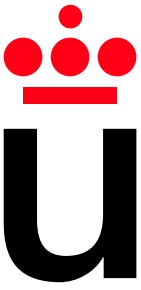

Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería Informática

Curso 2023-2024

Trabajo Fin de Grado

VISUALIZACIÓN DE PROYECCIONES DE DATOS EN ENTORNOS DE REALIDAD VIRTUAL EMPLEANDO LA LIBRERÍA A-FRAME DE JAVASCRIPT

Autor: Raúl Jiménez Trujillo
Tutor: Manuel Rubio Sánchez
Cotutor: Jesús María González Barahona

Agradecimientos

La entrega de este Trabajo de Fin de Grado representa la conclusión de una turbulenta etapa marcada por vaivenes emocionales, pero también por un enorme crecimiento personal y madurez que dará paso a la posibilidad de explorar incontables horizontes que jamás podría haber imaginado anteriormente.

Llegar hasta aquí supone haber cumplido con un objetivo que, durante más de seis años, ha sido prácticamente impensable y, todo sea dicho, de haber sido por mí mismo, habría acabado tirando la toalla. Es por eso por lo que quiero dedicar esta página a agradecer a todas las personas que se han quedado a mi lado y que me han abierto los ojos cuando yo no podía ver más allá.

Primero a mi familia por el apoyo incondicional desde el día uno en el que me embarqué en la aventura que para mí suponía comenzar una carrera universitaria.

A mis amigos que me han acompañado en cada una de las asignaturas y con los que he compartido momentos increíbles, pero también muy complicados. Pero sobre todo que me han hecho ver qué juntos éramos más fuertes y que entre nosotros sacábamos la mejor versión de cada uno.

Y por último, a Manuel y Jesús le doy las gracias por darme la oportunidad de estar bajo su tutela durante la elaboración de este proyecto, por sus consejos, sus observaciones, sus puntos de vista y, sobre todo, por su dedicación y su entusiasmo.

Resumen

A raíz de la necesidad de poder mostrar proyecciones de datos creadas en un entorno de tres dimensiones, nace la idea de este Trabajo de Fin de Grado. Mediante el uso de la librería *A-FRAME* y la visualización en Realidad Virtual, se consigue una experiencia inmersiva que nos traslada a una zona en la que podremos interactuar de diferentes maneras con las proyecciones previamente mencionadas.

En concreto, este trabajo describe el diseño y desarrollo de varios componentes de *A-FRAME*, junto a métodos de tratamiento de datos. Su finalidad, además de la representación de baja dimensionalidad de una muestra de datos como se realiza en Star Coordinates, podría ser la de una visualización de datos estáticos para la que el usuario no necesitase ningún conocimiento matemático.

De esta forma, este Trabajo de Fin de Grado se presenta como una forma de estudiar representaciones gráficas de datos en entornos controlados con la ayuda de dispositivos externos como son las gafas de Realidad Virtual, con las que podremos, además de analizar los resultados con más claridad, manipular de manera sencilla y única los elementos que tengamos en escena.

Palabras clave:

- Realidad Virtual
- A-FRAME
- Análisis de datos
- Star Coordinates
- Proyección Lineal

Índice de contenidos

Índice de tablas

Índice de figuras

Índice de códigos

1 Introducción

1.1. Contexto y motivación

Star Coordinates (SC) [10] es un método de visualización de datos numéricos de dimensión elevada. Reproduce proyecciones lineales de los datos a un espacio observable (de 2 o 3 dimensiones), con la ventaja de incorporar información tanto de los datos como de las variables en el mismo gráfico.

Generalmente, los datos se representan mediante puntos, mientras que las variables se ilustran con vectores y/o ejes etiquetados. SC se considera un método interactivo, ya que los usuarios pueden manipular la longitud y orientación de los vectores para generar diferentes proyecciones de los datos, permitiendo visualizarlos desde varios puntos de vista. De esta manera, SC se usa para realizar diversas tareas de análisis de datos, como detectar clusters o valores atípicos, o para buscar datos con características especiales.

Se puede detectar un problema a la hora de visualizar los resultados, ya que se encuentran en un espacio de dos dimensiones que dificultan tanto el análisis posterior como la estimación de los atributos.

Para resolver la limitación de dos dimensiones, se propone un espacio en tres dimensiones para trabajar con SC, ofreciéndose un entorno en realidad virtual para hacer más fácil el manejo y visualización de las proyecciones de los datos.

Tras el análisis de lo que podría significar convertir el plano de dos dimensiones a una escena en realidad virtual que permita usar tres dimensiones, se definen los siguientes beneficios:

- Mayor dimensionalidad: añadir una dimensión más con la profundidad para los componentes de la escena, aporta una nueva visión de los datos más fidedigna y mayor facilidad para interpretar los resultados obtenidos. Además de la posibilidad de ayudar con uno de los problemas o desventajas que encontramos en SC, como sería la dificultad de los usuarios para ubicar los vectores de ejes.

- Mayor facilidad para interactuar con los elementos gráficos: el uso de tecnologías de realidad virtual permiten vivir una experiencia inmersiva que añade profundidad y mejora la comprensión de la escena. A lo que es posible añadir la realidad aumentada, con funcionalidades que podrían permitir al usuario integrar la visualización de datos con su entorno en el mundo real, dando lugar a demostraciones en las que el usuario pudiera encontrarse en su puesto de trabajo mientras, mediante una videollamada, otros participantes pudiesen formar parte de la visualización de datos, encontrándose todos dentro del mismo entorno. Además, las tecnologías de realidad virtual están en auge en los últimos años, con una capacidad de mejora y progreso muy elevada debido a su versatilidad. Un campo que podría evolucionar mucho más de lo que lo hará la representación de datos en dos dimensiones, ya que con casos como este, podemos ver que está limitada.

- Nuevas funcionalidades y elementos gráficos: las escenas creadas con la ayuda de componentes de la librería de *A-FRAME* dan lugar a un extenso espectro de posibilidades a la hora de añadir o eliminar atributos de los mismos. Por otro lado, se facilita el uso de componentes de terceros que pueden aportar valor a las demostraciones o incluso una mayor personalización de los tuyos propios.

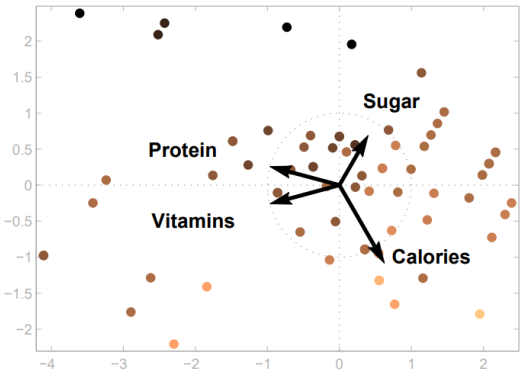

Figura 1.1: Representación de datos de baja dimensionalidad con SC en dos dimensiones. Tomado de [1].

1.2. Estructura del documento

El presente Trabajo de Fin de Grado se estructura en los siguientes capítulos:

- **Capítulo 2: Objetivos**

  En este capítulo se analizarán los objetivos de este proyecto.

- **Capítulo 3: Antecedentes y estado del arte**

  En esta parte se llevará a cabo el análisis de algunos métodos de visualización de datos, finalizando con un repaso por las tecnologías utilizadas y por último, un estudio de alternativas.

- **Capítulo 4: Descripción Informática**

  Este capítulo recorre desde los requisitos del proyecto, pasando por las metodologías comúnmente utilizadas y la arquitectura del sistema, hasta llegar a la implementación. Por último, también se analizan las dificultades encontradas a lo largo del desarrollo.

- **Capítulo 5: Pruebas y evaluación** Por un lado, se revisarán las pruebas llevadas a cabo con usuarios para analizar el funcionamiento del trabajo realizado. Por el otro lado, con el uso de siete principios, habitualmente utilizados para este tipo de evaluaciones, se analizará desde la portabilidad, hasta el rendimiento o mantenibilidad del proyecto.

- **Capítulo 6: Conclusiones y trabajos futuros**

  En este apartado se analizará el grado en el que se han alcanzado los objetivos del trabajo y los posibles trabajos futuros que se pueden llevar a cabo, aportando más valor al desarrollo ya realizado.

2

Objetivos

El objetivo principal del proyecto es la creación de un prototipo de adaptación de los gráficos de baja dimensionalidad visibles con Star Coordinates en una aplicación de realidad virtual procesables con la librería de *A-FRAME* [11], un entorno de desarrollo web que permite construir experiencias en realidad virtual haciendo uso de *three.js* [12], proporcionándole una estructura de componentes y entidades capaz de construir y combinar elementos de manera flexible y modular.

Por medio de estas librerías, será posible mostrar gráficamente en un entorno de tres dimensiones lo que se podía ver en la Figura 1.1. En el siguiente apartado se pasará a desarrollar esos componentes y entidades de manera más detallada.

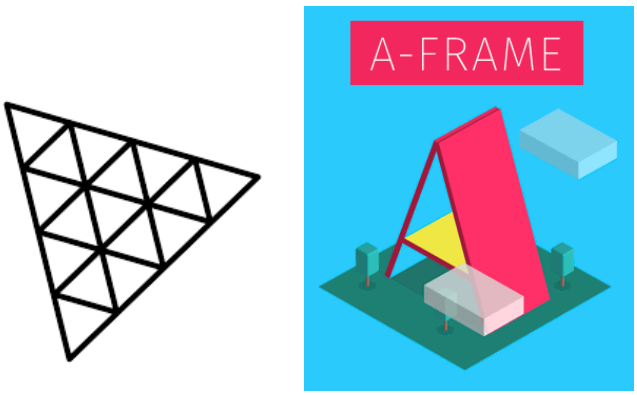

Figura 2.1: Logotipos de three.js y A-FRAME. Tomados de [2] y [3].

De manera específica, los objetivos delineados para este proyecto pueden ser subdivididos en cuatro problemáticas distintas, cada una abordando aspectos cruciales para el desarrollo y la funcionalidad del sistema.

En primer lugar, el objetivo más elemental y fundamental consistía en exhibir la representación de datos, asegurando una clara diferenciación de colores en las esferas que representan los ejes vectoriales. La importancia de esta meta radica en la necesidad de proporcionar una visualización nítida y comprensible de la información, permitiendo al usuario identificar de manera intuitiva cada uno de los ejes.

Tras alcanzar con éxito el primer objetivo, se planteó la necesidad de permitir al usuario visualizar diversas representaciones y facilitar la transición entre ellas de manera sencilla. Este propósito se materializó mediante la incorporación de selectores y botones, como se evidencia en varias instancias de las pruebas detalladas en el capítulo Evaluación y pruebas. La implementación de estos elementos interactivos no solo amplió la versatilidad del sistema, sino que también mejoró la experiencia del usuario al proporcionarle un medio intuitivo y eficaz para cambiar entre diferentes representaciones según sus necesidades o preferencias.

