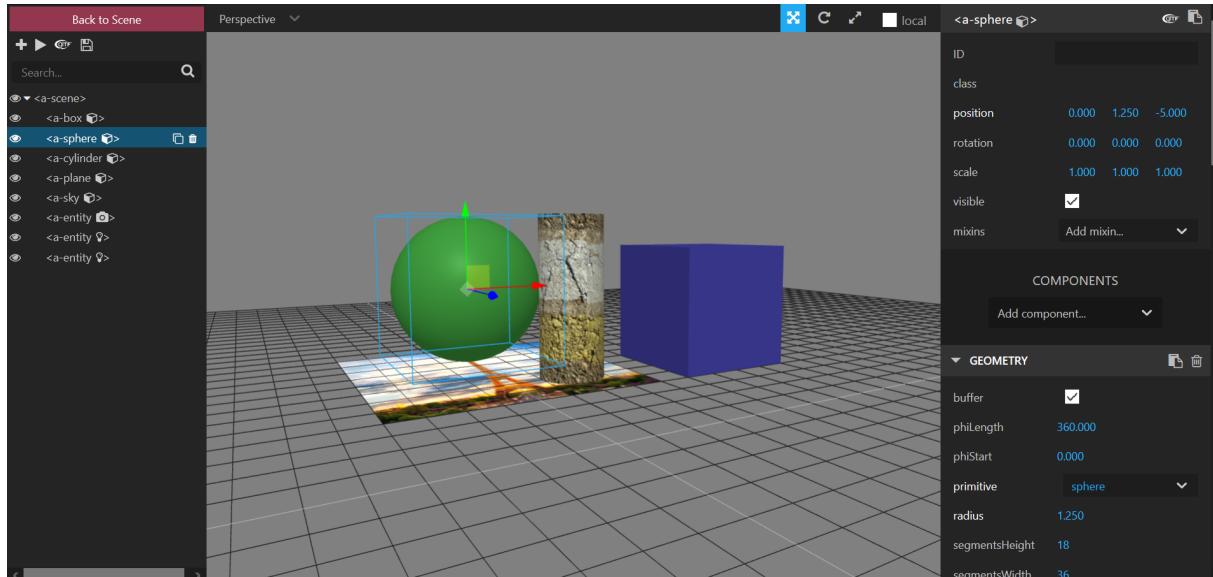


Figura 2.8: Ejemplo Inspector escena A-Frame

A-Frame es un framework construido sobre Three.js que sigue un patrón de arquitectura basado en el sistema de componentes y entidades (ECS), común en el desarrollo e implementación de videojuegos y en 3D, y sigue el principio de composición sobre herencia y jerarquía. En 2D, los elementos tienen un comportamiento fijo en una jerarquía. En 3D y VR es diferente, ya que hay infinitos tipos de objetos con un comportamiento ilimitado. ECS proporciona un patrón manejable para construir tipos de objetos. Los conceptos que involucran al modelo ECS son:

- Entidades: Son contenedores de objetos para poder insertar y tratar los componentes.
- Componentes: Son módulos reutilizables usados por las entidades para proporcionar comportamiento, apariencia o funcionalidad a las mismas.

En la siguiente imagen se muestra algunos ejemplos de diferentes tipos de entidades construidas a partir de la composición de diferentes componentes.

- Box = Position + Geometry + Material
- Light Bulb = Position + Light + Geometry + Material + Shadow
- Sign = Position + Geometry + Material + Text
- VR Controller = Position + Rotation + Input + Model + Grab + Gestures
- Ball = Position + Velocity + Physics + Geometry + Material
- Player = Position + Camera + Input + Avatar + Identity

Figura 2.9: Ejemplo Entidades A-Frame

En definitiva, este framework nos va a permitir construir la escena en realidad virtual y las entidades correspondientes a cada elemento del DOM.

2.4. WebXR

WebXR⁴ es una API que permite crear aplicaciones en realidad virtual y realidad aumentada para el navegador web, creando compatibilidad con dispositivos de realidad virtual y realidad aumentada para visualizar la escena, proporcionando acceso a las capacidades de entrada (información de auriculares y controladores) y salida comúnmente asociadas con los dispositivos de VR y AR. El módulo de realidad aumentada de WebXR permite que el contenido virtual se alinee con el entorno del mundo real antes de mostrarse a los usuarios. El estándar WebXR solo es compatible con páginas en HTML5 y está basado en JavaScript. Esta API es imprescindible a la hora de generar escenas con A-Frame y su posterior visualización en 3D a través de los dispositivos de VR.

A continuación se muestra un ejemplo de las posibilidades que nos ofrece esta API, usando la realidad aumentada para mostrar en un dispositivo móvil la distancia social recomendada de 2 metros.

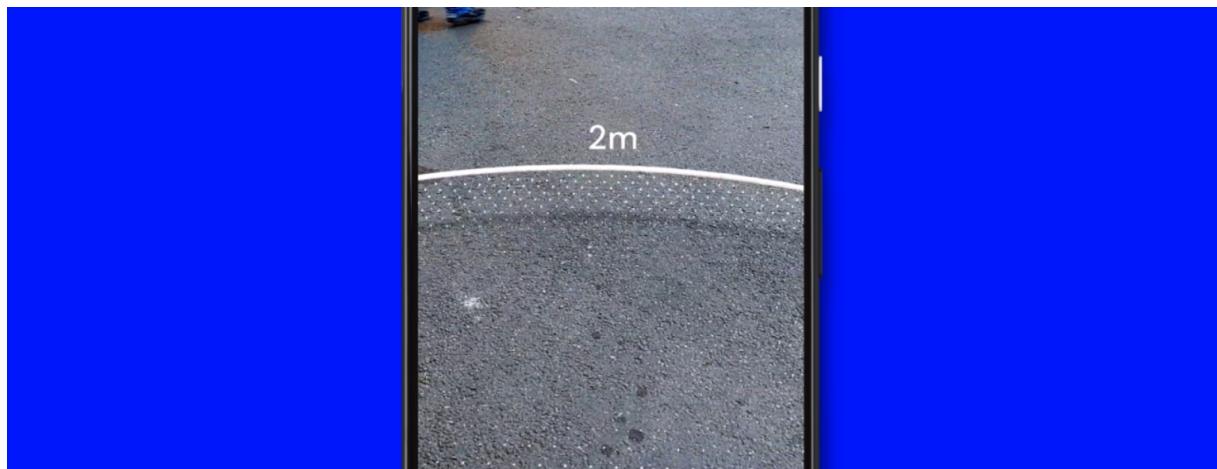


Figura 2.10: Ejemplo WebXR

⁴<https://immersive-web.github.io/webxr-samples/>

2.5. Otras Tecnologías

2.5.1. Three.js

Three.js⁵ es una librería que permite generar y animar gráficos en 3D dentro del navegador. A-Frame está basado en esta librería, de modo que proporciona acceso completo a la API. Se trata de una librería muy popular, relativamente nueva (nacida en 2010), y fácil de usar a través de JavaScript.

La librería Three.js se usa a menudo para generar escenas 3D con WebGL (Web Graphics Library), API implementada en JavaScript para la renderización de elementos en 3D. Esta API es más a bajo nivel para dibujar elementos como líneas, puntos o triángulos. Para más cosas útiles es necesario escribir código, y es ahí donde entra Three.js.

A continuación se muestra un pequeño diagrama de una pequeña app Three.js para hacernos una idea de cómo se conectan los objetos entre sí.

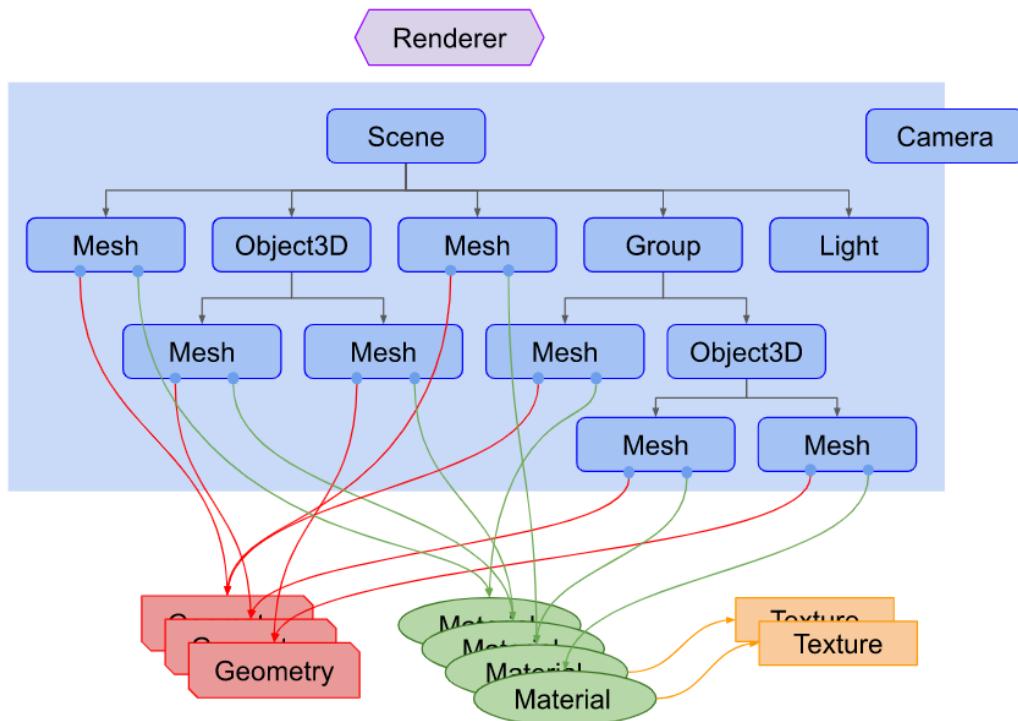


Figura 2.11: Diagrama Three.js

⁵<https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene>

En el diagrama se observa que el objeto principal es el *Renderer*, donde se le pasa una escena y representa la parte de la escena 3D que está dentro de la cámara como una imagen 2D del canvas. Como hemos comentado, el renderizador más completo es WebGL.

2.5.2. CSS

CSS⁶ son las siglas de Cascading Style Sheets, y se trata de un lenguaje que permite establecer la apariencia de un documento escrito mediante un lenguaje de marcas. CSS permite asociar reglas a los elementos que aparecen en una página web, las reglas indican como debe representarse el contenido de esos elementos. Algún ejemplo de lo que podemos hacer con CSS es el siguiente:

```
body {
    background-color: #dadce0;
}

#elem1 {
    font: 150% sans-serif;
    color: mediumblue;
    border-right: green 10px solid;
    border-left: red 5px solid;
    margin-left: 260px;
    padding-right: 0;
}
p {
    font-family: serif;
    font-size: 16pt;
}
img {
    margin-left: 260px;
}
```

Figura 2.12: Ejemplo CSS

En la imagen anterior se muestra un ejemplo de un documento CSS, donde se modifican algunas de las propiedades de diferentes elementos del HTML. En este caso, se cambia el color de fondo del elemento *Body*, se modifica el tipo de letra o el color del elemento *h1* o se añade un margen izquierdo en la imagen. El resultado de estas modificaciones es este:

⁶<https://developer.mozilla.org/es/docs/Web/CSS>



Figura 2.13: Ejemplo CSS

CSS ha ido evolucionando a lo largo de los años hasta la versión actual, CSS3, que carga más rápido que sus predecesoras, incluye múltiples nuevas propiedades y funciona con módulos. El soporte de CSS3 por parte de los motores de renderizado que usan los navegadores web no es igual en todos ellos, lo que ha hecho que ciertos navegadores hayan tardado más en añadir características de CSS3.

2.5.3. html2canvas

html2canvas⁷ se trata de una librería basada en JavaScript que permite convertir un elemento HTML en imagen. Consiste en leer el DOM y construir una representación del mismo en un elemento canvas existente. Es compatible con la mayoría de navegadores actuales, y en este proyecto la idea es utilizar esta librería para renderizar cada elemento del documento HTML como imagen. Para usarlo, basta con llamar a la librería pasándole el elemento que se desea renderizar, y devuelve un *Promise* conteniendo el elemento canvas.

```
html2canvas(document.body).then(function(canvas) {
  document.body.appendChild(canvas);
});
```

Figura 2.14: Ejemplo uso html2canvas

⁷<http://html2canvas.hertzen.com/documentation/>

A continuación se muestra un ejemplo de su funcionamiento, con un documento HTML simple que consta de un único elemento y una entidad Box construida en A-Frame. Mediante el uso de la librería html2canvas, representamos el elemento *h1* como imagen en un canvas que lo añadimos como textura en todas las caras de la caja. El resultado es el siguiente:



Figura 2.15: Ejemplo html2canvas

2.5.4. LateX

LateX⁸ es un sistema de composición de textos de alta calidad que proporciona una serie de órdenes para describir la estructura del documento, con el fin de que el usuario se preocupe más del contenido del documento que de su presentación. De esta forma, el usuario no tiene por qué preocuparse de hacer saltos de página, justificaciones, sangrías, índices del documento, etc. Cuando creamos un documento mediante LateX y con alguno de los muchos editores de texto disponibles para trabajar con LateX, se configuran una serie de ficheros, como son:

- Fichero .tex: Fichero de texto que contiene tanto el texto como las instrucciones para formatear ese texto. Se puede crear con cualquier editor de textos.
- Fichero .aux: Fichero auxiliar que contiene la información sobre las referencias, bibliografía, ...

⁸<https://www.latex-project.org/help/documentation/>

- Fichero .pdf: Resultado de la compilación.
- Fichero .log: Con los mensajes del compilador.
- Ficheros .toc, .lof, .lot: Con información relativa a índices, lista de figuras y lista de tablas.

Este proyecto ha sido redactado mediante este sistema de composición de textos.

2.5.5. GitHub

GitHub⁹ es un sitio web que permite a los desarrolladores almacenar y administrar su código, con el fin de llevar un control ante cualquier cambio sobre el mismo. Se basa en el sistema de control de versiones Git, que facilita un mayor control del proyecto a través de la bifurcación y la fusión.

Con la bifurcación, el usuario duplica parte del código fuente (llamado repositorio). Este desarrollador, luego puede, de forma segura, hacer cambios a esa parte del código, sin afectar al resto del proyecto. Luego, una vez que el usuario logre que su parte del código funcione de forma apropiada, esta persona podría fusionar este código al código fuente principal para hacerlo oficial.

Desde el principio, a partir del Sprint 0, se fue almacenando cada avance en el repositorio correspondiente al proyecto en mi página personal de GitHub, para poder tener un mayor control de cada paso efectuado en el proceso.