Como aspecto crucial entre los objetivos establecidos, se perseguía la funcionalidad de permitir la modificación constante de la posición de los ejes vectoriales. Este hito se alcanzó mediante la implementación del componente 'super-hands'. Sin embargo, este objetivo no se cumpliría completamente si, al mover uno de los ejes, no se recalculasen los datos de manera precisa. Este desafío se abordó eficazmente mediante el empleo de la posición relativa y la aplicación de proporciones. La utilización de estos conceptos aseguró la coherencia y la precisión en la representación de los datos.

Como último objetivo, decidido durante el transcurso del desarrollo del proyecto, se implementó el mirror con la intención de mejorar la accesibilidad. Esta adición tiene como finalidad facilitar la tarea de mover los ejes vectoriales al proporcionar una visión más amplia y general del gráfico, así como de las posiciones de los datos. La inclusión del mirror pretende ofrecer al usuario una perspectiva extendida que simplifique la manipulación de los ejes vectoriales, contribuyendo así a una experiencia más intuitiva y eficiente en la interacción con el sistema.

3

Antecedentes y estado del arte

3.1. Métodos de visualización

Existen multitud de métodos de visualización [13]. Este apartado menciona algunos de los más conocidos y utilizados.

3.1.1. Scatterplot Matrix

Una matriz de diagramas de dispersión, Scatterplot Matrix [14], es una cuadrícula (o matriz) de gráficos de dispersión utilizada para visualizar relaciones bi variadas entre combinaciones de variables.

El diseño consta de dos mitades divididas a lo largo de una diagonal. Es posible configurar cada mitad de la matriz, así como la diagonal, según las necesidades de visualización. Cada gráfico de dispersión en la matriz representa la relación entre un par de variables, lo que permite explorar muchas relaciones en un solo gráfico.

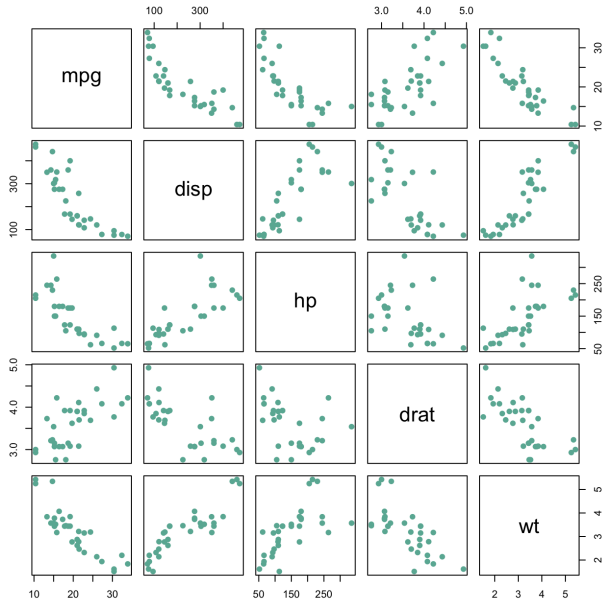

Figura 3.1: Ejemplo de ScatterPlot Matrix con datos de 'mtcars'. Tomado de [4].

3.1.2. Parallel Coordinates Plot

Este tipo de visualización se utiliza para representar datos numéricos multivariados. Representaciones generadas mediante el uso de Parallel Coordinates Plots [15] son ideales para comparar muchas variables juntas y observar las relaciones entre ellas. Por ejemplo, si tuvieras que comparar una variedad de productos con los mismos atributos.

A cada variable se le asigna un eje y todos los ejes se colocan en paralelo entre sí. Cada eje puede tener una escala diferente, dependiendo de la unidad de medida con la que funcione cada variable, o todos los ejes pueden normalizarse para mantener las escalas uniformes. Los valores se representan como una serie de líneas que están conectadas a lo largo de todos los ejes. Esto significa que cada línea es una colección de puntos colocados en cada eje, todos conectados. Se puede ver en este ejemplo.

El orden en que se disponen los ejes puede afectar la comprensión de los datos por parte del lector. Una razón para esto es que las relaciones entre variables adyacentes son más fáciles de percibir que para variables no adyacentes. Por lo tanto, reorganizar los ejes puede ayudar a descubrir patrones o correlaciones entre variables.

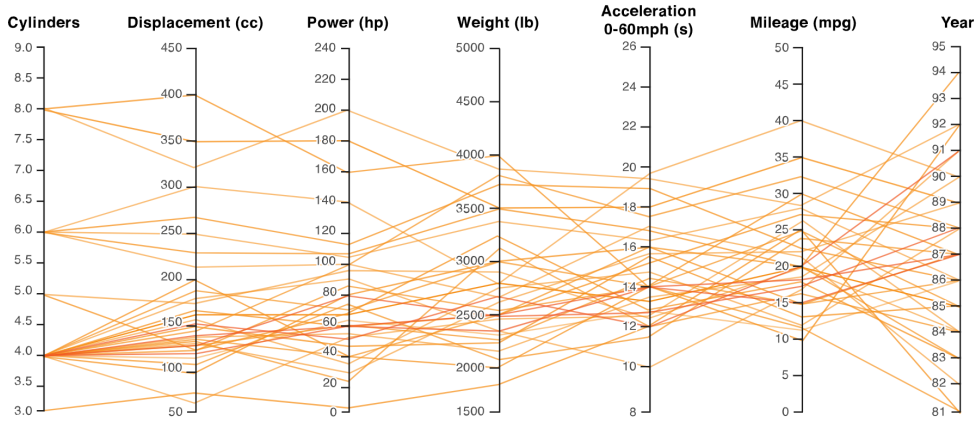

Figura 3.2: Ejemplo de Parallel Coordinates Plot con datos de automóviles. Tomado de [5].

3.1.3. t-SNE

t-SNE [16], t-distributed Stochastic Neighbor Embedding, es un algoritmo utilizado para la visualización de conjuntos de datos de alta dimensionalidad en espacios de dimensiones más bajas, comúnmente dos dimensiones. Fue desarrollado como una extensión de la técnica de Stochastic Neighbor Embedding (SNE) por Laurens van der Maaten y Geoffrey Hinton.

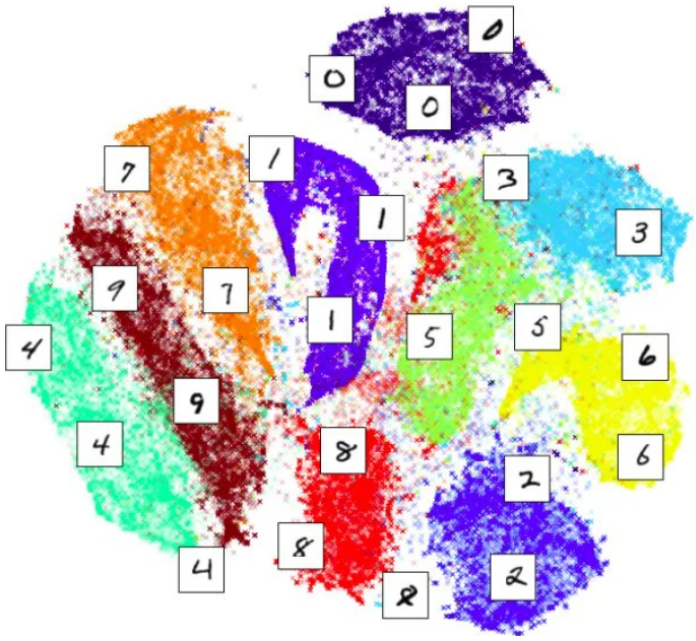

Figura 3.3: Ejemplo de t-sne de escritura a mano de dígitos. Tomado de [6].

La principal utilidad de t-SNE radica en su capacidad para preservar las relaciones locales entre puntos en el espacio original de alta dimensión durante la proyección a un espacio de menor dimensión. En otras palabras, puntos similares en el espacio de alta dimensión tienden a estar cercanos en el espacio de baja dimensión, y puntos diferentes tienden a estar más alejados. Esto hace que t-SNE sea especialmente útil para visualizar patrones complejos y estructuras en datos de alta dimensión.

En el ejemplo de la figura 3.3 podemos ver el análisis de datos que repasa la escritura a mano de los diez primeros dígitos, relacionando entre sí los datos donde podemos ver, por ejemplo, que el '9' y el '4' están ubicados en posiciones cercanas, ya que pueden ser confundidos dependiendo de la escritura que se utilice, el mismo caso ocurre con el '5' y el '6' o algunos '8' y '5'.

3.1.4. U-MAP Visualization

UMAP [17], Uniform Manifold Approximation and Projection, es un algoritmo de reducción de dimensionalidad no lineal utilizado para visualizar y analizar conjuntos de datos de alta dimensionalidad. Al igual que otros algoritmos de reducción de dimensionalidad, como el anterior algoritmo explicado t-SNE, UMAP busca representar datos complejos en un espacio de dimensiones más bajas para facilitar la interpretación visual y el análisis.

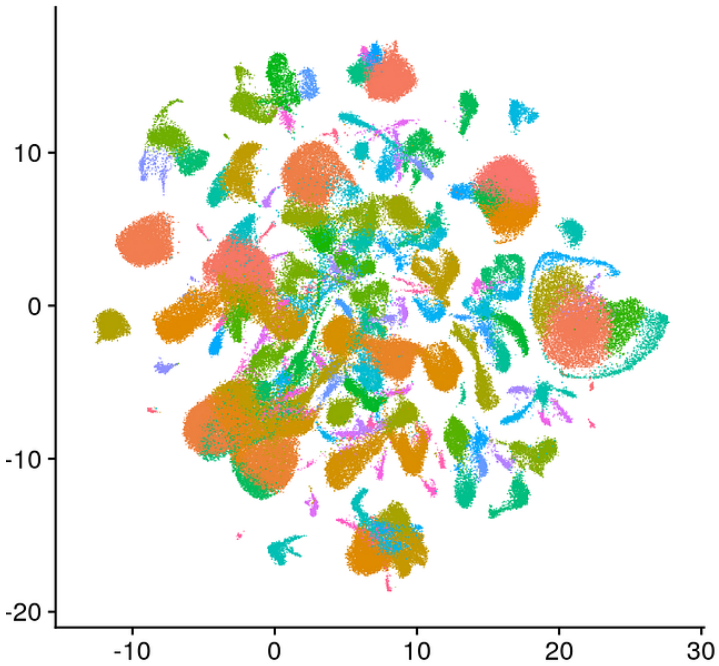

Figura 3.4: Ejemplo de UMAP con datos de [7]. Tomada de [8].

Se ha vuelto popular en la comunidad de aprendizaje automático y análisis de datos debido a su capacidad para preservar las estructuras complejas y no lineales en los datos de alta dimensión, así como su eficiencia computacional. Se utiliza comúnmente en visualización de datos y exploración de patrones en disciplinas como la ciencia de datos, bioinformática y más.

3.1.5. Star Coordinates

Por último, es importante mencionar Star Coordinates [10], ya que es el algoritmo en el que se basa este trabajo. Aunque ya se explicó en el apartado Contexto y motivación, cabe destacar algún inconveniente de este algoritmo, como puede ser que utiliza una representación gruesa de los datos. Por lo que para ajustar todas las entradas en el espacio bidimensional, los datos deben normalizarse.

Como resultado, no se puede realizar un análisis numérico. Además, hay muchas formas de mapear al mismo punto en el espacio. Las representaciones creadas con SC no suelen ser fáciles de leer y, por lo tanto, están principalmente limitadas a la analítica de datos. Todo esto, se refiere a representaciones en dos dimensiones.

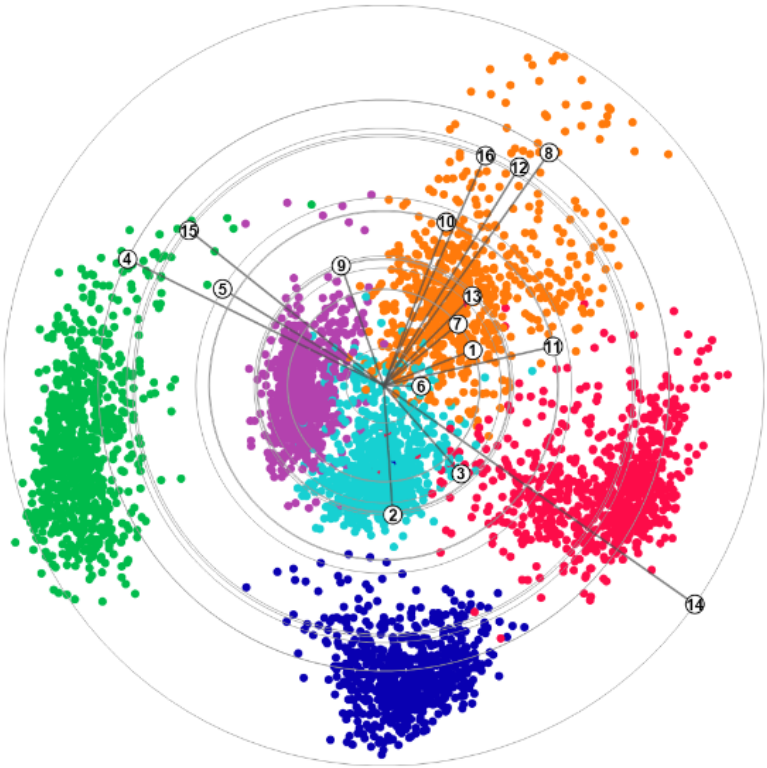

Figura 3.5: Ejemplo de Star Coordinates. Tomado de [9].

Pero antes de poder mostrar esos datos se necesitará realizar el tratamiento oportuno de los mismos. Este consiste en la multiplicación matricial de los dos archivos. El archivo A contiene una matriz formada por el número de muestras por la cantidad de ejes vectoriales que se desean tener en cuenta.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nn} \end{bmatrix} \tag{3.1}$$

Por otro lado, tenemos el archivo B con la cantidad de ejes vectoriales por el número de dimensiones disponibles, siendo este caso tres.

$$\mathbf{V} = \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ \vdots & \vdots & \vdots \\ v_{n1} & v_{n2} & v_{n3} \end{bmatrix} \tag{3.2}$$

De esta manera tendremos el siguiente cálculo:

$$\mathbf{P} = \mathbf{X} \times \mathbf{V} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ \vdots & \vdots & \vdots \\ p_{N1} & p_{N2} & p_{N3} \end{bmatrix} \tag{3.3}$$

Esta matriz resultante será la que se utilizará para mostrar los datos, previamente normalizados, en el gráfico. Serán datos estáticos que podrán modificarse haciendo un uso dinámico de los valores de los ejes vectoriales representados en el archivo referente a la matriz B.

3.2. Tecnologías Utilizadas

A continuación se paran a detallar las tecnologías utilizadas para poder llevar a cabo del desarrollo de este trabajo.

3.2.1. Meta Quest 2

Antes de centrarse en la parte de programación, es importante mencionar las gafas de realidad virtual utilizadas en todo momento para el desarrollo y pruebas del proyecto.

Las Meta Quest 2 [18], pertenecientes a la compañía Meta [19], anteriormente conocidas como Oculus Quest, son un sistema de realidad virtual inalámbrico compuesto por gafas, auriculares, micrófono y mandos también inalámbricos.

Creadas en 2020, son el dispositivo más utilizado para cualquier actividad relacionada con la realidad virtual.

Figura 3.6: Meta Quest 2.

3.2.2. JavaScript

JavaScript es un lenguaje de programación ligero, interpretado y orientado a objetos con funciones de primera clase. Aunque es reconocido principalmente por su papel como el lenguaje de programación para las páginas web, también se utiliza en diversos entornos que no requieren un navegador.

Es un lenguaje basado en prototipos, multiparadigma y de un solo hilo. Además, es dinámico y ofrece soporte para la programación orientada a objetos, tal como se mencionó anteriormente. JavaScript[20] es versátil, permitiendo

enfoques tanto imperativos como declarativos en el desarrollo de software. Su naturaleza dinámica y su capacidad para ser ejecutado en diferentes contextos hacen que sea una opción popular para una amplia gama de aplicaciones y proyectos.

Cabe destacar que aunque A-FRAME y three.js son también tecnologías utilizadas y pertenecen a JavaScript, formando parte del amplio catálogo de librerías disponibles de este lenguaje, podremos verlo más en detalle en el apartado de Estudio de alternativas.

Para la estructuración de las escenas y creación de demostraciones, se utiliza el lenguaje HTML[21], la relación entre HTML y JavaScript es fundamental para el desarrollo de páginas web dinámicas y atractivas. HTML actúa como el esqueleto estructural de una página web, definiendo la manera en que se presenta y organiza el contenido. Por otro lado, JavaScript agrega interactividad y dinamismo a la experiencia del usuario.

HTML se encarga de la estructura básica de una página web, como encabezados, párrafos, listas y enlaces. Sin embargo, cuando se busca una experiencia más allá de la simple presentación estática de información, es necesario incorporar JavaScript. Este lenguaje de programación permite realizar acciones como validaciones de formularios, animaciones, actualizaciones en tiempo real y manipulación del contenido de la página sin necesidad de recargarla.

La combinación de HTML y JavaScript permite crear aplicaciones web interactivas y altamente funcionales. HTML proporciona la base estructural, mientras que JavaScript agrega la capa dinámica, permitiendo a los desarrolladores crear experiencias de usuario más atractivas y receptivas. En conjunto, estos dos lenguajes desempeñan un papel crucial en la creación de páginas web modernas, mejorando significativamente la interactividad y la usabilidad para los visitantes. En resumen, HTML y JavaScript colaboran estrechamente para ofrecer una experiencia web completa y satisfactoria.

Figura 3.7: Logos CSS - HTML - JS.

Por otro lado, la presentación visual y el diseño de la página, incluyendo aspectos como colores, tipografías y diseño responsivo, son gestionados por CSS [22]. Este lenguaje complementa la estructura (HTML) y la interactividad (JavaScript), asegurando una experiencia web completa y atractiva para los usuarios.

3.3. Estudio de alternativas

Como ya se ha mencionado anteriormente, la idea inicial, que se pudo llevar a cabo hasta el final, fue la de utilizar las herramientas previamente nombradas. A continuación se detallarán que componentes y funcionalidades nos ofrece *A-FRAME* con los que podamos cumplir los requisitos.

La característica principal de la librería sería la de usar y crear componentes, pudiendo personalizarlos mediante el uso de sus atributos. Pero para ello necesitamos entidades, estas son objetos a los que se les añaden componentes que les proporcionan su comportamiento, funcionalidad y apariencia.

Inicialmente, se pone a nuestra disposición las entidades primitivas, de las que podemos hacer uso con sus comportamientos predeterminados, pero que también ofrecen la posibilidad de adquirir nuevas funcionalidades agregando componentes creados por nosotros mismos.

Vamos a revisar en detalle algunos de los primitivos utilizados:

- *a-entity* - Es la etiqueta de entidad básica, no tiene apariencia ni atributos, aunque todas las entidades están intrínsecamente vinculadas a los componentes de *posición, escala y rotación.*
- *a-scene* - Es el objeto raíz en el que se contienen el resto de entidades, prepara una plantilla estándar con *WebVR* y *three.js*, configurando cosas como la cámara y luz por defecto.
- *a-camera* - Determina lo que verá el usuario, podremos cambiar la posición o rotación de la cámara además de poder añadir el atributo de *wasd-controls-enabled* que permitirá al usuario moverse por la escena, reposicionando la entidad referente a la cámara con las teclas w-a-s-d.
- *a-light* - Con ella podremos modificar valores como la posición, el ángulo, el color, la distancia o el tipo de luz con el que se iluminará la escena.
- *a-text* - muestra texto en plano, algo muy sencillo visto en las dos dimensiones, pero que requiere del uso de *three-bmfont-text* [23].

- *a-sky* - Añade a la escena un color o una imagen 360º previamente seleccionada por el usuario como fondo.
- *a-cursor* - Permite al usuario realizar acciones como clicar sobre objetos de la escena para demostraciones en las que no se utilice la realidad virtual.
- *a-box* - Crea un cubo con una gran cantidad de atributos predeterminados como puede ser el color, la altura, la profundidad, la opacidad, ... Todos, atributos que pueden ser ajustados a gusto del usuario.
- *a-sphere* - Al igual que a-box esta muestra una figura geométrica, en este caso una esfera. De gran utilidad para este proyecto, ya que ha sido utilizada para referirse a cada dato mostrado en los gráficos.

Como una pequeña demostración se puede ver en la siguiente ilustración una demo con un elemento a-box y otro a-sphere para comprender como funcionan esas entidades.

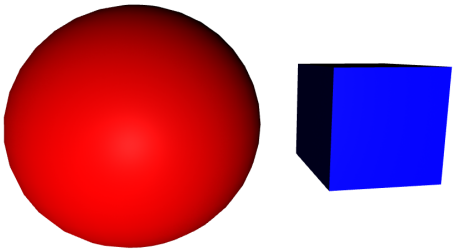

Figura 3.8: Demostración de la visualización de objetos primitivos.

Conocido el funcionamiento de *A-FRAME* el siguiente paso que se propuso fue el utilizar en la medida de lo posible los componentes creados en el proyecto de *BabiaXR* [24] con el fin de poder crear los gráficos.*BabiaXR* es un conjunto de herramientas para visualizar datos en tres dimensiones la cual ofrece un gran abanico de componentes, que pasaré a detallar a continuación, para visualizar los datos de maneras muy variadas.

Componente	***Descripción***
babia-pie	Este componente muestra un gráfico con forma circular cuyas porciones referencian los datos introducidos
babia-doughnut	Tiene un funcionamiento similar al anterior, pero en este caso con forma de un donut.
babia-bars — *babia-barsmap*	El primero muestra gráficos de barras en dos dimensiones, utilizando solo los ejes x e y, mientras que el map, hace lo mismo dando uso a las tres dimensiones.
babia-bubbles	Muestra gráficos de esferas en tres dimensiones.
babia-cyls — *babia-cylsmap*	Al igual que con los gráficos de barras, estos muestran figuras cilíndricas en dos y tres dimensiones respectivamente.
babia-city — *babia-boats*	El componente de city muestra un gráfico que representa la estructura de una ciudad, mientras que el boats cumple la misma función pero mostrando la ciudad encima de un bote.
babia-network	Este último muestra un gráfico en forma de red.