2.6. Bootstrap

Bootstrap¹⁰ es un framework font-end que permite la construcción de páginas web adaptativas. Se trata de un software de código abierto creado en el año 2011 por Mark Otto y Jacob Thornton, empleados de Twitter.

Bootstrap ha evolucionado desde ser un proyecto enteramente basado en CSS hasta incluir múltiples plugins Javascript e iconos junto con formularios y botones. Presenta una cuadrícula de doce columnas y 940px de ancho[6].

⁹<https://github.com/>

¹⁰<https://getbootstrap.com>

La creación de páginas web mediante Bootstrap se simplifica al disponer de elementos predefinidos fácilmente incluibles en cualquier página web, tales como menús desplegables, botones, iconos, ventanas emergentes, etc.

La página web del proyecto, así como alguna de las demos para el resultado final, han sido hechas mediante Bootstrap.

Capítulo 3

Desarrollo del Proyecto

A continuación se detalla el proceso llevado a cabo para la visualización del árbol de nodos de un documento HTML como entidades A-Frame.

3.1. Introducción

El proyecto se concibió desde el principio para ser implementado inspirándonos en una metodología ágil SCRUM, sin llegar a los niveles de complejidad que puede llegar esta metodología, basada en entregas parciales y regulares de diferentes tareas (denominadas sprint), aumentando la complejidad de cada una de ellas cada vez, en intervalos de tiempo y de duración variable, ya que si bien en principio cada sprint tenía un tiempo fijo establecido de unas 3 semanas aproximadamente, este podía ser ampliado en función de las dificultades encontradas en el desarrollo del sprint o simplemente por motivos ajenos al proyecto. Siguiendo con la analogía con la metodología SCRUM, los roles existentes en este proyecto serían el de SCRUM Master, representado en este caso por el director del proyecto, y el equipo de desarrollo estaría formado únicamente por mí. Esta metodología permitió llevar a cabo el proyecto en diferentes partes escalables más sencillas y monitorizar de una manera más sencilla el proyecto con el director del proyecto.

En un principio, se estableció como tarea inicial o Sprint 0 una introducción para llevar a cabo pequeñas pruebas y tutoriales con tecnología que eran ajena a mí en ese momento, como A-Frame. A partir de entonces, en cada reunión mostraba los avances y resultado obtenido en ese sprint, así como posibles problemas que hubieran surgido en el desarrollo del mismo, y

se establecían los requisitos y objetivos a alcanzar para el siguiente sprint.

Las tareas o Sprints implementadas y desarrolladas en este proyecto se detallan a continuación.

3.2. Desarrollo e implementación

A continuación se detalla el desarrollo y la implementación del proyecto partir de cada sprint realizado, detallando las tareas ejecutadas en cada una de ellas.

3.2.1. Sprint 0

Se trata del Sprint inicial, donde tras establecer con Jesús el planteamiento general del proyecto a realizar, me indica esta primera tarea de documentación, pequeñas pruebas y tutoriales con el framework A-Frame, para así acercarme poco a poco a la tecnología en 3D en general y a este framework en particular, el cuál desconocía en ese momento.

Este Sprint tiene una duración aproximada de unas cuatro semanas, y los objetivos principales durante este tiempo son la lectura de parte de la documentación de A-Frame contenida en su propia página web y llevar a cabo sencillas pruebas y tutoriales a partir de esta documentación, además de poder mostrar unas pequeñas escenas con algunas de las principales entidades del framework. Las escenas y resultados obtenidos durante este sprint se muestran a continuación:

- En esta primera escena construida con A-Frame ya se observan algunas de las principales entidades contenidas en dicho framework, que no son más que objetos a los que podemos añadirles apariencia, comportamiento y funcionalidad. Esta escena en concreto contiene elementos como un cilindro, una caja, un plano y una esfera. Tras definir las entidades, podemos manipular componentes de cada uno como la posición, rotación o escala. Además, se incluye en alguna de ellas apariencia añadiendo una imagen como material de la misma. El código básico para construir la escena es el siguiente:

```
<a-scene>
  <a-box
    position="3.5 1 -3"
    rotation= "0 90 0"
    scale= "2 2 2" color="#2F2E77">
  </a-box>

  <a-sphere position="0 1.25 -5" radius="1.25" color ="#2F772E"></a-sphere>
  <a-cylinder position="1 1.25 -3" radius="0.5" height="2.5" src="lib/bands2.jpg"></a-cylinder>
  <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4" src="lib/paris.jpg"></a-plane>
  <a-sky color="grey"></a-sky>
```

Figura 3.1: Código escena básica

El resultado de la escena se muestra a continuación:

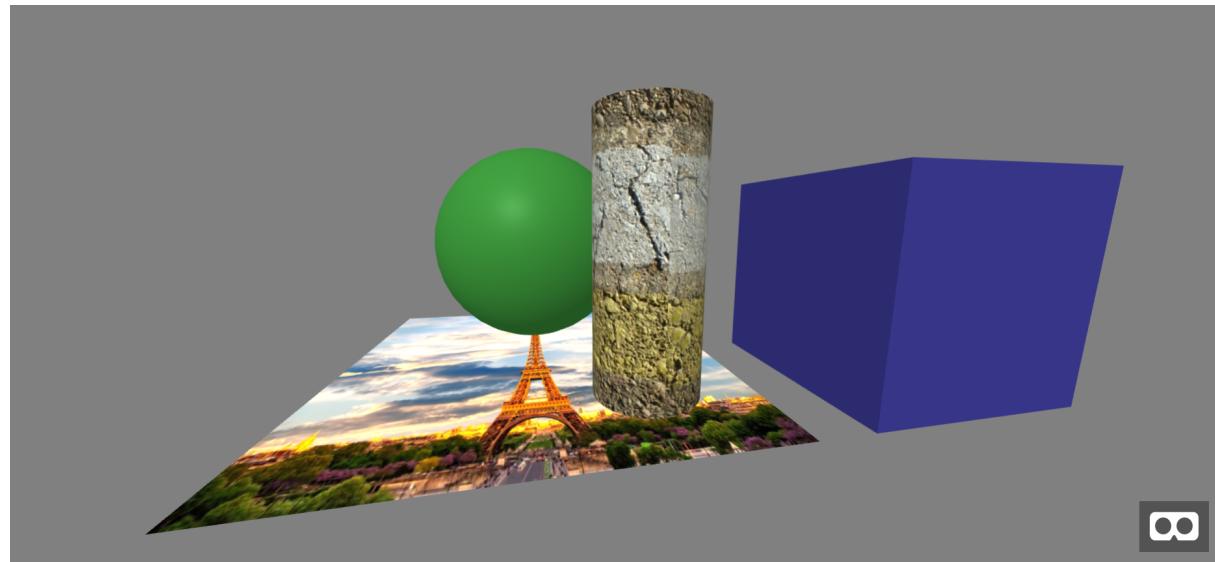


Figura 3.2: Escena básica Sprint 0

- A partir de la escena básica anterior, se añade alguna funcionalidad extra como un pequeño texto o un pequeño cursor que nos permite hacer click en él y conseguir una interactividad básica con la escena en dispositivos que no disponen de controladores de mano. La apariencia básica de este cursor es un pequeño anillo.

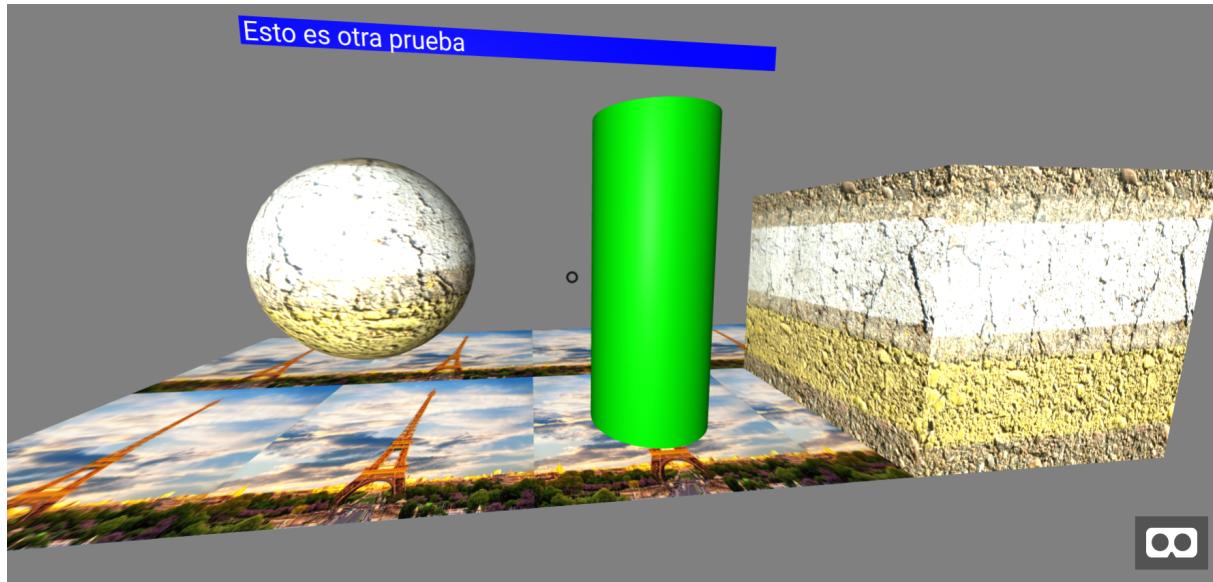


Figura 3.3: Escena básica 2 Sprint 0

- También, podemos completar un poco más la escena añadiendo unos planos curvilíneos y un panel de estadísticas donde se muestra una interfaz de usuario con métricas relacionadas con el rendimiento. Este componente de estadísticas aplica solo al elemento <a-scene>. En este panel podemos observar métricas como los fps (frames por segundo), el número de entidades o vértices en la escena, la latencia, el tiempo que tarda en cargarse la escena completa o el número de geometrías Three.js en la escena. Un ejemplo de la escena con este panel de métricas se muestra a continuación:

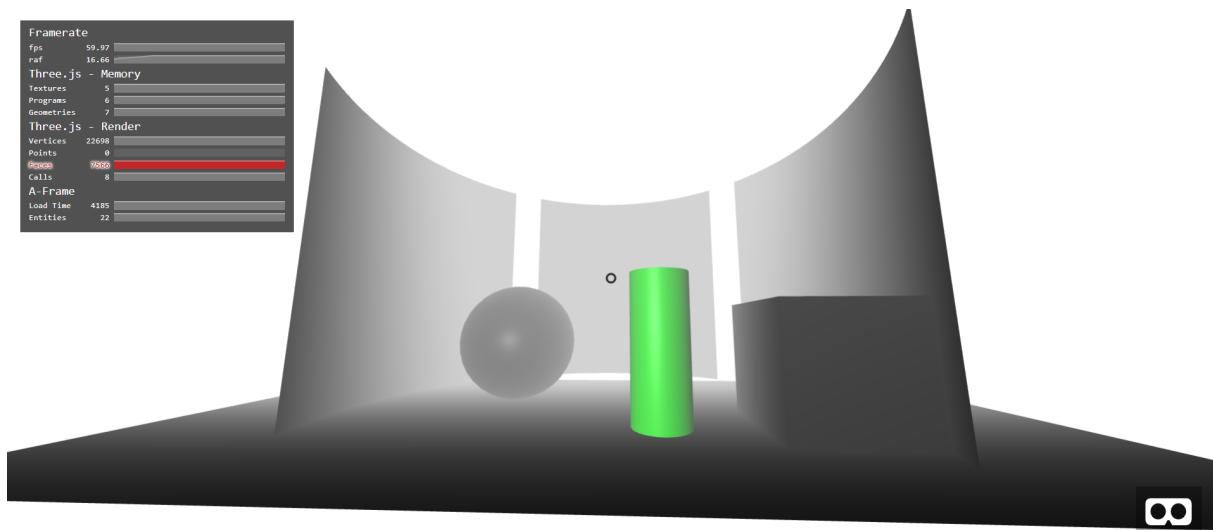


Figura 3.4: Escena básica 3 Sprint 0

3.2.2. Sprint 1

Durante este sprint el objetivo fue seguir llevando a cabo pruebas con entidades A-Frame mediante gráficas de barras en este caso y jugar con las posibilidades que nos ofrece este framework en la visualización de los datos que obtenemos de dichas gráficas. Además, empiezo en esta parte los primeros puntos clave del proyecto, y uno de los requisitos principales durante este sprint es poder recorrer recursivamente los elementos pertenecientes al Body de un HTML y sus respectivos hijos, para poder mostrarlo por consola posteriormente.