Tabla 3.1: Componentes BabiaXR.

Como la idea desde un inicio era la de mostrar la representación linear de los datos con objetos que se refieran a cada una de las muestras, el componente que más encajaba con la idea era *babia-bubbles*. Un componente creado para mostrar valores introducidos mediante un fichero ubicado en el mismo repositorio con un formato de objeto JSON o insertándolo como valor a uno de los atributos.

Este componente utiliza esferas como entidad que representa cada valor dado, ubicándolo en la escena en la posición relativa calculada mediante los datos de cada columna del fichero mencionado anteriormente, al que se debe de especificar que campos deben componer los datos de los ejes x, y, z.

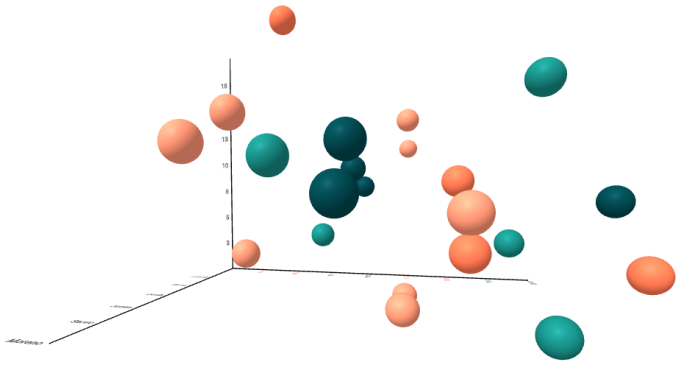

Figura 3.9: Muestra de un gráfico creado con *babia-bubbles*.

4

Descripción informática

4.1. Metodología

4.1.1. Metodologías disponibles

A continuación pasaré a explicar las metodologías más utilizadas y mejor valoradas para el desarrollo de software. Concluyendo con la metodología que ha sido seleccionada para este trabajo.

Metodologías ágiles

Actualmente, son las metodologías más utilizadas, se basan en la entrega de código de calidad en cortos plazos de tiempo, los requerimientos se definen de antemano, pero dan margen para adaptarse a lo largo del desarrollo del proyecto. Entre ellas podemos destacar:

- Lean.

  Se enfoca en la minimización de desperdicios, estableciendo las mínimas fases posibles para optimizar así la productividad y reducir costes. Se basa en *Lean Manufacturing*, un método de organización de trabajo que se centra en la mejora continua de la comunicación y el trabajo en equipo.

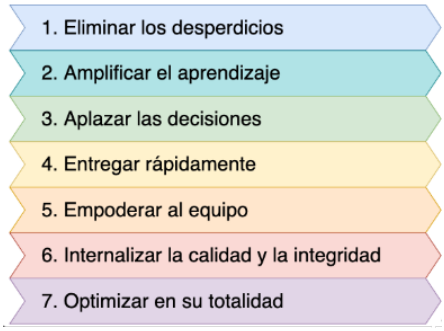

Figura 4.1: Flujo de trabajo Lean.

Está planteada de tal manera que un equipo de desarrollo pequeño pueda elaborar cualquier tarea en el menor tiempo posible.

- Programación extrema (XP).

  Esta metodología se centra en los proyectos que no tienen unos requisitos bien definidos, fomentando las buenas prácticas para que los equipos rindan mejor. El pilar fundamental es conseguir un feedback constante del cliente.

Figura 4.2: Flujo de trabajo de la programación extrema.

Para conseguir esto, se busca el diseño sencillo, una buena planificación, entregas frecuentes, programación en parejas, testing y propiedad colectiva del código. Convirtiéndose así en la metodología más flexible para adaptarse a cambios.

- Kanban.

  Consiste en dividir las tareas en unidades lo más sencillas posible, para poder organizarlas en un tablero, dividiéndolas en diferentes etapas dependiendo de lo avanzadas que se encuentre. Separándose en pendientes, en curso y finalizadas.

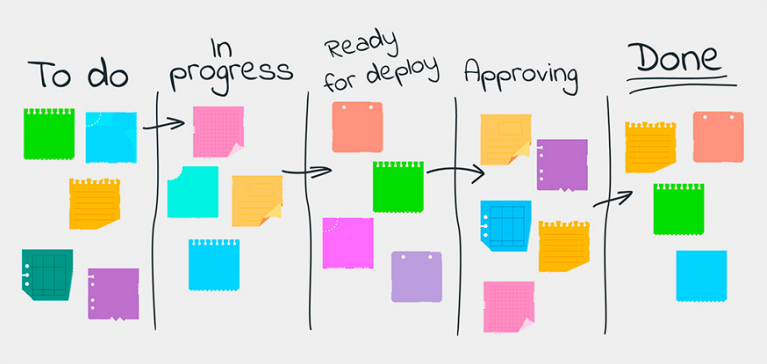

Figura 4.3: Flujo de trabajo Kanban.

De esta manera se crea un flujo de trabajo donde es posible incrementar el valor del producto dependiendo de la prioridad de las tareas. Cabe destacar que el reparto de trabajo depende de cada miembro del equipo, siendo el flujo de tareas constante, y buscando la proactividad por parte de ellos para aceptar nuevas tareas cuando terminan las anteriores.

Como curiosidad, mencionar que esta metodología fue creada por una empresa en concreto, la empresa de automóviles Toyota.

- Scrum.

  Es similar a la Kanban, utilizando también la división de requisitos y tareas. Dividiendo al equipo en grupos de trabajo autónomos.

  La característica que diferencia a esta de las anteriores es la de iterar en bloques de tiempo cortos y fijos, de entre dos a cuatro semanas, conocidos como sprints.

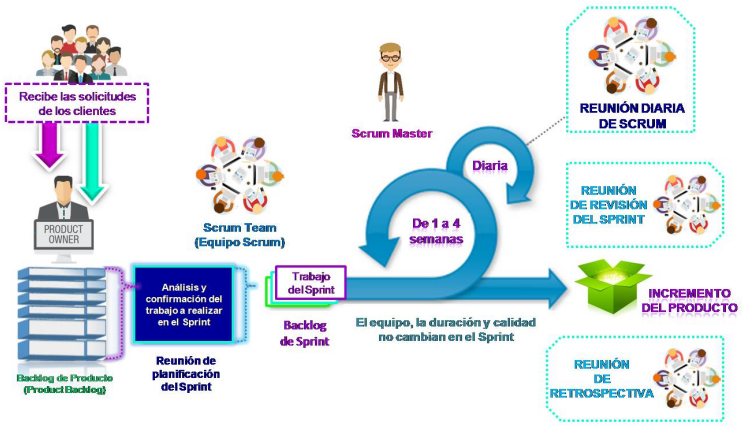

Figura 4.4: Flujo de trabajo Scrum.

Dentro de cada sprint se diferencian cuatro etapas: planificación del bloque (sprint planning), ejecución(sprint), reunión diaria (daily meeting) donde cada miembro del equipo comunica su situación actual y el progreso conseguido el día anterior, y por último, la presentación de los resultados (sprint review). Al final de cada sprint, se busca entregar un producto final, previamente acordado, al cliente.

Metodologías Tradicionales

Las metodologías tradicionales se caracterizan por presentar una rigidez total y una necesidad de definir todos los requisitos desde el inicio del proyecto, dando pie a pocos o ningún cambio y por consiguiente a una flexibilidad escasa.

- Espiral

  La metodología en espiral se basa en un enfoque iterativo e incremental para realizar el desarrollo, combinando el modelo de cascada con el desarrollo iterativo. Es el modelo más apropiado si lo que se busca es minimizar riesgos en proyectos a largo plazo y que supongan una gran inversión económica.

Figura 4.5: Flujo de trabajo en espiral.

Se divide en cuatro etapas: planificación, análisis de riesgo, desarrollo de un prototipo, junto al diseño, y valoración del cliente.

- Cascada

  Es la metodología más utilizada tradicionalmente. Sigue una serie de pasos estrictos y en orden, con una estructura lineal y unas etapas que se organizan de arriba abajo, de ahí el nombre.

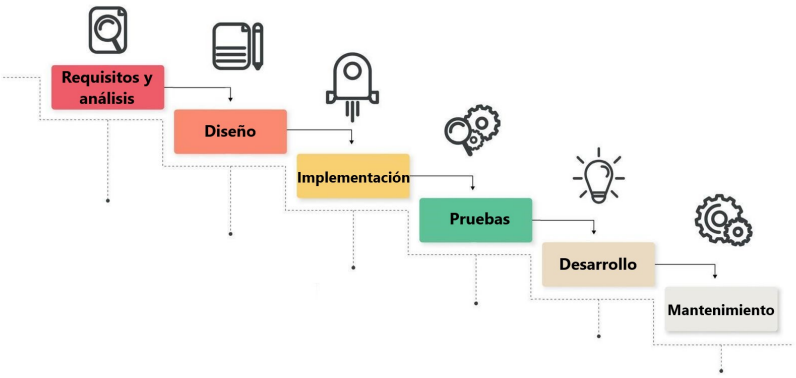

Figura 4.6: Flujo de trabajo en Cascada.

A menudo se usa en proyectos grandes con un plazo generoso. Los requisitos y especificaciones iniciales no son propensos a cambiarse. Una gran diferencia con los métodos ágiles es que debido a su estricta planificación por etapas, no serán posible presentar el producto al cliente hasta que no se haya alcanzado una etapa bastante avanzada del proyecto.

- Diseño rápido de aplicaciones

  Con esta idea, se pretende desarrollar software en un periodo de tiempo mucho menor al de otras metodologías, pero con un coste mucho más alto y necesitando un desarrollo más flexible, lo que puede provocar conflictos en la fase de evaluación por parte del cliente.

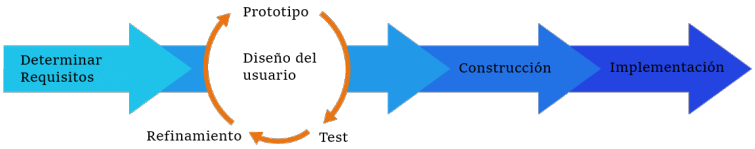

Figura 4.7: Flujo de trabajo RAD.

De esta manera, lo que te busca es poder presentar un prototipo funcional lo antes posible, sacrificando tiempo de las etapas de análisis, diseño, pruebas e implementación.

- Incremental

  Por último, la metodología incremental podría la que más se acerca a las ágiles entre las tradicionales. En cada etapa incremental del desarrollo se añaden nuevas funcionalidades, pudiendo así ver resultados mucho antes en comparación a otras metodologías tradicionales.

Gracias a esta manera de trabajar se puede comenzar a utilizar el producto antes de finalizarlo, dando facilidades también a los usuarios de proponer nuevas ideas o encontrar problemas, que pueden ser resueltos sin provocar grandes modificaciones en los progresos.