- Al principio de esta etapa el objetivo fue realizar pequeñas gráficas de barras mediante componentes A-Frame e introduciendo conceptos como la entidad camera, cargar los datos a través de documentos JSON, etc. En A-Frame, los componentes son módulos en JavaScript para construir la funcionalidad y apariencia de una escena. Para este ejemplo, defino un componente básico denominado histogram3 llamado desde la propia escena como atributo HTML. Una vez registrado e inicializado el componente, defino los métodos o propiedades que lo componen. Para este caso, los métodos *Schema*, que describe las propiedades del componente, y *init*, que es llamado una vez que comienza el ciclo de vida del componente. La apariencia del componente se muestra a continuación:

```

init: function() {
    var datos = [3,5,7,9,11];
    var data = this.data;
    var entityEl = this.el;

    var relativeX, relativeY, relativeZ;

    relativeX = data.position[0];
    relativeZ = data.position[2];

    console.log("Posicion X: " + relativeX + ", posicion Y: " + relativeY + " --- position Z: " + relativeZ);

    var alturaInicial = data.position[1];

    for(var i=0; i < datos.length; i++) {
        var height = datos[i];

        relativeY = alturaInicial + (height / 2);

        var elBox = document.createElement('a-box');
        elBox.setAttribute('width', data.width);
        elBox.setAttribute('height', height);
        elBox.setAttribute('depth', data.depth);
        elBox.setAttribute('color', data.color);

        elBox.setAttribute('position', {
            x: relativeX,
            y: relativeY,
            z: relativeZ
        });

        elBox.addEventListener('click', function () {
            this.setAttribute('color', 'red');
        });

        entityEl.appendChild(elBox);

        relativeX += 1;
    }
}

```

(a) Componente Schema

(b) Componente Init

Figura 3.5: Componentes A-Frame

El resultado de estos componentes es un gráfico de barras en que se controla las acciones del ratón, de modo que cuando el cursor se coloca en alguna de las barras se cambia el color de ésta:

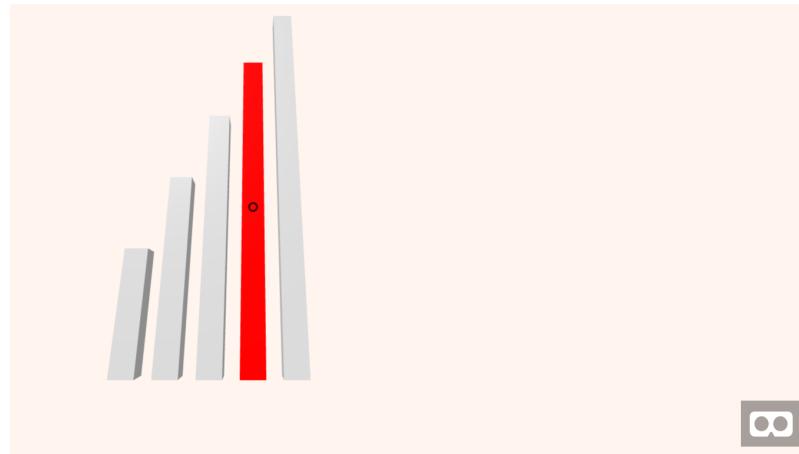


Figura 3.6: Escena Histograma

A partir de esto, añado alguna funcionalidad más como la entidad *cursor* dentro de la entidad <a-camera> para controlar las acciones del ratón pero sin necesidad de mostrar el cursor como pasaba en el ejemplo anterior. Además, se incluyen unos ejes XYZ a la gráfica y se incorpora un cuadro de diálogo en la ventana superior que informa de la altura de la barra en la que está situado el cursor del ratón. La definición de la entidad *cursor* así como el resultado se muestra a continuación:

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

Figura 3.7: Entidad Cursor

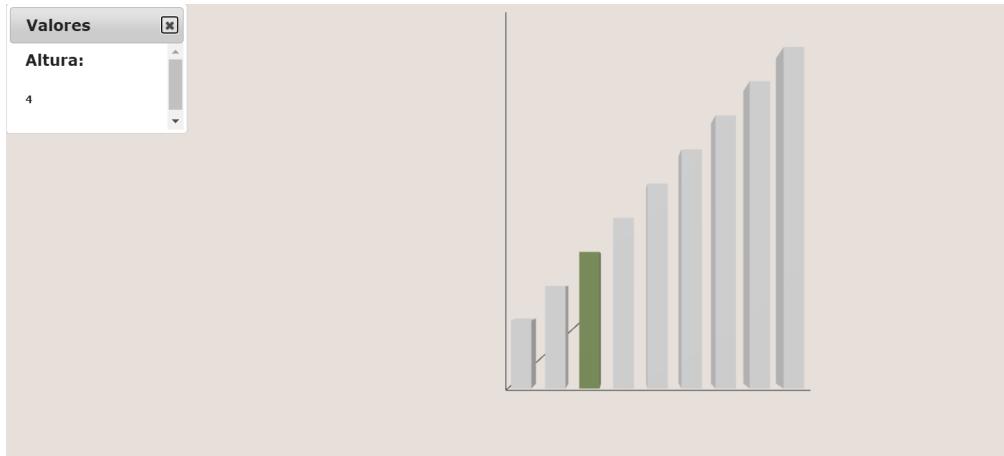


Figura 3.8: Resultado Gráfica

- Tras esto, el objetivo es centrarse en uno de los aspectos fundamentales del proyecto, como es el poder recorrer recursivamente los elementos que componen el *Body* del documento HTML, así como sus respectivos hijos. Para ello, creo un documento HTML sencillo con unos pocos elementos, como este:

```
<body>
    <h1> Prueba con los Elementos del DOM</h1>
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Duis auctor dui vel ante imperdiet hendrerit. In sollicitudin ante ultrices,
        finibus elit sit amet, accumsan enim. Integer imperdiet tempor varius. Aliquam
        hendrerit ultrices ligula, eleifend bibendum ligula suscipit tempor. Suspendisse
        aliquam dictum tristique. Curabitur luctus nisi sit amet dolor varius, sed egestas
        purus tristique. Integer aliquam non lacus at pellentesque. Etiam vel nibh elementum
        odio scelerisque ullamcorper. Nam fermentum lorem sed auctor pretium. Pellentesque
        id scelerisque nisi.
    </p>
    <ol>
        <li> Primer elemento de la lista</li>
        <li> Segundo elemento de la lista</li>
        <li> Tercer elemento de la lista
            <ul>
                <li> Elemento 1 de la segunda lista</li>
            </ul>
        </li>
    </ol>
</body>
```

Figura 3.9: Ejemplo HTML

El objetivo es mostrar en consola todos los elementos del *Body*, así como sus nodos hijos, y éstos a su vez sus nodos hijos si los tuvieran, y así sucesivamente. Para ello, creo una función recursiva donde le pasamos por parámetro el propio elemento Body y el nivel 0, de forma que en dicha función incrementamos el nivel para llamar recursivamente a la misma pasándole el elemento del DOM en el que se encuentre y el nivel del profundidad. La apariencia en consola de esta función es la siguiente:

```

-----  

    ▼ Object { nodeName: "BODY", node: body ⚡ , nivel: 1, hijos: (3) [...] }  

      ▶ hijos: Array(3) [ ..., ..., ... ]  

      ▶ nivel: 1  

      ▶ node: <body> ⚡  

      ▶ nodeName: "BODY"  

      ▶ <prototype>: Object { ... }
-----
```

Figura 3.10: Ejemplo recursividad DOM

De forma que por consola nos muestra una colección de elementos con cuatro propiedades: el nombre del Nodo, el propio nodo con todos sus atributos, el nivel de profundidad en el que se encuentra dentro del árbol del DOM y un array de sus nodos hijo si los tuviera (y cada uno estos hijos muestra estas cuatro mismas propiedades).

3.2.3. Sprint 2

En esta etapa el objetivo principal es poder recrear y visualizar una primera escena en realidad virtual, a partir de los avances conseguidos en la etapa anterior a la hora de recorrer los elementos del DOM que componen el HTML.

Para ello, construyo un HTML con algunos elementos sin hijos en el que incluyo la escena donde se llama al componente *comp-html*, que es donde se va a incluir la función recursiva para recorrerse el DOM y la representación de cada elemento como entidad A-Frame.

```

<body>
  <h1> Prueba con los Elementos del DOM</h1>

  <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.  

    Duis auctor dui vel ante imperdiet hendrerit. In sollicitudin ante ultrices,  

    finibus elit sit amet, accumsan enim. Integer imperdiet tempor varius. Aliquam  

    hendrerit ultrices ligula, eleifend bibendum ligula suscipit tempor. Suspendisse  

    aliquam dictum tristique. Curabitur luctus nisi sit amet dolor varius, sed egestas  

    purus tristique. Integer aliquam non lacus at pellentesque. Etiam vel nibh elementum  

    odio scelerisque ullamcorper. Nam fermentum lorem sed auctor pretium. Pellentesque  

    id scelerisque nisi.

  </p>

  <p id="pruebaP">  

    El quinto planeta era muy curioso. Era el más pequeño de todos, pues apenas cabían en él un farol  

    y el farolero que lo habitaba. El principio no lograba explicarse para qué servirían allí, en el cielo,  

    en un planeta sin casas y sin población un farol y un farolero.
  </p>

  <h3> Ejemplo de Encabezado h3 </h3>

  <div id="myEmbeddedScene">
    <a-scene id="escena" embedded>
      <a-entity camera look-controls position="0 3.6 33"></a-entity>
      <a-entity id="comp-html" comp-html=""/>
      <!-<a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere> -->
    </a-scene>
  </div>
```

Figura 3.11: Ejemplo HTML Sprint 2

La idea es representar el *Body* como una entidad plana rectangular sobre la que se van a situar el resto de elementos. En el DOM, los elementos tienen una serie de dimensiones

como altura o anchura a los que se puede acceder fácilmente mediante los métodos existentes en la API DOM de JavaScript, como son *element.clientHeight* para obtener la altura de un elemento en píxeles, incluyendo el padding pero no las barras horizontales, el borde o el margen, o *element.clientWidth* para sacar la anchura del elemento.

Por otro lado, las entidades A-Frame disponen del componente *position* para colocar el objeto en el espacio 3D a partir de un sistema de coordenadas determinado por tres números (coordenadas X,Y,Z). Se basa en el sistema de coordenadas de la mano derecha, de manera que el eje X positivo se extiende a la derecha (dedo pulgar), el eje Y positivo hacia arriba (dedo índice) y el eje Z positivo hacia ti (dedo corazón).

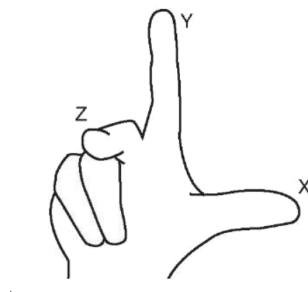


Figura 3.12: Sistema Coordenadas A-Frame

Por lo tanto, conociendo las dimensiones de cada elemento, lo que hago en un primer momento es colocar en la posición A-Frame (0,0,0), que equivale al centro de la escena, la entidad correspondiente al *Body* y hago un cambio en sus dimensiones para que se ajuste a la escena, de tal forma que las dimensiones de la entidad son las siguientes:

- El lado X se corresponde con la altura del elemento del DOM dividido por un factor 10
- El lado Y se corresponde al nivel del elemento del DOM
- El lado Z se corresponde con la anchura del elemento del DOM dividido por un factor 100

Para el resto de elementos del DOM, las dimensiones siguen estos mismos patrones, pero su posición es relativa al elemento padre, en este caso el Body. El resultado final, tanto la escena representada en el navegador como en el inspector de A-Frame, es el siguiente:

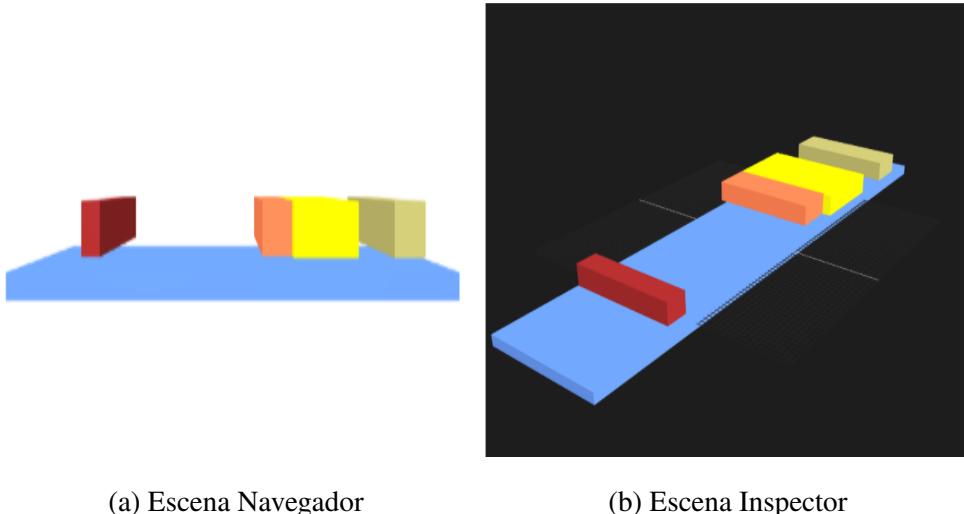


Figura 3.13: Escena A-Frame

Como observamos en la imagen, el elemento Body se corresponde con la entidad base representada en este caso con el color azul, y sobre ella se posan el resto de elementos. La estructura del resto de elementos sigue el orden del documento HTML, de forma que el primer elemento corresponde al primer elemento definido en el DOM (`<h1>`), el segundo elemento al `<p>`, y así sucesivamente. La anchura de cada elemento debe encajar como máximo en las dimensiones de la entidad Body, y su posición representada más arriba o más abajo irá en función de la posición del elemento del DOM con respecto al *top* del documento o el margen superior que tiene con el elemento superior. En el ejemplo anterior, al elemento `<h3>` se le ha añadido un valor de 200 a la propiedad *margin-top*, que indica el margen de un elemento con respecto a elemento anterior. Por ello, en la escena, la entidad de color rojo, que corresponde al elemento `<h3>`, está separada de la entidad anterior.

3.2.4. Sprint 3

Una vez construida la escena A-Frame de un documento HTML sin hijos para poder visualizarlo en 3D, el siguiente objetivo es poder representar y visualizar en realidad virtual un documento HTML más complejo donde sus elementos tengas hijos y éstos a su vez puedan tener algún hijo. Por lo tanto, partimos de un HTML idéntico al utilizado en la etapa anterior, pero añadiendo una lista con hijos, como la que se muestra a continuación:

```
<ol>
    <li> Primer elemento de la lista</li>
    <li> Segundo elemento de la lista</li>
    <li> Tercer elemento de la lista
        <ul>
            <li> Elemento 1 de la segunda lista</li>
        </ul>
    </li>
</ol>
```

Figura 3.14: Ejemplo HTML Sprint 3

Modifico las ecuaciones del cambio de coordenadas, de modo que obtengo la coordenada X de la entidad A-Frame a partir de propiedades del propio elemento como el offsetTop (distancia del elemento actual respecto al borde superior del nodo offsetParent) o el clientHeight (Altura del elemento), además de propiedades del elemento padre como su altura u offsetTop. De igual manera, modiflico la forma de obtener la coordenada Z. Ambas ecuaciones quedarías así:

```
obtenerCoordenadaX: function(elemento, parent) {

    let top = elemento.offsetTop;
    let height = elemento.clientHeight;
    let offTopParent = (parent.offsetTop / 10);

    //Altura del elemento padre / 2
    let parent_height = (parent.clientHeight / 10) / 2;

    return offTopParent + parent_height - (top / 10) - ((height / 10) / 2);
},
```

(a) Coordenada X

```
obtenerCoordenadaZ: function(client_left, client_width, parent) {

    let parent_width = (parent.clientWidth / 100) / 2;

    return (client_left / 100) + ( (client_width / 100) / 2) - parent_width;
},
```

(b) Coordenada Z

Figura 3.15: Ecuaciones Coordenadas A-Frame

El resultado final se muestra a continuación, tanto en el navegador como en el inspector A-Frame:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis auctor dui vel ante imperdier hendrerit. In sollicitudin ante ultrices, finibus elit sit amet, accumsan enim. Integer imperdier tempor varius. Aliquam hendrerit ultrices ligula, eleifend bibendum ligula suscipit tempor. Suspendisse aliquam dictum tristique. Curabitur luctus nisi sit amet dolor varius, sed egestas purus tristique. Integer aliquam non lacus at pellentesque. Etiam vel nibh elementum odio scelerisque ullamcorper. Nam fermentum lorem sed auctor pretium. Pellentesque id scelerisque nisi.