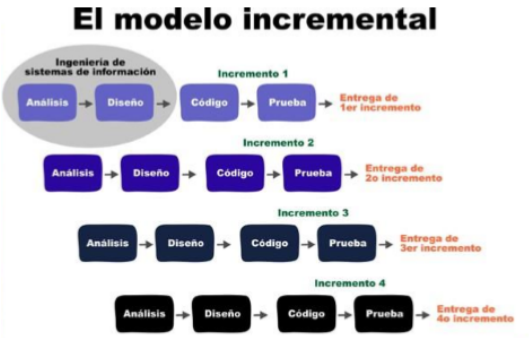

Figura 4.8: Flujo de trabajo incremental.

4.1.2. Metodología empleada

Después de haber revisado el funcionamiento de las metodologías más utilizadas en el desarrollo de software, la seleccionada fue la metodología Scrum.

En el caso particular de este Trabajo de Fin de Grado, la mayoría de perfiles que se buscan para este método de organización se han agrupado en los perfiles de los tutores y el mío. Los tutores toman el papel de clientes, que esperan el producto a final de cada sprint y mi figura ha sido la del scrum master y equipo de desarrollo.

La organización se basó en sprints de aproximadamente cuatro a seis semanas, en los que se ha seguido la estructura de Sprint Planning - Sprint - Sprint Review. Formándose el equipo de desarrollo de un solo miembro, las Daily meetings se sustituyeron por sesiones de planificación individuales en las que marcaba tareas como pendientes para cumplir ese sprint. Para agilizar más aún el proceso, tanto la Sprint Planning como la Sprint Review, se realizaban en el mismo momento, recibiendo así feedback constante de los tutores.

La decisión de elegir esta metodología se debe a varios motivos, principalmente la necesidad de optimizar el uso del tiempo lo máximo posible. De esta manera y gracias a tener un equipo de trabajo muy pequeño, ha sido posible en todo momento ajustar la duración de los sprints como fuese necesario. Otro de los motivos es el de facilitar el seguimiento del trabajo de fin de grado por parte de los tutores, ya que usando otra metodología como la de cascada, no hubiese sido posible mostrar un prototipo en una etapa alpha o beta.

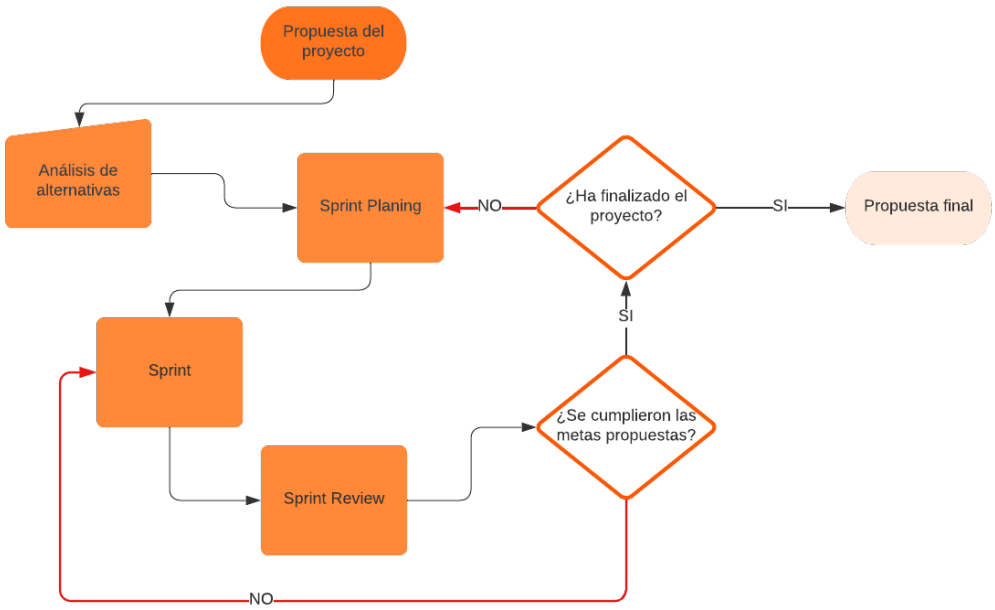

Figura 4.9: Diagrama de flujo de la metodología empleada.

4.2. Requisitos

4.2.1. Requisitos funcionales

En este apartado se pasarán a detallar los requisitos funcionales necesarios para la realización del desarrollo, lo que nos facilitará la interpretación de los resultados de los gráficos. No se tendrá en cuenta los desarrollos añadidos propuestos en el trascurso del trabajo que aportan más funcionalidad al producto final, solo los cálculos o funcionalidades extraídas de lo ya visto en experimentos en dos dimensiones.

ID	Descripción del requisito
RF-1	El sistema deberá poder calcular la matriz de datos resultante de la multiplicación de los objetos provenientes de los dos archivos iniciales.
RF-2	El sistema deberá poder realizar un tratamiento de esos datos para normalizarlos.
RF-3	El sistema deberá poder mostrar la matriz de datos resultante de estos cálculos en un gráfico.
RF-4	El sistema deberá diferenciar de manera clara los datos referentes a los ejes vectoriales de los de la muestra de datos.
RF-5	El sistema deberá permitir al usuario interactuar con los ejes vectoriales mediante el gesto de agarrar la esfera con las manos.
RF-6	El sistema deberá recalcular la muestra de datos basándonos en el reposicionamiento de los ejes vectoriales realizados por el usuario.

Tabla 4.1: Requisitos funcionales.

4.2.2. Requisitos no funcionales

Por otro lado, en ese apartado se marcarán los requisitos de implementación para mejorar el desarrollo.

ID	Descripción del requisito
RNF-1	El desarrollo de código debe tener un buen rendimiento y ser eficiente, tiendo en cuenta que se busca realizarlo para gafas de realidad virtual, con un procesamiento inferior al de un ordenador.
RNF-2	El desarrollo de código debe ser intuitivo, visual y legible, favoreciendo así posibles aportaciones de otros usuarios a los componentes propuestos.

Tabla 4.2: Requisitos no funcionales.

4.3. Arquitectura del Sistema

En este apartado procederé a explicar con el uso de varios diagramas el funcionamiento técnico del proyecto. Por un lado, se mostrará un diagrama UML de clases en el que se especificará la estructura de clases que se utilizan, añadiendo un archivo html en el que se debe incluir un componente *toprint-config* y otro *bubbles-star-coordinates*.

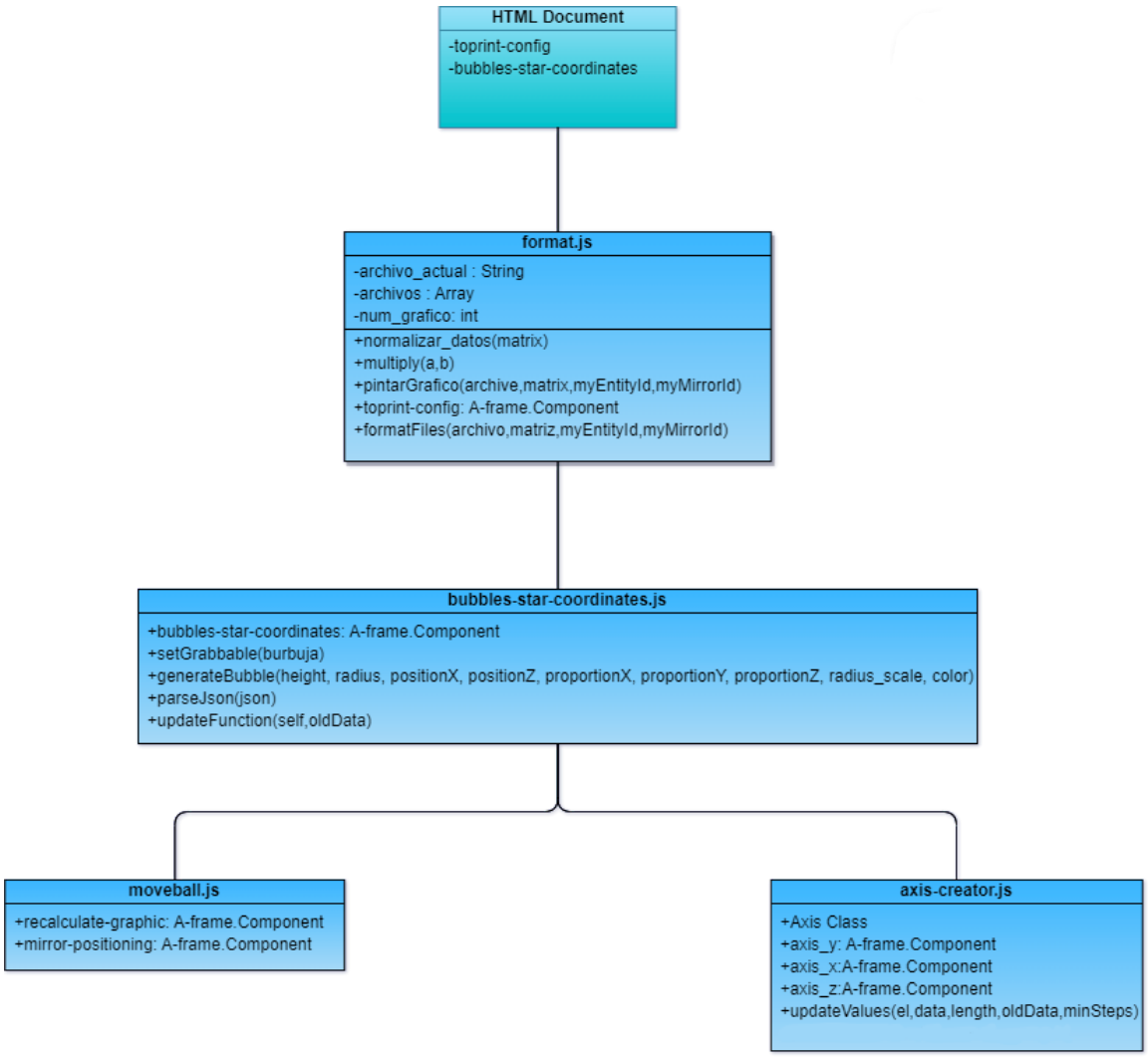

Figura 4.10: Diagrama UML de las clases.

Como diagrama complementario al mostrado en la página anterior, ahora veremos un diagrama UML que detallará los componentes que se utilizan en el proyecto y las relaciones entre sí.