El quinto planeta era muy curioso. Era el más pequeño de todos, pues apenas cabían en él un farol y el farolero que lo habitaba. El principio no lograba explicarse para qué servirían allí, en el cielo, en un planeta sin casas y sin población un farol y un farolero.

1. Primer elemento de la lista
 2. Segundo elemento de la lista
 3. Tercer elemento de la lista
 - o Elemento 1 de la segunda lista
 4. Cuarto elemento de la lista
1. Primer elemento de la Segunda lista

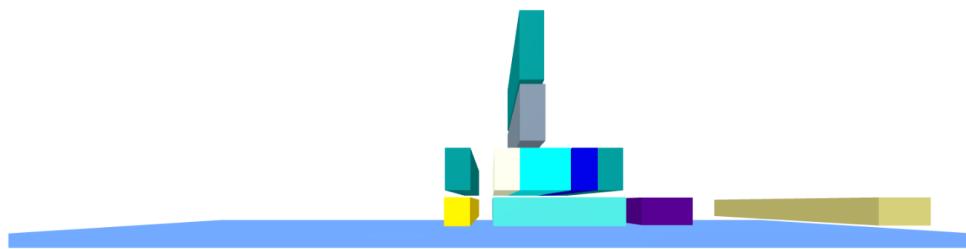
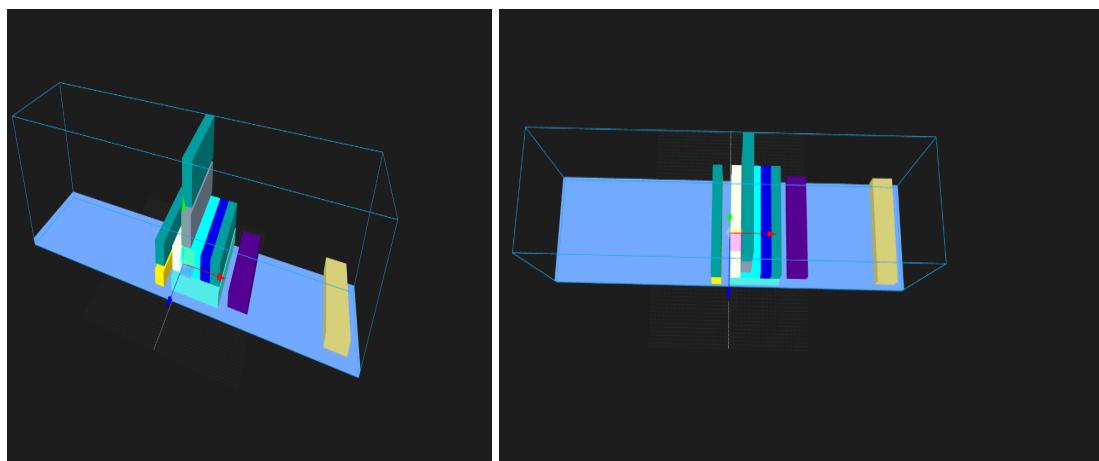


Figura 3.16: Escena A-Frame Sprint 3



(a) Escena Inspector 1

(b) Escena Inspector 2

Figura 3.17: Inspector A-Frame

3.2.5. Sprint 4

Tras representar una escena de un documento HTML con hijos y visualizar en realidad virtual la estructura jerárquica de los nodos que componen el HTML, el siguiente objetivo es llevar a cabo un par de funcionalidades extras al resultado obtenido en el sprint anterior. A continua-

ción, se detalla el proceso seguido para poder incluir ambas funcionalidades al proyecto.

- Cada elemento que encontramos en un documento HTML se halla contenido en una caja rectangular, la cuál cuenta con una serie de propiedades que afectarán en cierta medida a la hora de mostrar el elemento. Entre estas propiedades están algunas de las que ya hemos hablado anteriormente, como *height* y *width*, que definirán la altura y anchura de nuestra caja, y otras como *padding*, *margin* o *border*, para organizar con mayor control el elemento. Para poder entenderlo mejor, se muestra en una imagen estas propiedades:

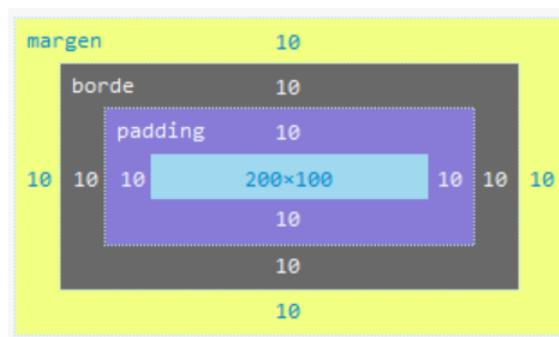


Figura 3.18: Propiedades caja HTML

Por tanto, como se observa en la figura, el ancho real y la altura real de la caja que contiene el elemento HTML ha de tener en cuenta todas estas propiedades:

```
Ancho = margin-left + border-left + padding-left + width + padding-right + border-right + margin-right
Altura = margin-top + border-top + padding-top + height + padding-bottom + border-bottom + margin-bottom
```

Figura 3.19: Ancho y Alto caja HTML

El navegador, por defecto, agrega un margin y padding a los elementos para mejorar su claridad y legibilidad. Estos valores por defecto dependen del navegador utilizado.

Por todo ello, se añade a cada entidad una caja contenedora, de un color semi transparente, que representa las dimensiones totales de la caja que contiene el elemento en el documento HTML, es decir, incluyendo los margin o el resto de características de estilo que se han podido incluir en el elemento.

Por ejemplo, en el documento HTML que usamos para el proyecto , al elemento *h1* se le añade un margin-left de 50px, de modo que tiene una separación en el lado izquierdo de 50px con el elemento que tiene al lado. Visto de una manera gráfica:

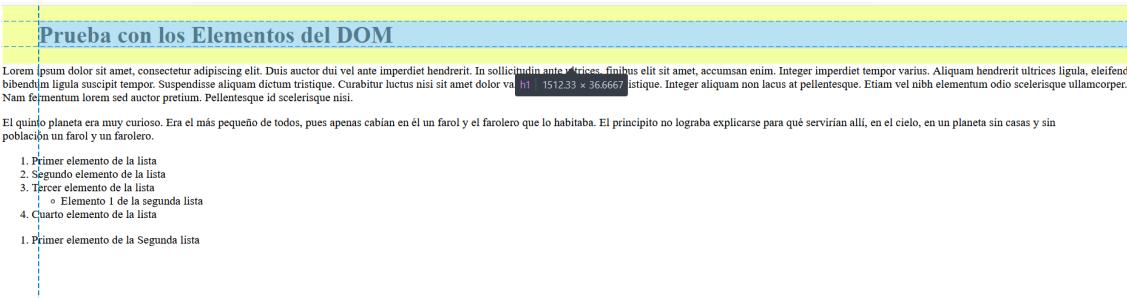


Figura 3.20: Elemento *h1* HTML

En la imagen, se observa con el color azul las dimensiones reales del elemento *h1*, con una pequeña separación en el lado izquierdo respecto al elemento *Body* y el resto de propiedades como su altura o anchura son determinados por el navegador. También se aprecia un pequeño margen tanto en el lado superior como en el inferior con el tope de la caja contenedora que lo contiene. Con el color amarillo se observa las dimensiones totales de la caja que representa el elemento *h1* y cuyas características también son determinados por el propio navegador, de manera que su anchura encaja con la anchura del elemento *Body*.

Para poder observar de forma más clara ambas entidades y sus dimensiones, tanto de la caja que representa al elemento como la caja contenedora del mismo, represento este elemento *h1* y el *Body* en realidad virtual mediante entidades A-Frame, dando como resultado la siguiente escena:



Figura 3.21: Escena elemento h1

La escena, representada en el Inspector A-Frame:

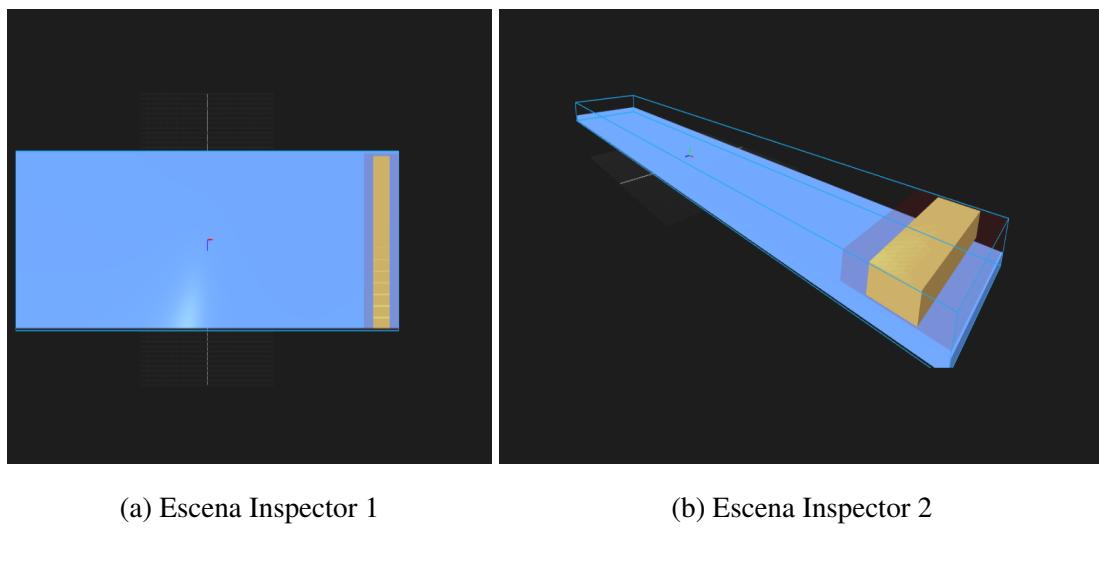


Figura 3.22: Inspector A-Frame

En la imagen podemos ver como la entidad azul representa al elemento *h1* y la entidad semi transparente representa la caja contenedora del mismo. Esta entidad contenedora se visualizará para cada elemento del HTML.

- Mediante el uso de la librería html2canvas, explicada en el apartado correspondiente a las tecnologías utilizadas, el objetivo es renderizar cada elemento contenido en el documento

HTML como imagen para añadirla como textura a un plano colocado en la cara superior de cada entidad.

Para ello, se llevan a cabo una serie de pruebas con unos pocos elementos HTML para comprobar el correcto funcionamiento de esta renderización. Primero con un documento HTML con un par de elementos *h1* y *h2* en el que se incluye una escena A-Frame con dos entidades Box de diferentes tamaños y colocadas en distintas posiciones para cada elemento y dos elementos *canvas* que van a contener la representación en imagen de cada elemento HTML. A la librería html2canvas se le pasa el elemento que se desea renderizar, ya sea *h1* o *h2*, y una serie de opciones en función de lo que se desee configurar. En este caso le añadimos la opción *canvas* para indicar que el elemento *canvas* del HTML lo usaremos como base para representar la imagen. Posteriormente, a la entidad *Plane* creada le añadimos el componente *material*, con el atributo *src* igual a ese *canvas* donde está representada la imagen. De esta forma, al plano se le añade como textura la renderización obtenida del html2canvas. El documento HTML quedaría así:

```
<body>
<div id="elem1">
    <h1>Take screenshot of webpage with html2canvas</h1>
</div>

<h2> Prueba elemento 2 para la entidad segunda</h2>

<a-scene start>
    <a-assets>
        <canvas id="canvas" ></canvas>
        <canvas id="canvas2" ></canvas>
    </a-assets>

    <a-entity id="elemBox1"
        geometry="primitive: box; width: 4; height: 2; depth: 8"
        position="8 0.5 -4" rotation="0 0 0"
        material="color: blue">
    </a-entity>

    <a-entity id="elemBox2"
        geometry="primitive: box; width: 8; height: 2; depth: 8"
        position="-3 0.5 -4" rotation="0 0 0"
        material="color: #2E7736">
    </a-entity>

    <a-camera wasd-controls position="0 5.5 9"></a-camera>
</a-scene>
</body>
```

Figura 3.23: Documento HTML prueba html2canvas

La escena en el navegador se muestra a continuación, donde se observa los planos colocados en la cara superior de cada caja con la representación en imagen de cada elemento HTML correspondiente. Al plano se le ha incluido una rotación para tener la misma orientación que en la página, de manera que se pueda leer desde arriba de la misma forma que en el documento HTML.