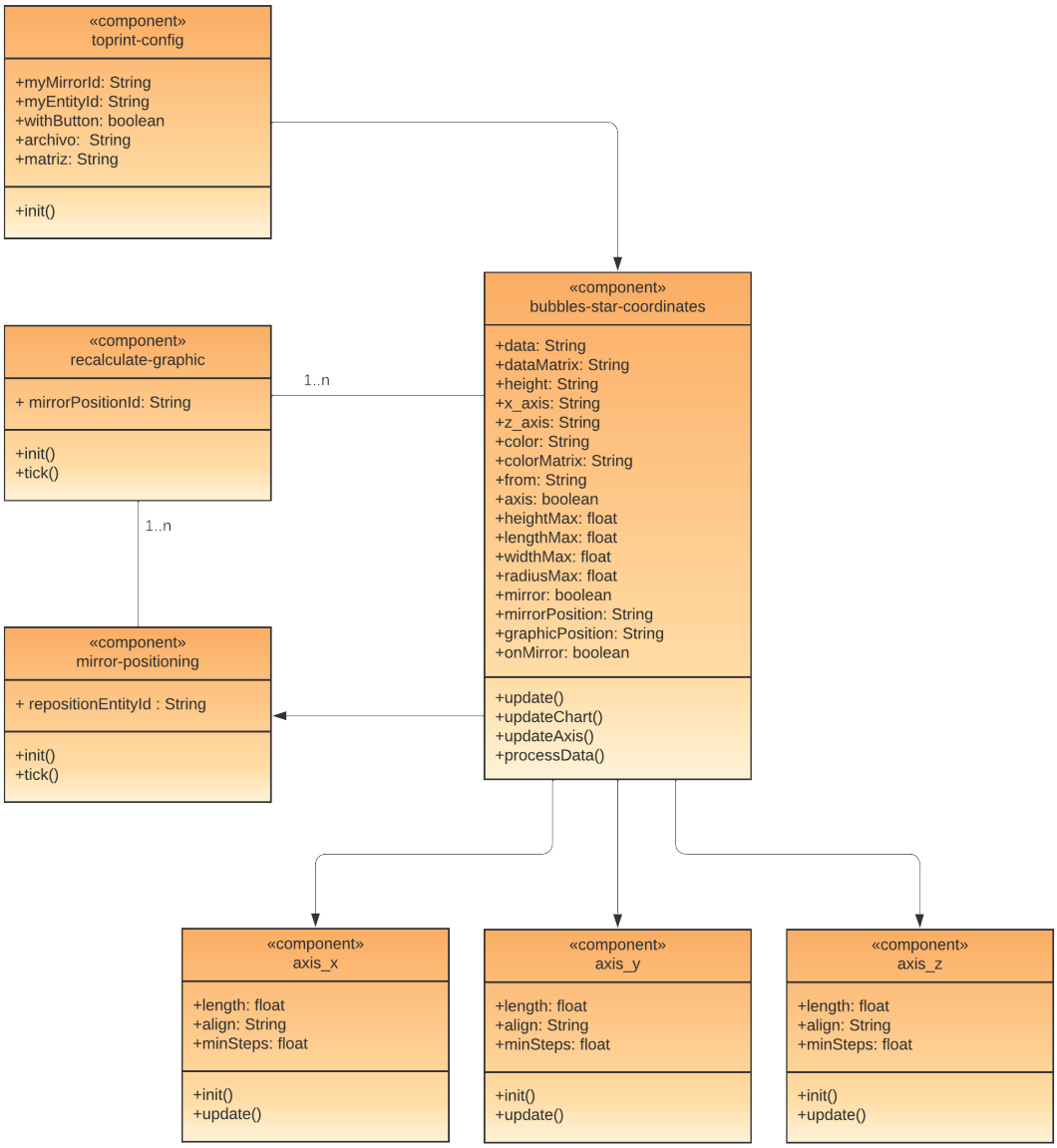

Figura 4.11: Diagrama UML de los componentes.

4.4. Implementacion

En este apartado se detallará la implementación del código utilizado con explicaciones sobre los bloques de código para poder comprender mejor lo que se irá analizando. Para hacer más sencilla la comprensión del desarrollo se utilizarán diagramas de secuencia que darán más contexto al proceso que se lleva a cabo de una manera lineal.

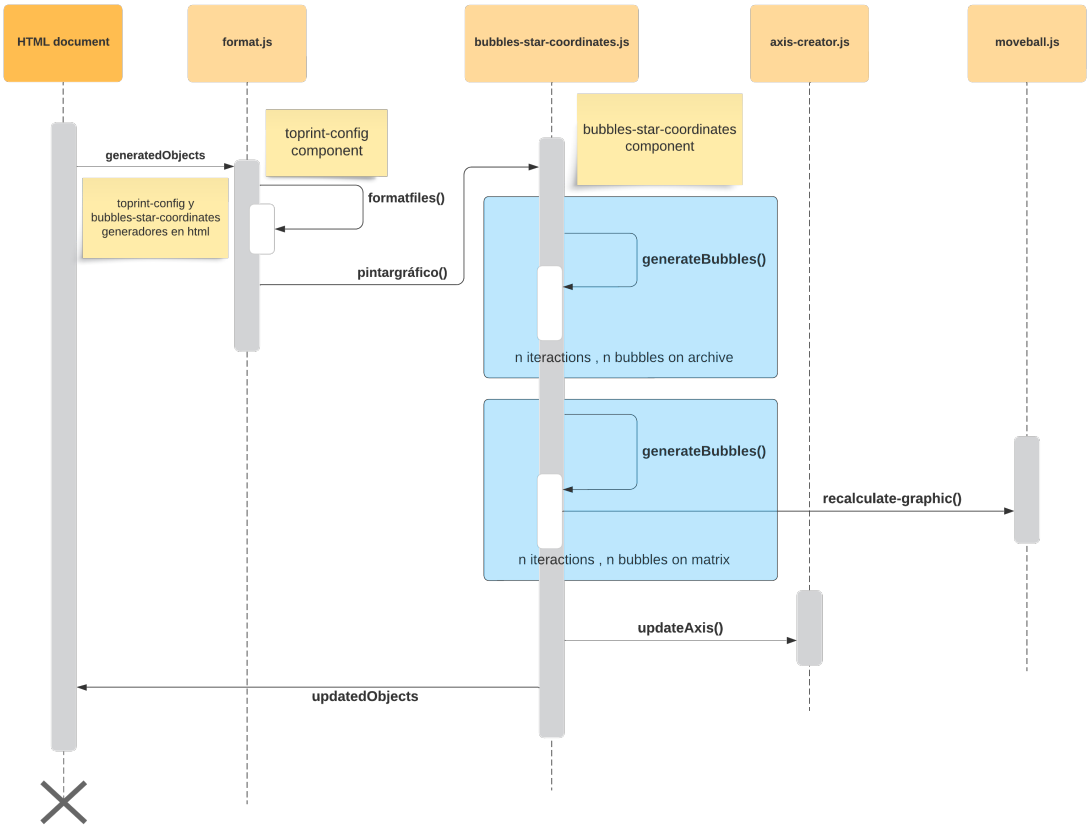

Figura 4.12: Diagrama de secuencia / sin espejo.

Se puede diferenciar una estructura dividida en clases, que se conectan entre sí con métodos. Cabe destacar que se especifica donde se ubican los dos componentes que se utilizan en el archivo HTML, con las notas amarillas que se pueden ver en las clases *format.js* y *bubbles-star-coordinates.js*. Por otro lado, los bloques marcados con un color azul de fondo se refieren a bucles de n iteraciones, donde n depende del número de datos en los archivos iniciales.

En este diagrama se define la secuencia específica en la que se muestran gráficos sin el uso de botones como selectores para cambiar uno por otro en tiempo real, ni se utiliza el componente mirror, el cual se explicará más adelante junto al siguiente diagrama de secuencia. Comienza con la generación de los dos componentes en el archivo html, *toprint-config* y *bubbles-star-coordinates*. Estos se añaden a una entidad *a-entity*, como se muestra en los siguientes sub apartados.

toprint-config

Código 4.1: Declaración de componente *toprint-config*.

```
<a-entity toprint-config="archivo: archivosjson\
    cereal_data_no_missing.json;
matriz: archivosjson\cereal_no_missing_matriz_PCA.json">
</a-entity>
```

El código del bloque 4.1 es un ejemplo de como se declararía el componente *toprint-config* dando uso a algunos de sus atributos. Para comprender bien el funcionamiento del componente y dar más contexto al bloque de código anterior, se utilizará la siguiente tabla como documentación específica de los atributos:

Nombre	Tipo de dato	Valor por defecto	Descripción
myMirrorId	String	None	Id de la entidad en la que se quiera ubicar el componente mirror de bubbles-star-coordinates
myEntityId	String	None	Id de la entidad en la que se quiere aplicar los valores de los archivos, uso necesario en demostraciones con más de un gráfico mostrado al mismo tiempo
withButton	boolean	false	booleano necesario solo en caso de querer realizar una demo con un selector de botones en el que se quiera cambiar entre varios archivos, deberá aplicarse a cada entidad asociada a un botón
archivo	String	”	url al archivo de datos
matriz	String	”	url al archivo que contiene la matriz con los ejes vectoriales

Tabla 4.3: Atributos de *toprint-config*.

Como característica especial de este componente, el atributo 'withButton' permite utilizar esto como un switch, en el que adjuntando el componente a una caja (a-box), una esfera(a-sphere) u otras entidades primitivas agarrables en la escena, se podría cambiar entre archivos, mostrándolos de uno en uno, solo los seleccionados como se puede ver en la demostración de la siguiente imagen.

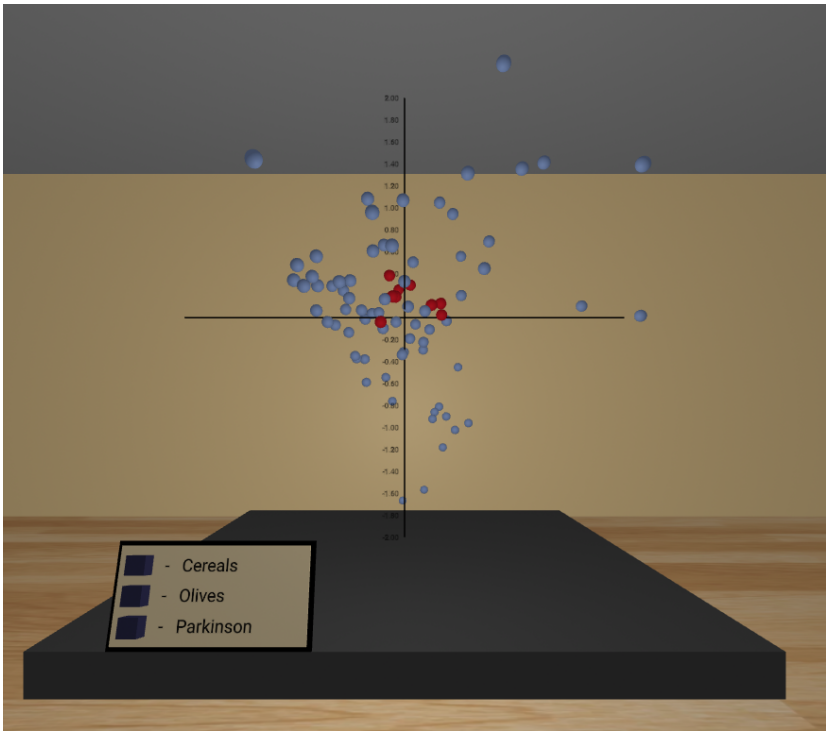

Figura 4.13: Demostración con botones como selectores.

bubbles-star-coordinates

Código 4.2: Declaración de componente *bubbles-star-coordinates*.

```
1 <a-entity bubbles-star-coordinates='axis:false; radiusMax:
    0.05;  heightMax:2; lengthMax:2; widthMax:2; data:[];
    dataMatrix:[];
2             x_axis: 0; z_axis: 2; height: 1' position="0 2.25 0
                 ">
3           </a-entity>
```

El bloque de código 4.2 muestra un ejemplo de la asignación del atributo *bubbles-star-coordinates* a una entidad, donde los campos de data y dataMatrix se colocan vacíos, ya que se asignan sus valores mediante el uso del componente previamente analizado *toprint-config*. Al igual que se hizo con este, se pasará a analizar los atributos.

Nombre	Tipo de dato	Valor por defecto	Descripción
data	String	None	array de datos producto de la multiplicación de los dos archivos
dataMatrix	String	None	array de datos que forman los ejes vectoriales
height	String	'height'	columna de datos de los arrays referentes a la altura o eje y
x_axis	String	'x_axis'	columna de datos de los arrays referentes al eje x
z_axis	String	'z_axis'	columna de datos de los arrays referentes al eje z
color	String	'#7C93C3'	color de las burbujas de data
colorMatrix	String	'#c1121f'	color de las burbujas de dataMatrix
axis	boolean	true	booleano por referir a si se quiere mostrar los ejes o no
heightMax	integer	10	altura máxima (y mínima, usando el valor en negativo) que puede alcanzar una esfera
lengthMax	integer	10	longitud máxima (y mínima, usando el valor en negativo) que puede alcanzar una esfera
widthMax	integer	10	profundidad máxima (y mínima, usando el valor en negativo) que puede alcanzar una esfera
radiusMax	integer	None	radio máximo que puede tener una esfera
mirror	boolean	false	opción de elegir si se genera o no el component mirror
mirrorPosition	String	None	en caso de querer ubicar el mirror en un lugar específico y no el predeterminado, se añade la id de la entidad donde está ubicada, no es necesario que se añada por el usuario en el documento html, viene dado por el myMirrorId de toprint-config

graphicPosition	String	None	al igual que en el caso anterior, se necesita la id del gráfico al que hace referencia ese mirror, viene dado por el myEntityId de toprint-config
onMirror	boolean	false	un atributo interno del componente que se utiliza para ubicarse dentro del proceso de creación del mirror, se explicará detalladamente más adelante

Tabla 4.4: Atributos de *babia-star-coordinates*.

Los cinco primeros atributos podrían definirse como los principales, todos los demás pueden ser declararos con los valores por defecto, o sin necesidad de declararlos, pero estos cinco necesitan tener valores para poder mostrar las representaciones de datos. Por un lado, tanto data como dataMatrix se rellenan con los archivos que anteriormente se refirieron en los atributos de toprint-config, lo que significa que no tendría que añadirlo el usuario en el documento HTML.

Los colores de los dos tipos de datos y las alturas máximas son métodos de personalizar nuestro gráfico con el tamaño que se necesite para ocupar el espacio del que se dispone o el color que se desea. El único de estos campos de personalización que no requiere de ningún valor ni tiene uno predeterminado es radiusMax, ya que en caso de no tener ninguno, cada esfera tendrá un radio predeterminado de por sí, ya que a-sphere mantiene sus atributos primitivos con los que se aplica '1' como valor en el radio.

Los últimos cuatro atributos se explicarán más detalladamente junto al siguiente diagrama de secuencia en el que se hará uso del mirror.

Para continuar con el flujo del diagrama de secuencia de la figura 4.12, el siguiente paso viene en el archivo format.js, donde se ubica el componente toprint-config y donde se realizan varias acciones, primero de todo, se leen los valores añadidos en los componentes del documento HTML y se llama al método *formatfiles()* para transformar los archivos en objetos que posteriormente se puedan multiplicar y manipular.

Después de llamar al método de formarfiles, comienza el proceso de *pintargrafico()*, donde se multiplican los archivos, se normalizan los datos y añaden los valores conseguidos en los atributos de data y dataMatrix, lo que provoca una reacción en cadena donde se actualiza el objeto referente al gráfico. Además de eso, previamente se ha realizado un análisis sobre el HTML para buscar las entidades con bubbles-star-coordinates y así guardar los archivos iniciales de dos maneras diferentes, dependiendo de la cantidad de gráficos que haya en la escena al mismo tiempo, para modificaciones posteriores.

El siguiente paso viene en el script bubbles-star-coordinates.js, que tiene como bloque principal, el componente con su mismo nombre. Al modificar previamente los datos del atributo se lanza la función *update()* que se ejecuta cada vez que se actualiza un atributo del componente.

Esta función actúa como un generador completo del gráfico, en primer lugar, recoge los datos de los arrays data y dataMatrix y calcula los valores máximos de cada dimensión para conseguir las proporciones con las que se muestra el gráfico. Esto tiene un valor muy importante porque sirve para recalcular las posiciones y dar valores a los arrays con los datos originales, reutilizando la proporción de cada eje a la inversa previamente.

Posteriormente, comienzan los bucles representados en color azul en el diagrama, donde se generan las burbujas. En este caso se recorren dos bucles, para crear las burbujas de la representación de datos y, por otro lado, los ejes vectoriales, con este bucle en específico se añade un nuevo componente, *recalculate-graphic*.

Código 4.3: Método para la generación de las burbujas.

```
function generateBubble( height, radius, positionX, positionZ,
    proportionX, proportionY, proportionZ, radius_scale, color)
    {
     if (proportionY) {
         height = proportionY * height
     }
     if (radius_scale) {
         radius = radius_scale * radius
     }
     let entity = document.createElement('a-sphere');
     entity.setAttribute('radius', radius);
     entity.setAttribute('color',color);
     entity.setAttribute('proportionX',proportionX);
     entity.setAttribute('proportionY',proportionY);
     entity.setAttribute('proportionZ',proportionZ);
     entity.setAttribute('position', { x: positionX, y: height,
        z: positionZ });

     return entity;
}
```

El último paso en la creación del gráfico sería generar los ejes, que se realiza utilizando los valores máximos de cada dimensión. La clase que contiene los componentes y métodos es axis-creator.js.

Por último, con las esferas que representan los ejes vectoriales se utiliza el componente *recalculate-graphic*, que funciona en cada tick del procesamiento, donde se comprueba que si una burbuja ha sido agarrada con el componente *super-hands-component* [25] y posteriormente soltada, se recoge la nueva posición y utilizando los valores de las proporciones se modifica la matriz de lo ejes vectoriales, y se vuelve a llamar al método de *pintargrafico()* para mostrar los nuevos datos.

Para concluir con la implementación del código se utilizará este nuevo diagrama de secuencia donde se hará uso de los botones como selectores y la creación del mirror.

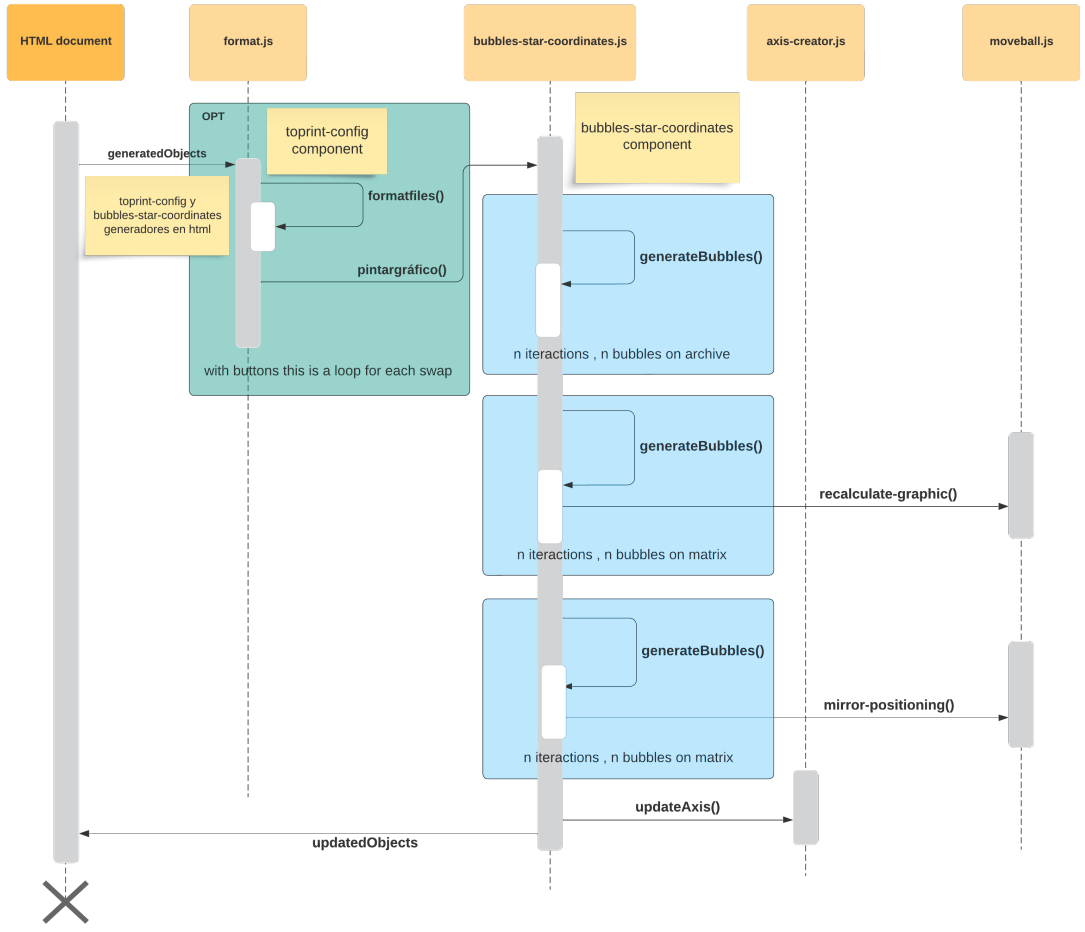

Figura 4.14: Diagrama de secuencia / espejo + botones.

Ya que el funcionamiento de la mayoría de los componentes se ha explicado, en este apartado solo será necesario centrarse en el uso de los selectores y sobre todo, en el objeto mirror.

Al dar uso del componente *toprint-config*, en el documento HTML, en este caso se colocarán las entidades primitivas, por ejemplo, a-box, al llegar al script format.js, en la función init() del componente, se mantendrá la llamada al método *pintargrafico()* dentro de un evento onClick para que su ejecución sea condicional y esté a la espera de pulsar uno de los botones o selectores.

De esta manera se puede utilizar el mismo componente para los selectores y para el ejemplo del otro diagrama con este condicional:

Código 4.4: Condicional para la manera de crear el gráfico.

```
if(withButton){
    this.el.addEventListener('click',function(){
        formatfiles(archivo,matriz,myEntityId,myMirrorId);
    })
}else{
    formatfiles(archivo,matriz,myEntityId,myMirrorId);
}
```

Y por último, el objeto mirror que varias veces ha sido mencionado. Este componente está asociado al gráfico que se esté mostrando en la escena, se enlaza a él como un 'hijo' de la entidad generadora de la representación de datos.