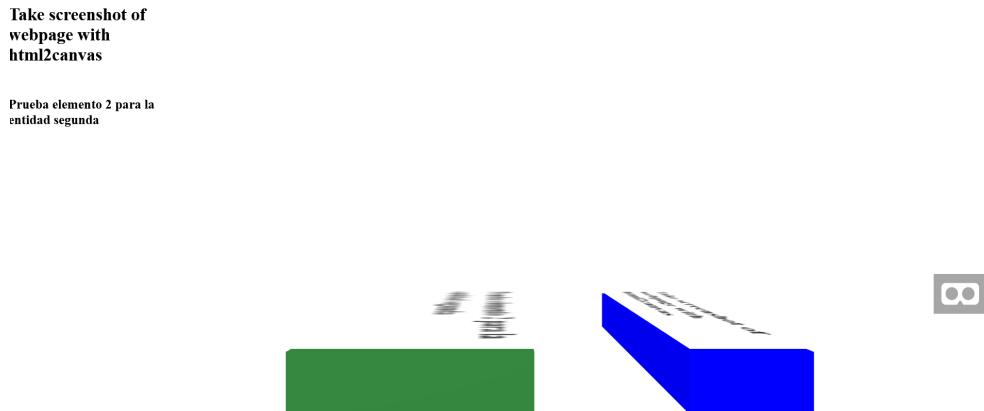
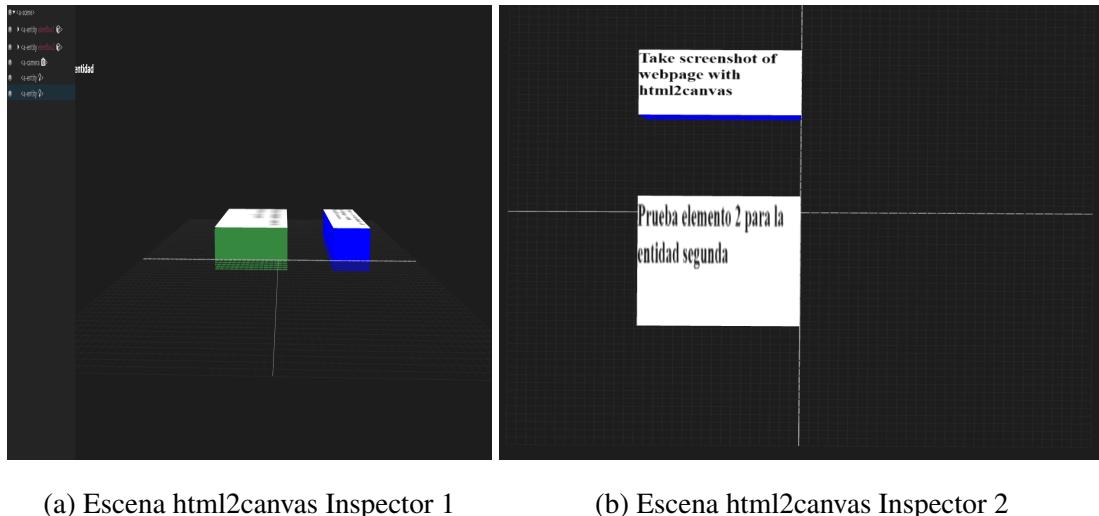


Figura 3.24: Escena A-Frame con html2canvas



(a) Escena html2canvas Inspector 1

(b) Escena html2canvas Inspector 2

Figura 3.25: Inspector A-Frame

La siguiente prueba consiste en hacer una pequeña modificación sobre este HTML utilizado, sustituyendo el elemento *h2* por una imagen. El HTML quedaría así:

```
<div id="elem1">
  <h1>Ejemplo documento HTML con imagen</h1>
</div>
<div>
  
</div>
```

Figura 3.26: Documento HTML prueba 2 html2canvas

El procedimiento es similar al anterior, con dos diferencias: en primer lugar, la llamada a la librería html2canvas para el elemento *img* se hace añadiendo una opción más, *allowTaint: true*, que permite representar en el canvas imágenes externas. Por otro lado, la librería html2canvas accede a la imagen con llamadas a la API de JavaScript, de manera que la imagen y el script *html2canvas.min.js* con la API de la librería deben estar ubicados en el mismo origen. Por ello, colocamos la imagen que se desee renderizar en el mismo directorio del proyecto y el resultado de la escena es el siguiente:

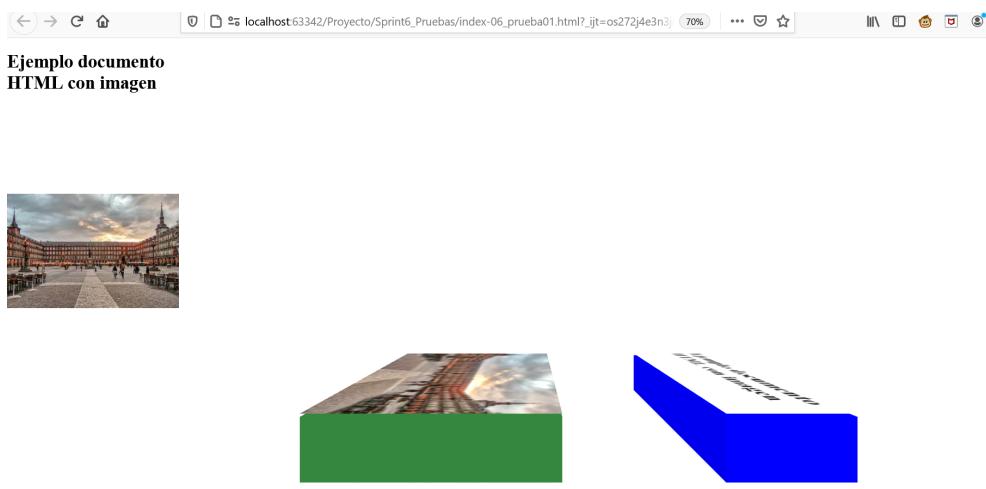
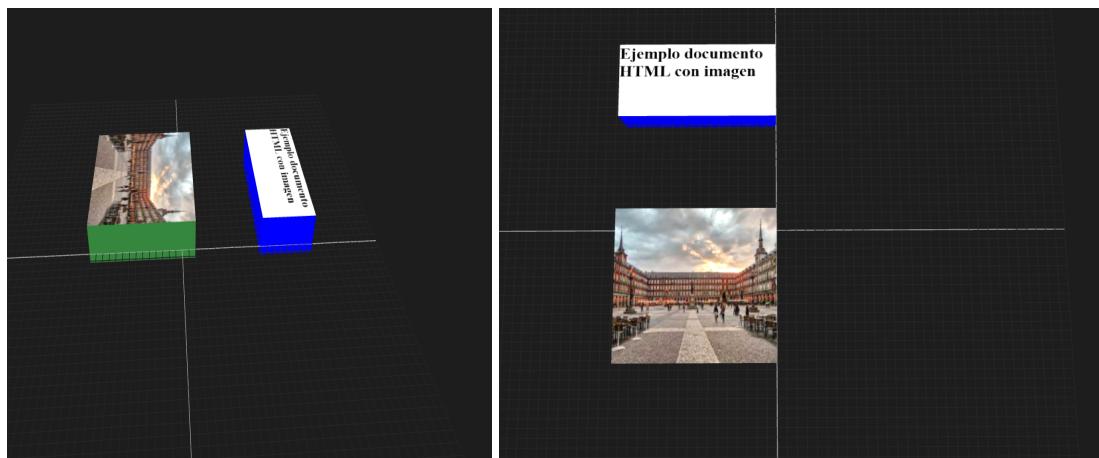


Figura 3.27: Escena A-Frame con html2canvas e imagen



(a) Escena html2canvas 2 Inspector 1 (b) Escena html2canvas 2 Inspector vista Top

Figura 3.28: Inspector A-Frame

Capítulo 4

Resultados

En este apartado, se detallan los pasos efectuados para obtener el resultado final entregable. Se parte del resultado obtenido en el sprint 3 y las funcionalidades extras comentadas en el sprint 4.

4.0.1. Manual de Usuario

En el apartado anterior, se ha detallado las etapas llevadas a cabo para construir una escena en la que visualizar el documento HTML en realidad virtual. Añadiendo además las funcionalidades extras comentadas anteriormente, a continuación se muestra las demos efectuadas como resultado final, para ver que el módulo implementado en el proyecto funciona para diferentes documentos HTML.

- Para la primera demo, se utiliza un diseño HTML basado en la página web del Departamento de Sistemas Telemáticos y Computación (GSyC) con menos componentes HTML para no saturar de entidades la escena final. El documento se divide en cuatro etiquetas *div* donde cada una define una sección del documento y agrupa varios elementos de bloque. El primer elemento *div* contiene el mismo banner de la página web del departamento, con el logo del GSyC y la foto del edificio.

```
<!-- Banner con logo Gsyc y Foto-->
<div id="visual-portal-wrapper">
  <table class="headerbg">
    <tr>
      <td class="headermid" >
        <a id="aPortalLogo" accesskey="1" href="index.html">
          | 
        </a>
      </td>
      <td class="headercorner"></td>
    </tr>
  </table>
</div>
```

Figura 4.1: Etiqueta div Banner HTML final

El segundo elemento *div* agrupa el logo de la Universidad Rey Juan Carlos y el contenido del texto, donde se ha incluido una descripción del departamento de Sistemas Telemáticos y Computación (GSyC) y un enlace de la ubicación del campus de Fuenlabrada.

```
<!-- Logo Urjc y Contenido Departamento-->
<div id="Contenido Departamento">
  <table id="portal-columna">
    <tr>
      <!-- Columna con el Logo-->
      <td id="portal-column-one">
        <div class="portalUrjc">
          <p>
            <a class="external-link" href="http://www.urjc.es/"><br/></a></p>
            <p> <a id="ets" href="http://www.etsit.urjc.es/">ETS. de Ingeniería de Telecomunicación</a></p>
          </div>
        </td>
      <!-- Columna con el Texto-->
      <td id="portal-column-two">
        <h1 id="TituloGsyC">Departamento de Sistemas Telemáticos y Computación (GSyC) </h1>
        <p id="Descripcion">
          El Departamento de Sistemas Telemáticos y Computación de la
          <a href="http://www.urjc.es/">Universidad Rey Juan Carlos</a> --
          Universidad Rey Juan Carlos
          (Madrid, España) tiene como actividades
          principales la docencia y la investigación en las áreas de Ingeniería
          Telemática y Ciencias de los Computadores. Poco más tarde, en 1995, su
          acrónimo es Gsyc (Grupo de Sistemas y Comunicaciones).&nbsp;Está adscrito a la
          Escuela Técnica Superior de Ingeniería de Telecomunicación
          <a href="http://www.etsit.urjc.es/"> Escuela Técnica Superior de Ingeniería de Telecomunicación</a> --
        </p>
        <p id="Ubicacion">Ubicación: <br> <br>&nbsp;Cómo llegar al&nbsp;
          <a class="external-link" href="http://www.urjc.es/comcollegar/fuenlabrada/c11_fuenlabrada.html">Campus de Fuenlabrada</a>.
        </p>
        <ul id="Ubicacion2">Cómo llegar al&nbsp;
          <a class="external-link" href="http://www.urjc.es/comcollegar/fuenlabrada/c11_fuenlabrada.html">Campus de Fuenlabrada</a>.
        </ul>
      </td>
    </tr>
  </table>
</div>
```

Figura 4.2: Etiqueta div Contenido HTML final

El tercer elemento *div* incluye únicamente una imagen de la Universidad Rey Juan Carlos, ubicada en la carpeta img dentro del directorio del proyecto.

```
<!-- Imagen Uni -->
<div>
  | 
</div>
```

Figura 4.3: Etiqueta div imagen Uni HTML final

Para apreciar mejor los elementos que componen el documento HTML y los respectivos hijos de cada uno, a continuación se representa el árbol de elementos del HTML utilizado:

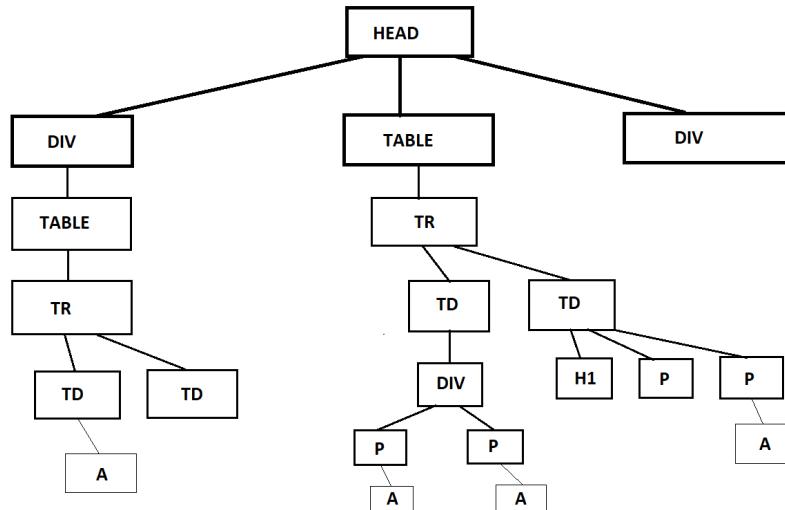


Figura 4.4: Representacion cajas HTML demo 01

El documento HTML final tiene el siguiente aspecto:

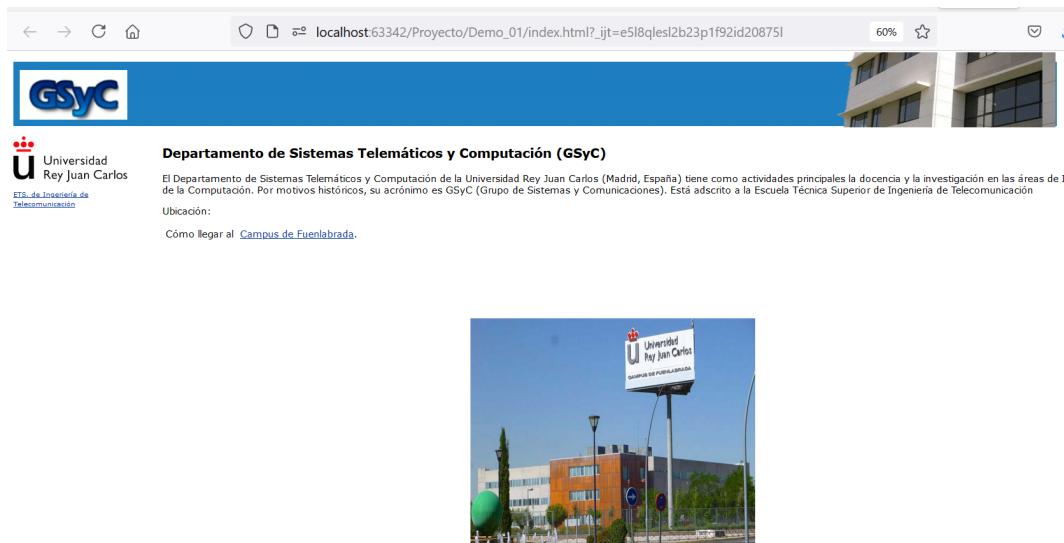


Figura 4.5: Documento HTML demo 01

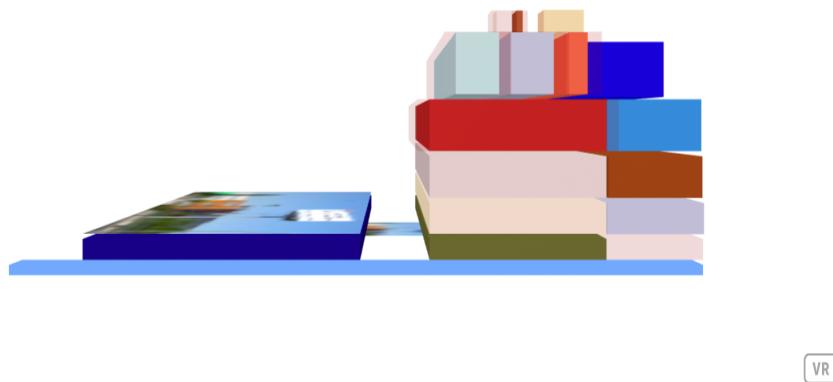


Figura 4.6: Escena demo 01

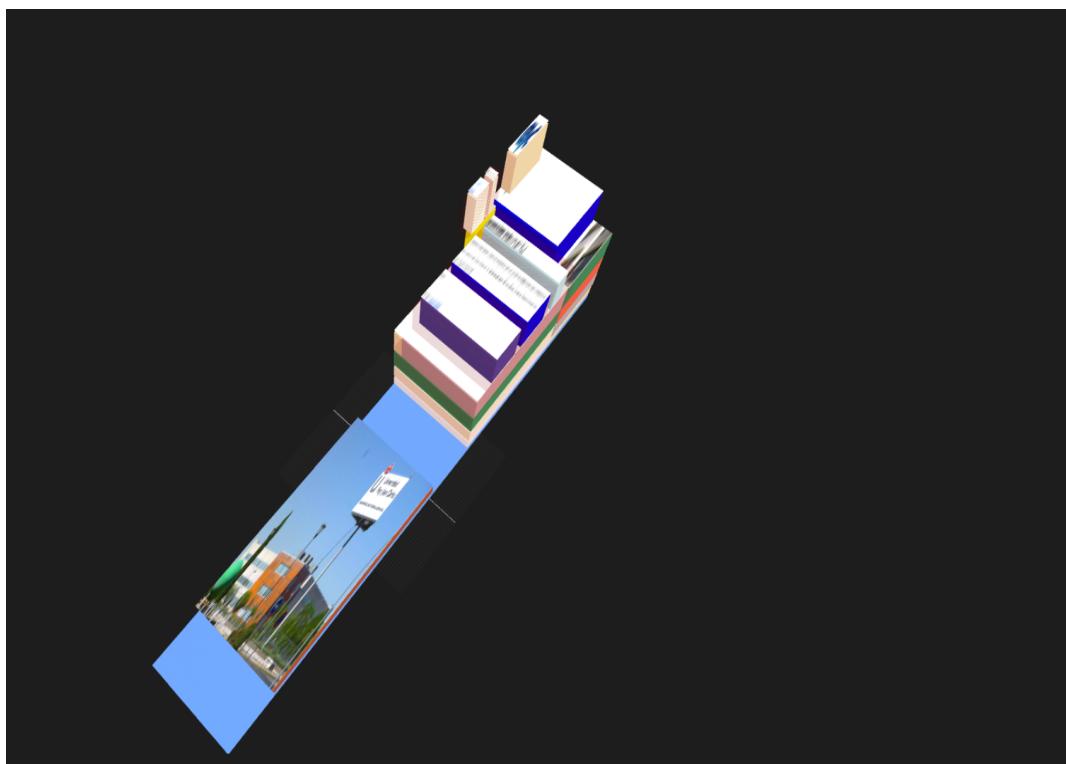


Figura 4.7: Inspector Escena demo 01

Otras perspectivas de la escena mediante el Inspector de A-Frame son las siguiente:

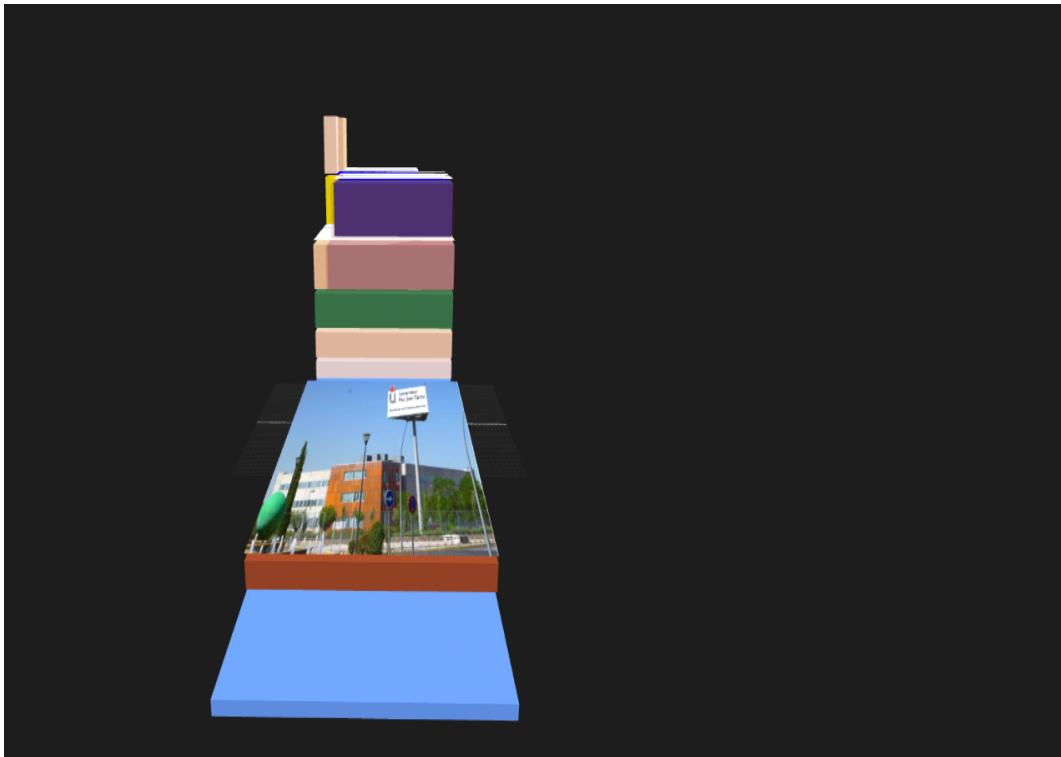


Figura 4.8: Inspector 2 Escena demo 01

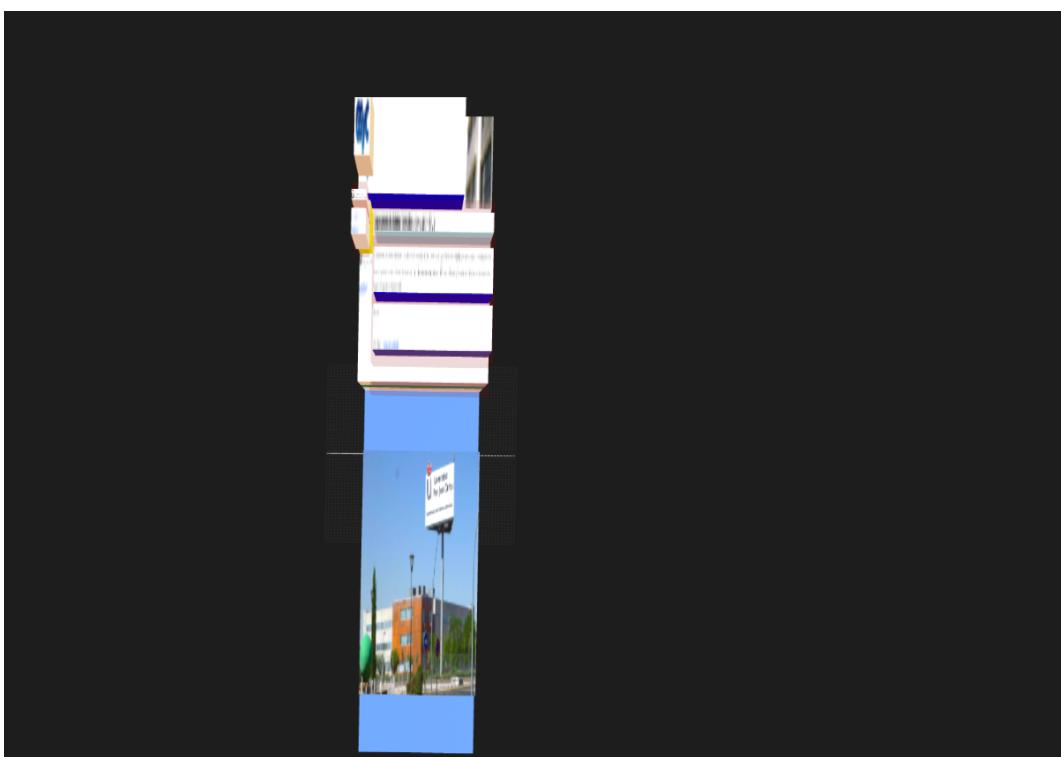


Figura 4.9: Inspector 3 Escena demo 01

- Para la segunda demo, utilice un documento HTML no muy sobrecargado de contenido al igual que en la demo anterior. Contiene un banner con el título y descripción, una lista de páginas que redireccionan a contenido del GSyC de la URJC y una imagen de la Universidad para ver más fácilmente la renderización de los elementos del HTML.

Para apreciar mejor los elementos que componen el documento HTML y los respectivos hijos de cada uno, a continuación se representa el árbol de elementos del HTML utilizado:

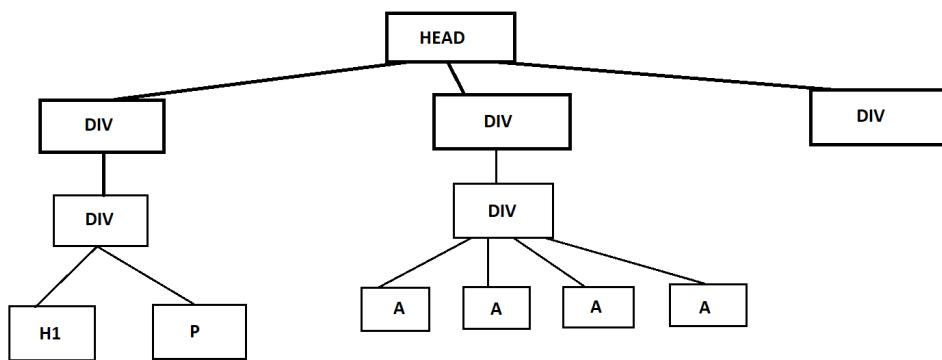


Figura 4.10: Representación cajas HTML demo 02

El documento HTML tiene el siguiente aspecto:

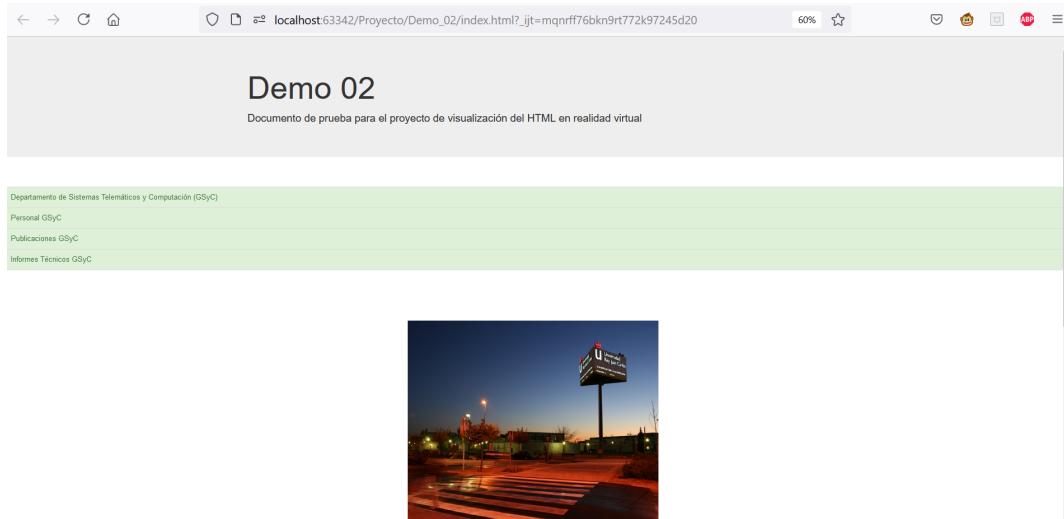


Figura 4.11: Documento HTML demo 02

A continuación se muestra el resultado de la escena, desde el navegador y desde el ins-

pector de A-Frame:

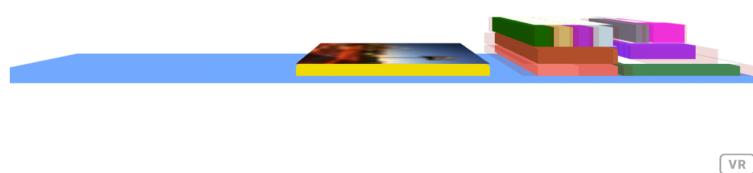


Figura 4.12: Escena demo 02

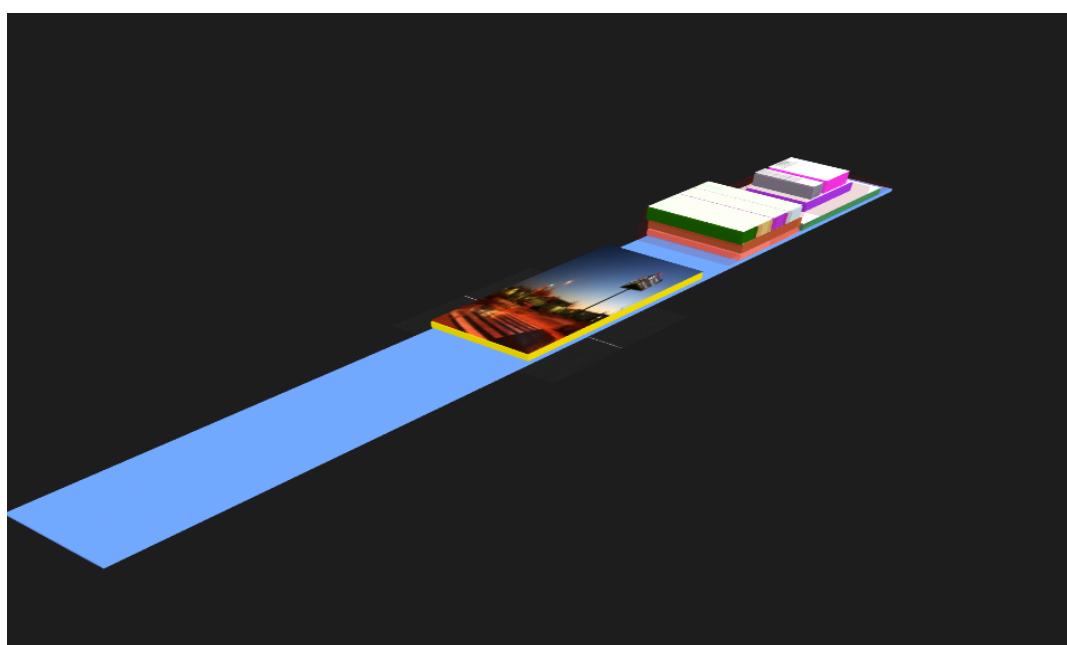


Figura 4.13: Escena Inspector 1 demo 02

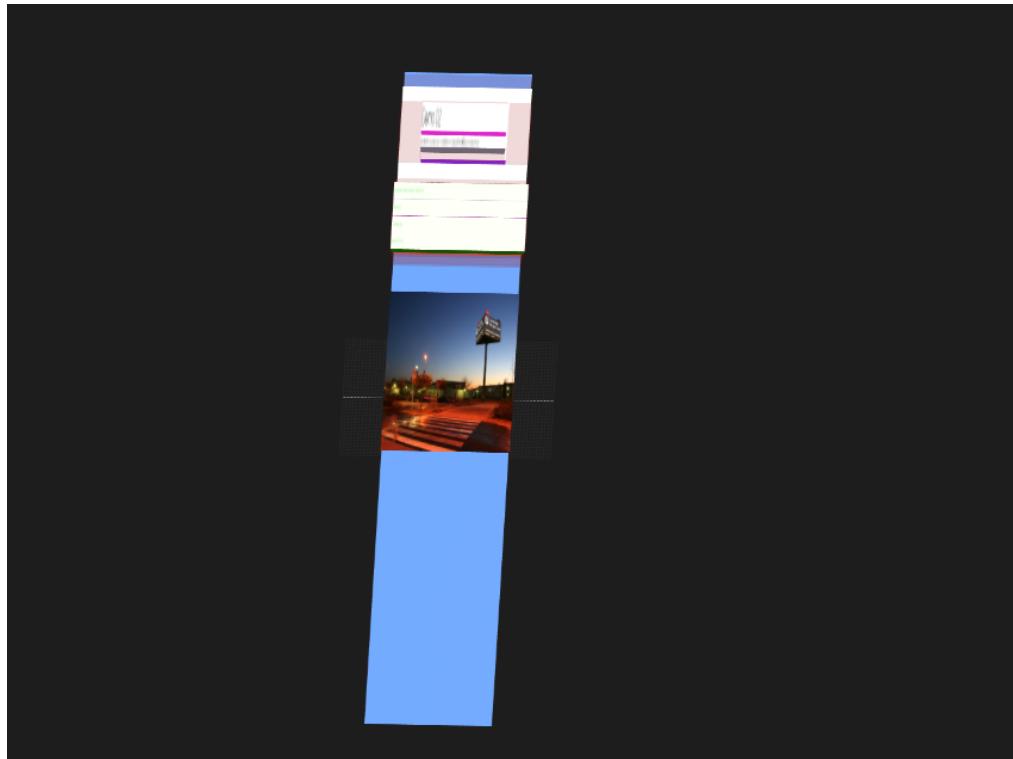


Figura 4.14: Escena Inspector 2 demo 02

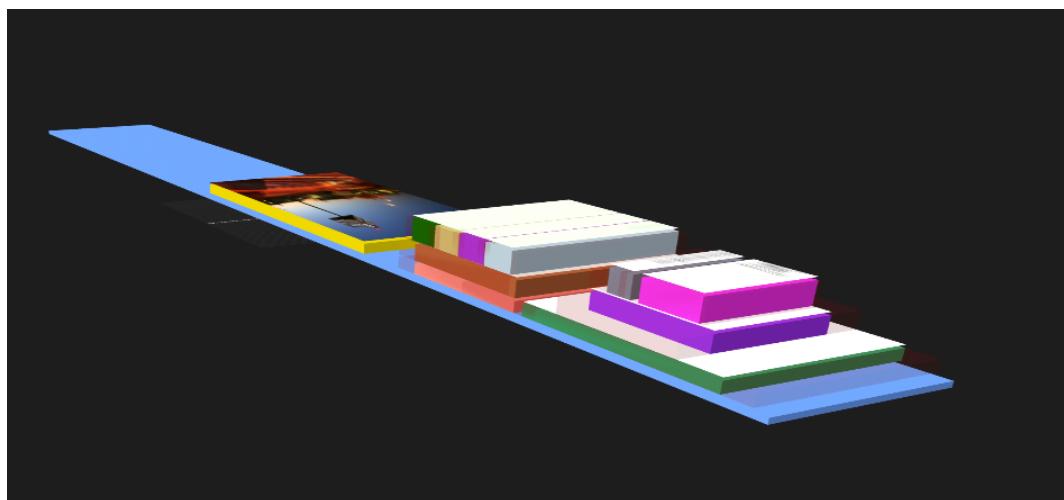


Figura 4.15: Escena Inspector 3 demo 02

- La tercera demo se lleva a cabo con un documento HTML con la siguiente estructura de árbol de elementos:

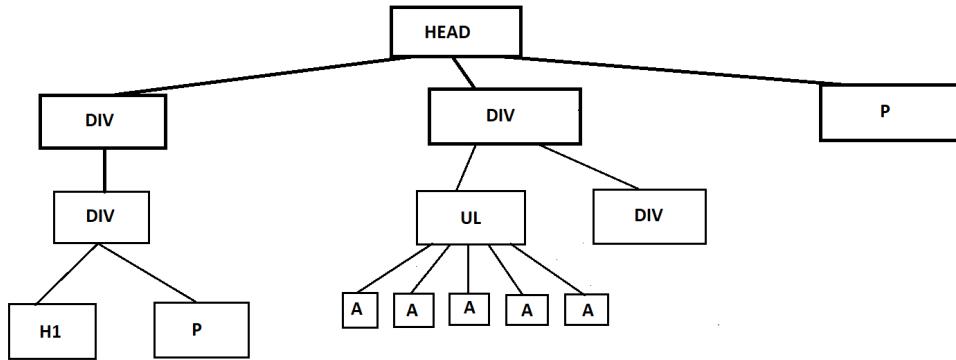


Figura 4.16: Representacion cajas HTML demo 03

La apariencia del HTML es la siguiente:

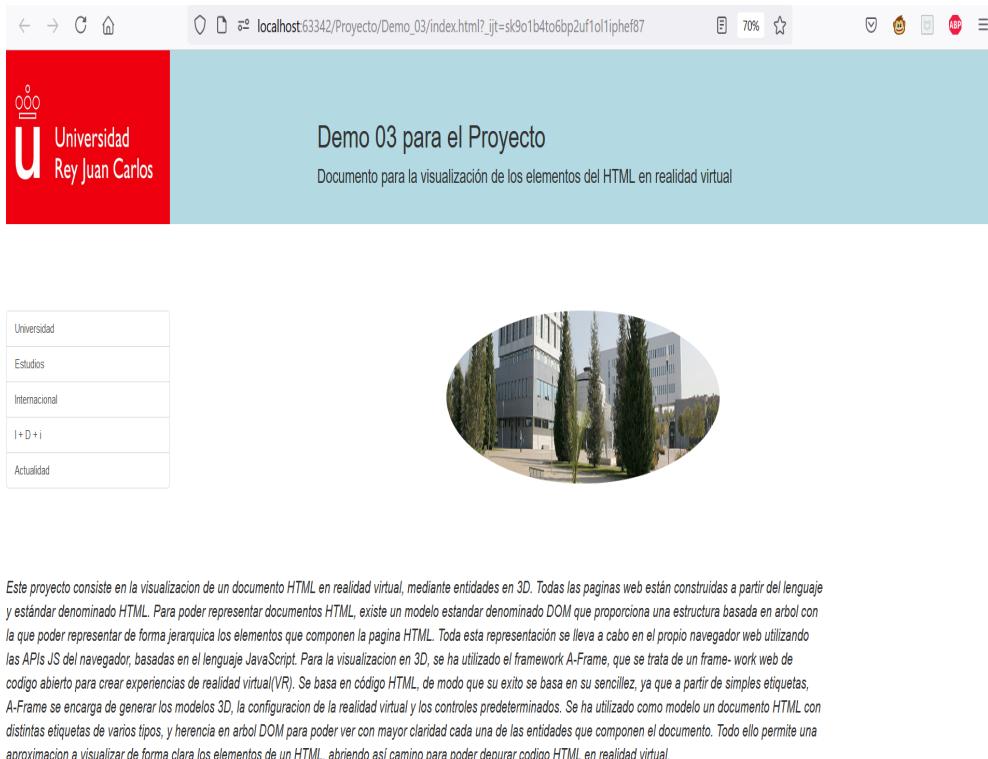


Figura 4.17: Documento HTML demo 03

A continuación se muestra el resultado de la escena, desde el navegador y desde el inspector de A-Frame:



Figura 4.18: Escena demo 03

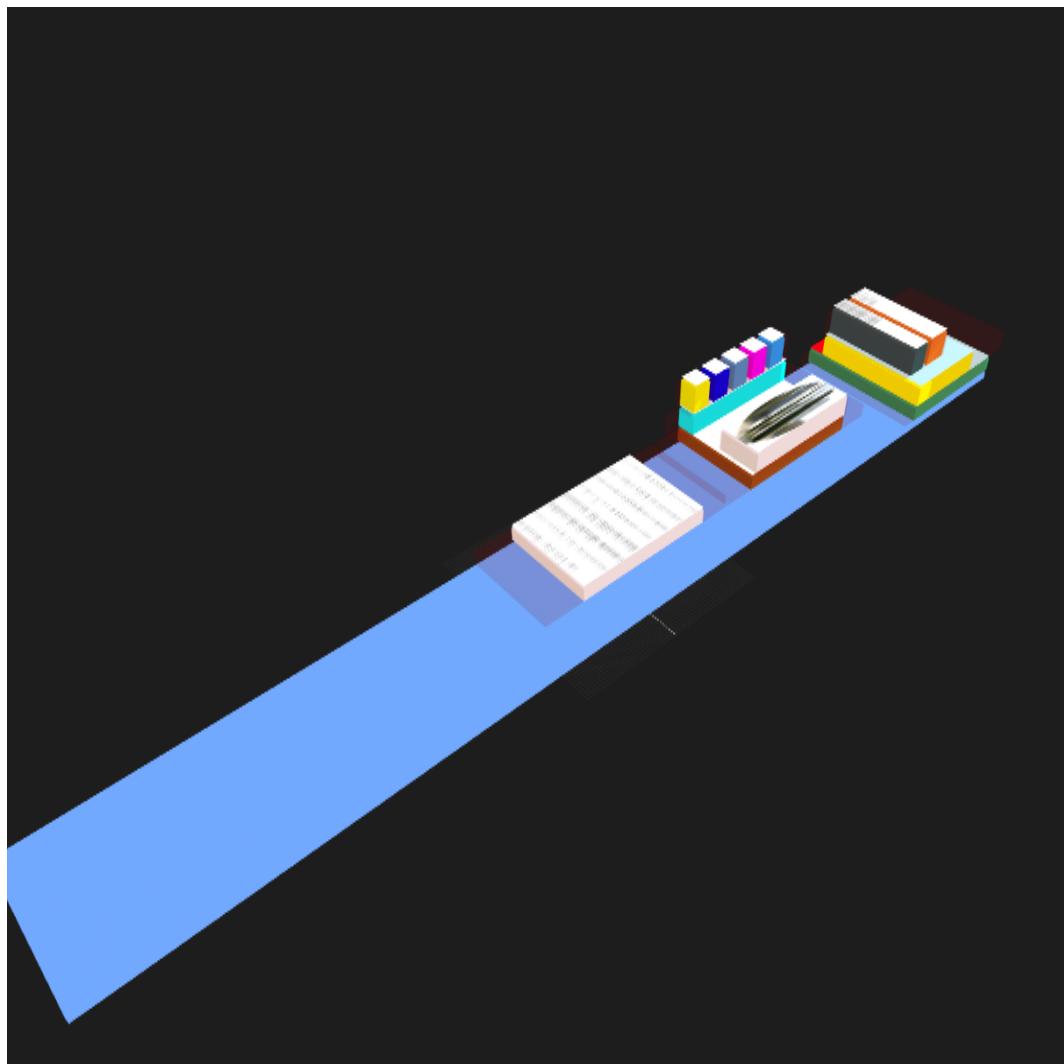


Figura 4.19: Escena Inspector 1 demo 03

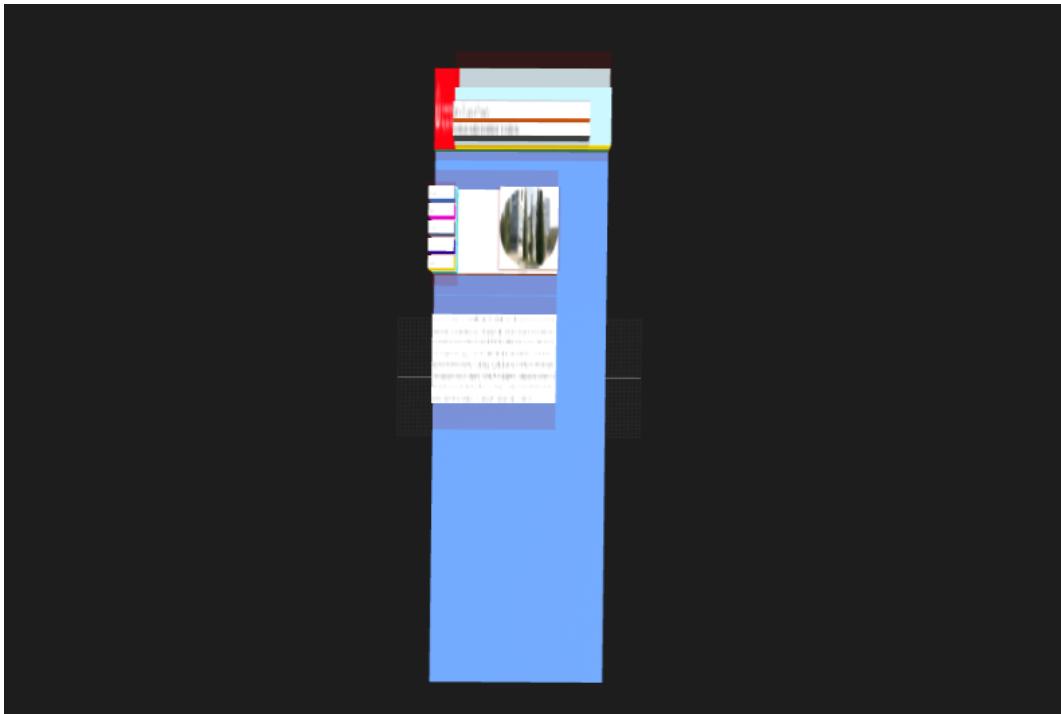


Figura 4.20: Escena Inspector 2 demo 03

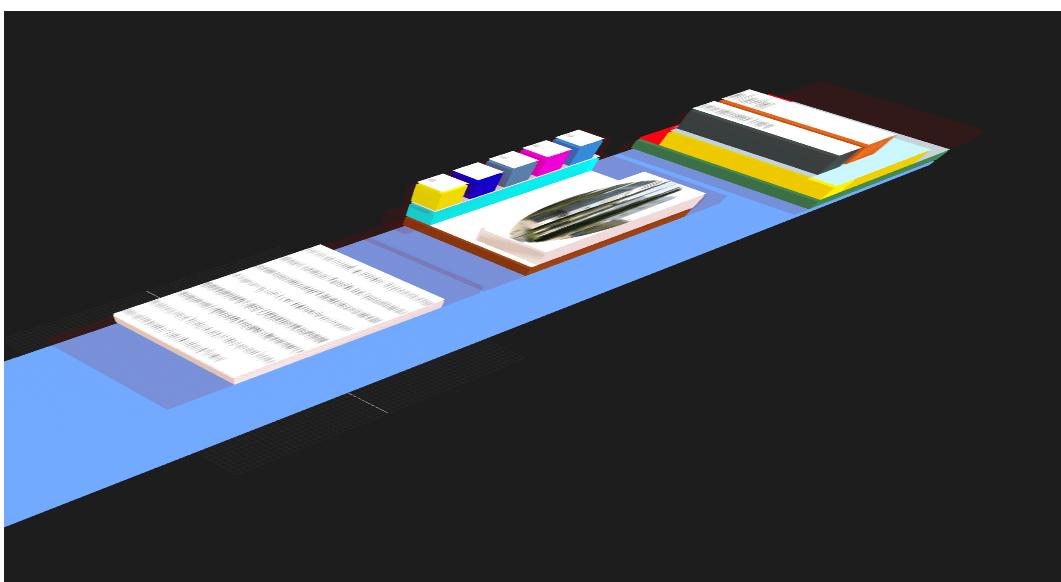


Figura 4.21: Escena Inspector 3 demo 03

Capítulo 5

Conclusiones

5.1. Consecución de objetivos

En el apartado relativo al objetivo general a conseguir en el desarrollo del proyecto, se comentaba que la principal finalidad era poder visualizar un documento HTML en realidad virtual. Este objetivo ha sido plenamente conseguido al poder representar en 3D el árbol de elementos DOM que componen una página web, a través de las posibilidades que nos ofrece A-Frame, visualizando de manera clara cada elemento del DOM, así como sus hijos.

La claridad y sencillez a la hora de visualizar cada elemento del DOM era uno de los pilares en el proyecto, y se puede observar en el resultado final como es fácil distinguir cada entidad en 3D que corresponde con su elemento HTML, ya que la colocación de los diferentes elementos sobre la entidad que representa al *Body* es exacta al orden que sigue en el documento HTML, es decir, la entidad colocada en primer lugar representa al primer elemento hijo del *Body* y así sucesivamente, solapando en el eje vertical los hijos que pudiera tener cada elemento. Además, la renderización de cada elemento como imagen para añadirla a cada entidad hace que se pueda ver la escena desde arriba como si fuera el propio documento HTML.

También se han podido conseguir los objetivos específicos que se plantearon al comienzo del desarrollo del proyecto, en especial el uso de frameworks nuevos para mí en su momento como era A-Frame, cuya documentación y ejemplos prácticos me sirvieron de gran ayuda para coger soltura con las múltiples posibilidades que nos ofrece. Aunque solo hemos visto una parte de lo que A-Frame nos ofrece, las experiencias en realidad virtual que podemos crear con él son muy amplias y diversas, y será aún mayor en el futuro. En cuanto a los lenguajes,

tanto HTML como JavaScript me eran conocidos, pero este desarrollo me ha permitido conocer nuevos aspectos de ambos y posibilidades que antes desconocía, como la visualización del árbol DOM de elementos o los componentes basados en JS que componen A-Frame.

5.2. Planificación temporal

La planificación del proyecto ha sido bastante atípica y duradera, sin lugar a dudas. Jesús González Barahona me presentó una idea inicial, allá por Febrero de 2016, que nada tenía que ver con la expuesta en este proyecto. Dicha idea consistía en la realización de diversos plugins que pudieran mejorar la visualización de datos obtenidos a través de una base de datos no relacional como es Elasticsearch y que se mostraban en la plataforma Kibana. Esta plataforma se trata precisamente de una herramienta de análisis y visualización de datos Elasticsearch, permitiendo la monitorización, consolidación y análisis de logs generados en múltiples servidores.

Durante el año 2016 llevé a cabo el análisis y desarrollo de esta idea inicial, sin llegar a nada concreto ni a alguna idea que pudiera evolucionar en algo concreto para poder desarrollar y posteriormente exponer. Además, por motivos personales y laborales, tuve que paralizar mi trabajo con el TFG durante el siguiente año.

Así, a principios de 2018, Jesús me habló del framework A-Frame y las posibilidades que ofrecía para la visualización en 3D y me resultó muy atractiva. Durante ese año llevé a cabo un análisis del proyecto a realizar, con objetivos como el completar mis conocimientos acerca del lenguaje de programación JavaScript, que no usaba desde hacia tiempo, indagar acerca del DOM y sus posibilidades o investigar el framework A-Frame y todas las funcionalidades que ofrecía. Además de esto, fui llevando a cabo pequeños hitos desde lo más básico como recorrerme los nodos del DOM o pequeños tutoriales de A-Frame.

Durante el año 2019 y hasta Marzo de 2020 aproximadamente, fui continuando con los hitos del proyecto, aumentando su complejidad y funcionalidad en cada uno, hasta poder representar una escena base en el navegador web.

A principios del 2021, empecé a redactar y dar forma a la memoria.

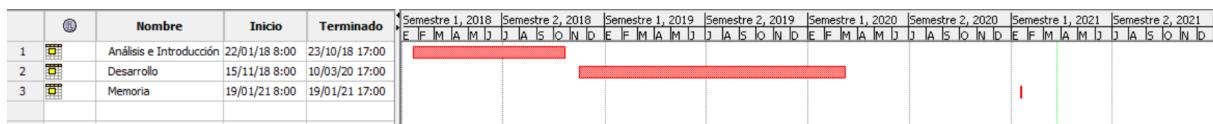


Figura 5.1: Planificación temporal

La duración de las jornadas ha cambiado a lo largo de la duración del proyecto.

- De Enero de 2018 a Marzo de 2020: Unas dos o tres horas, tres días a la semana, en el análisis y desarrollo del proyecto, con periodos de inactividad más o menos prolongados en este tiempo.
- De Enero de 2021 hasta la finalización del proyecto: Dos horas al día, unos dos o tres días a la semana, para la realización de la memoria.

5.3. Aplicación de lo aprendido

Durante los años de estudio en el Grado en Tecnología de las Comunicaciones, aprendí varios temas y conocimientos que me han servido y he podido poner en práctica en este trabajo.

Algunos de los más relevantes son:

- Conocimientos en lenguajes orientados a objetos como Java, en las primeras asignaturas de programación del Grado, me permitió tener una base en este tipo de lenguajes que luego sería fundamental en el aprendizaje de otros lenguajes como JavaScript.
- Conocimientos en lenguajes orientados al desarrollo de páginas web como HTML, CSS y JavaScript en asignaturas como Sistemas de Aplicaciones Telemáticas o Desarrollo de Aplicaciones Telemáticas, donde se llevaron a cabo prácticas como la implementación de un mashup de servicios o una revista de noticias de usuarios utilizando estos lenguajes mencionados, además de frameworks o librerías como JQuery, Bootstrap, etc. Estas asignaturas fueron de gran ayuda y me proporcionaron un conocimiento que ha sido fundamental para el desarrollo de este proyecto.
- Conocimientos en los sistemas de composición de textos LateX durante el tiempo que cursé las asignaturas más centradas en lenguajes orientados al desarrollo de contenido

web me ha permitido llevar a cabo este proyecto con este sistema teniendo una cierta base sólida a la hora ejecutarlo.

5.4. Lecciones aprendidas

La realización de este proyecto me ha permitido aumentar mis conocimientos en lenguajes o tecnologías que ya conocía y aprender nuevas tecnologías o frameworks muy útiles para el futuro. Cabe destacar entre lo aprendido lo siguiente:

- Entre los conocimientos aprendidos es fundamental el aprendizaje acerca de los entornos en realidad virtual, modelos en 3D y el framework A-Frame para la construcción de experiencias en realidad virtual. Mis conocimientos sobre todo ello antes de empezar el proyecto era casi nula y solo conocía de lejos las posibilidades que la realidad virtual podía ofrecer. Con la documentación e investigación inicial que llevé a cabo sobre la realidad virtual, la API WebXR y las experiencias en 3D, además de la posterior realización del proyecto, he descubierto que las posibilidades que ofrece y su impacto pueden llegar a ser enormes, permitiendo cosas inimaginables hace años como pueden ser eliminar las barreras geográficas, ya que cualquier persona podrá ver un partido de fútbol o un concierto desde su sofá sintiendo que está en el estadio in situ, o las ventajas didácticas que nos ofrece al permitir a una persona adquirir conocimientos mediante experiencias naturales y directas, de forma implícita, sin ser conscientes de estar aprendiendo algo, que al final es la mejor manera de que ese conocimiento adquirido se nos quede un mayor tiempo y de manera más clara.
- De una forma más específica, este proyecto me ha permitido conocer el framework A-Frame para la construcción de experiencias en realidad virtual. Este proyecto solo abarca una pequeña parte de lo que este framework nos puede ofrecer, pero sus posibilidades son también muy amplias, y ya existen demos y experimentos construidos con A-Frame y WebXR bastante alucinantes, como múltiples juegos arcade, tour virtuales donde puedes recorrer museos que contienen pinturas, esculturas y otros objetos animados, experimentos donde puedes ver movimientos de baile de la vida real presentados en un colorido entorno 3D, y así numerosos proyectos donde el impacto del framework es mayor, y podemos observar toda su utilidad y funcionalidad.

- En menor medida que los aspectos comentados anteriormente, pero también importante, debo recalcar que este proyecto me ha permitido ahondar en mis conocimientos sobre lenguajes orientados al desarrollo de contenido web como son HTML, CSS y JavaScript, ya que desde que terminé todas las asignaturas del Grado no he podido implementar muchos proyectos en estos lenguajes y mi camino laboral me ha llevado por otros caminos no tan relacionados con el contenido web. Por tanto, la realización de este proyecto me ha ayudado a recordar aspectos de estos lenguajes que había olvidado, adquirir nuevos conocimientos y tener la posibilidad de realizar un proyecto usando tecnologías y lenguajes con los que llevaba años sin implementar ninguna funcionalidad.
- La realización de la memoria usando el sistema de composición LateX me ha ayudado a escribir un proyecto tan complejo como este de forma más clara y mejor estructurada, sin tener que preocuparme en aspectos como la numeración, los índices o la colocación de las imágenes, ahorrándome tiempo y ayudándome a que me centrara únicamente en el contenido que quería para la memoria.

5.5. Trabajos futuros

Como trabajos futuros, a continuación expongo algunas ideas o posibles funcionalidades extras a implementar, que ya sea por falta de tiempo u otros motivos no he podido llevar a cabo:

- Una funcionalidad añadida a este proyecto que podría llevarse a cabo es la integración del mismo en alguno de los navegadores web, mediante el uso de extensiones como GreaseMonkey para Mozilla Firefox o TamperMonkey para Chrome.

El uso de estos plugins permite, mediante código proporcionado por el desarrollador, modificar el comportamiento de páginas web. Con esto, la idea es añadir nuestro código implementado y un simple botón a la hora de acceder a cualquier página web, de forma que al hacer click en él se mostrara en esa misma ventana o en otra el contenido HTML de esa página web en realidad virtual mediante entidades en 3D, de forma análoga a lo realizado en el proyecto con el documento HTML utilizado. A continuación se muestra un ejemplo de una página web en la que se ha añadido a través del plugin GreaseMonkey funcionalidad extra:



Figura 5.2: Página Web con Plugin GreaseMonkey

Como se observa, al acceder a páginas web como Google.com, se muestra un botón extra que indica que al hacer click en él se mostrará el contenido del árbol DOM de la página principal de Google en realidad virtual. Esta implementación no se pudo realizar por falta de tiempo, pero sin duda es un gran aliciente y una funcionalidad muy atractiva para poder visualizar cada página web que deseemos en realidad virtual y cada elemento del HTML como entidades en 3D.

- Ampliando la funcionalidad descrito en el apartado anterior, también se podría crear un plugin que implementara el módulo realizado, de forma que al utilizarlo se ejecutará el código y se visualizara el documento HTML en realidad virtual, sin necesidad de extensiones.
- Otras funcionalidades que se podrían implementar están relacionadas con las entidades en 3D creadas para la representación de cada elemento del HTML, de manera que el usuario pudiera hacer click sobre cada una de ellas y se recalcará el contenido HTML al que representa, o se mostrara la entidad sobre la que se ha pinchado en primer plano con los datos del contenido HTML representado, o incluso pudiéramos cambiar el orden de las entidades en 3D dentro de la entidad base sobre la que se posan, de manera que también cambiarán en el documento HTML. Habría muchas posibilidades a implementar sobre la propia escena creada y las entidades representadas en la misma, y en un primer momento me planteé llevar a cabo alguna, pero me ha sido imposible realizarlas por falta

de tiempo. Sin embargo, sería una manera muy llamativa visualmente de proseguir con este proyecto y jugar con las entidades en 3D.

- Otra idea sería jugar con la representación de la escena, insertando animaciones a la hora de visualizar cada entidad o añadiendo cierto comportamiento dinámico a los componentes. Por ejemplo, cada entidad que representa cada elemento HTML se puede ir añadiendo a la entidad base cada poco tiempo (unos pocos segundos) y rebotando sobre ella hasta acabar colocándose en su posición, o que cada entidad entrara en la escena por diferentes partes de la página, etc. A-Frame permite muchas opciones a la hora de representar la escena y ese dinamismo haría la escena mucho más sugerente en términos visuales.

Bibliografía

- [1] A-Frame Documentation. *Version 1.0.2.* <https://aframe.io/docs/1.2.0/introduction/>
- [2] MDN Web Docs <https://developer.mozilla.org/>
- [3] Jose Dimas Lujan Castillo. *HTML5, CSS y JavaScript. Crea tu web y apps con el estandar de desarrollo.* RC Libros, 2016.
- [4] Learn JavaScript. <https://www.codecademy.com/learn/introduction-to-javascript>
- [5] About Html2Canvas <https://html2canvas.hertzen.com/documentation>
- [6] J. Spurlock. *Bootstrap.* O'Reilly Media, Inc., 2013.
- [7] LaTeX Documentation <https://www.latex-project.org/help/documentation/>