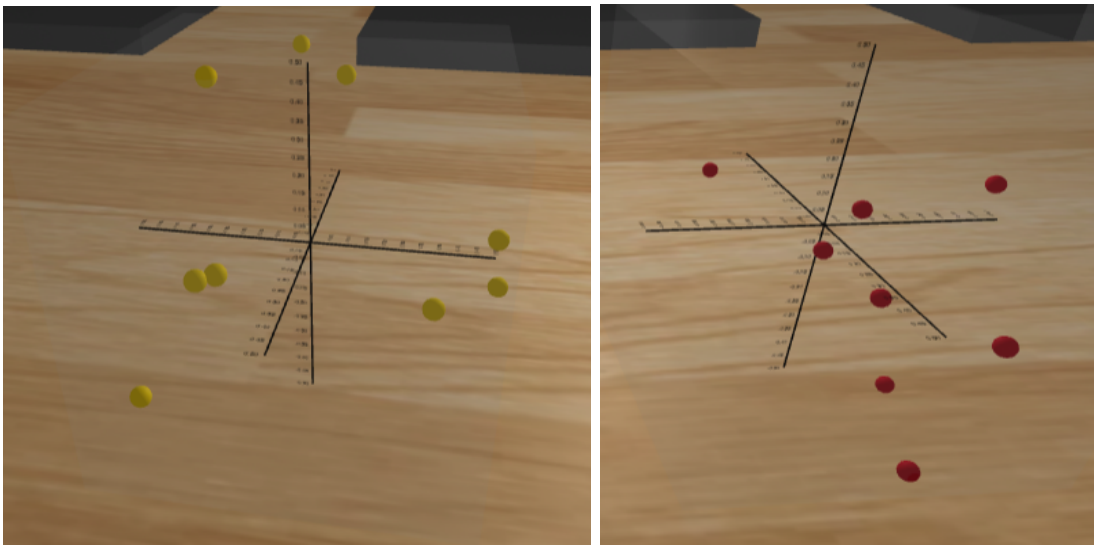

Figura 4.15: Ejemplo de mirrors en escena.

Este gráfico con un cuarto de las dimensiones del gráfico principal es una representación en escala de los ejes vectoriales. Estas burbujas tienen la función grabbable de super-hands al igual que las esferas que representan los ejes vectoriales del gráfico principal, pero como su nombre indica, funcionan como un espejo.

Al mover una de estas esferas que pueden estar más cerca del usuario en la escena, la esfera a la que hace referencia de su padre, se moverá de la misma manera, lo que hará que al soltar la esfera, y al haberse movido la del padre, se recalcularán los datos de la misma manera de la que se hace al agarrar una burbuja del gráfico principal.

super-hands-component

Se ha mencionado varias veces el componente super-hands en este capítulo, así que para dar más contexto de lo que se está mencionando y con la intención de aprender sobre esta herramienta, analizaré el funcionamiento de dicho componente.

Para facilitar las demostraciones y mejorar inmensamente la usabilidad de las mismas, el componente super-hands pone a disposición del programador que desee usarlo una lista de componentes reactivos con los que es posible clicar, arrastrar y soltar, dejar caer, agarrar, dar flotabilidad o hacer estirables las entidades como cajas o esferas.

Los componentes que pone a disposición este proyecto son:

- clickable: permite que los objetos sean clicados con las manos al agarrarlos, pero sin moverlos.
- draggable: convierte a una entidad en arrastrable para después soltarla en una entidad droppable.
- droppable: convierte a una entidad en un lugar para soltar entidades con el componente draggable.
- grabbable: permite que una entidad sea agarrable con lo que se puede modificar su posición con el gesto mientras se presiona un gatillo del mando.
- stretchable: permite modificar la escala de una entidad realizando un gesto con las dos manos a la vez.

4.5. Dificultades encontradas

Unificar dos representaciones de datos

La primera gran dificultad que se afrontó en el proyecto fue la de unificar el gráfico con la representación de datos, las esferas azules en la figura 5.2, con los ejes vectoriales , las esferas rojas.

El primer enfoque que se llevó a cabo fue el de generar dos entidades bubbles-star-coordinates con sus 'data' correspondientes, uno para las azules y otro para las rojas, en el que se diferenciara el funcionamiento de los ejes vectoriales con algún atributo booleano que marcara que esa entidad se refería a los ejes. Eso daría paso a otro conflicto, las proporciones, que analizaré más adelante.

Ese primer enfoque generaba problemas que eran difícilmente solubles con poco esfuerzo, por lo que se buscó un segundo y definitivo enfoque en el que la separación entre unas burbujas y otras se hacía con dos atributos diferentes en el mismo componente bubbles-star-coordinates, data y dataMatrix, de esta manera también era más sencillo trabajar con las proporciones, y diferenciar con condicionales unas esferas de otras.

Proporciones unificadas

El segundo problema a resolver venía dado por las proporciones que se utilizaban para representar los datos, de manera que fuese sencilla la representación gráfica, la modificación de los datos de los archivos originales para ser recalculadas las posiciones y, sobre todo, para que las posiciones de cada esfera tuviesen coherencia con la posición que tendrían si no fuese necesaria una redimensionalización.

Para conseguir esto, se crearon tres atributos que se agregan a cada a-sphere con la proporción de cada eje, esa proporción viene nada por el valor máximo que se le dio a ese eje al declarar el componente en el documento HTML y el valor máximo de las esferas en cada eje, de esa manera se realiza una división entre esos valores, lo que proporciona un coeficiente de proporción que se aplica a todos los valores en ese eje.

Los máximos utilizados son valores absolutos, por lo que no sobresaldrá una esfera en los límites negativos, si el valor máximo de una esfera es superior en un valor negativo que en uno positivo.

Mirror

La tercera dificultad y más grande ha sido la creación y optimización del espejo.

En primer lugar, el replicar el funcionamiento y posición de los ejes vectoriales, pero con características diferentes, ha significado un reto interesante. Lo primero que se abordó fue el añadir un condicional que con el uso de un atributo booleano que avisa del uso del mirror o no, se realiza una llamada recursiva al método *update()* del gráfico y se genera una entidad hija de la misma entidad que contiene al padre, para añadir el espejo.

Después de conseguir modificarlo para que funcione como espejo de los ejes vectoriales originales, otro reto fue la posición y tamaño del mismo, que se solucionó estableciéndolo como un valor estático, del cuarto del tamaño original del padre. La posición se ajustó mediante prueba y error hasta encontrar una ubicación en la parte frontal derecha del padre, que fuese reajustable con relación al tamaño.

Por último, como se añadió la opción de ubicar el mirror donde el usuario quería, se tuvo que modificar la manera de relacionar el mirror con su padre, ya que no se podía suponer que fuesen hijos del mismo padre y por consiguiente estuviesen relacionados de esa manera, por eso se tuvo que realizar el uso de Ids para las entidades.

5

Evaluación y pruebas

5.1. Evaluación

En la evaluación de este Trabajo de Fin de Grado, se han aplicado los principios del modelo de calidad ISO/IEC 25010. Este marco define las características de calidad que se considerarán al evaluar las propiedades de un producto de software específico. La calidad del producto de software se entiende como el nivel en el cual cumple con los requisitos de los usuarios, proporcionando así un valor agregado.

Fiabilidad

El proyecto ha demostrado ser confiable, ya que se ejecuta sin errores significativos. Durante las pruebas con familiares, no se reportaron fallos o comportamientos inesperados, en las fases finales del proyecto. La estabilidad del software ha contribuido positivamente a la experiencia de los usuarios.

Usabilidad

La usabilidad del software es aceptable, siendo relativamente fácil de usar. Sin embargo, podría haber oportunidades para mejorar la intuición de ciertas interacciones en el entorno de realidad virtual (VR). El feedback positivo de los familiares sugiere que, en general, el software es accesible.

Rendimiento

Se han identificado ciertos problemas de rendimiento en el proyecto, especialmente relacionados con las limitaciones de procesamiento de las gafas VR. La ejecución puede ser lenta en dispositivos con hardware más antiguo.

La evaluación se ha visto limitada en términos de pruebas debido a restricciones de hardware, ya que se contó únicamente con unas gafas VR Meta Quest 2. En consecuencia, no se tiene certeza de cómo el rendimiento del software podría verse afectado en dispositivos VR más antiguos, lo cual constituye una debilidad en la evaluación.

Mantenibilidad

La estructura modular de A-Frame facilita la mantenibilidad del proyecto. Sin embargo, se debe prestar atención a la documentación y a la claridad del código para asegurar que futuras mejoras y modificaciones sean manejables y comprensibles.

Es importante tener en cuenta que el proyecto está abierto a modificaciones por parte de cualquier usuario, lo que podría resultar en un posible deterioro de la calidad del código en el futuro. Sin embargo, siempre y cuando estas modificaciones se mantengan en proyectos personales, no debería haber inconvenientes.

Seguridad

En este aspecto, no se ha requerido de seguridad muy avanzada, ya que no contiene información personal, ni ningún tipo de base de datos con información sensible. Se da uso de la seguridad proporcionada por las github pages.

Escalabilidad

Aunque el software ha demostrado manejar una carga de trabajo típica, las limitaciones de procesamiento de las gafas VR pueden afectar la escalabilidad.

Portabilidad

A-Frame, por naturaleza, ofrece una buena portabilidad, permitiendo que el software se ejecute en diferentes plataformas de realidad virtual. Sin embargo, es esencial considerar las especificaciones de hardware específicas para este tipo de proyectos, aunque se pueda utilizar en ordenador y dispositivo móvil, las gafas de realidad virtual son los únicos dispositivos que ofrecen una experiencia completa.

5.2. Pruebas

Para la realización de las pruebas se crearon seis versiones diferentes de demostraciones en las que se probaron ambos componentes de diferentes maneras.

Todas las pruebas que se van a definir a continuación se llevaron a cabo con el mismo grupo de personas, conformado por 5 miembros, 3 de ellos usuarios de una edad superior a 50 años, con un conocimiento nulo de la realidad virtual o el uso de las gafas VR, ninguno de ellos con conocimientos superiores al nivel usuario de informática y con solo una pequeña explicación previa del funcionamiento de los mandos y el posicionamiento dentro de la escena.

Por otro lado, los otros dos usuarios, uno de ellos de más de cuarenta años y el otro no más de veinte, tenían conocimientos previos del funcionamiento de las gafas VR, teniendo el más joven experiencias previas en el uso de las mismas, en ámbitos de juegos recreativos. Para estos dos se utilizó una técnica diferente en la que no se explicó nada del funcionamiento de la escena, lo que permitió conocer de mejor manera si era intuitivo para alguien con experiencia.

No se presentaron problemas causados por el movimiento dentro de la escena que pudieran provocar mareos. Sin embargo, algunos usuarios, especialmente aquellos sin experiencia previa, necesitaron un breve período de adaptación.

Demo básica

En la primera de las demos se realizó una prueba básica donde se colocó un componente toprint-config con las urls a los dos archivos utilizados y un componente bubbles-star-coordinates en los que reflejar la representación de datos producto de la multiplicación matricial y los ejes vectoriales.

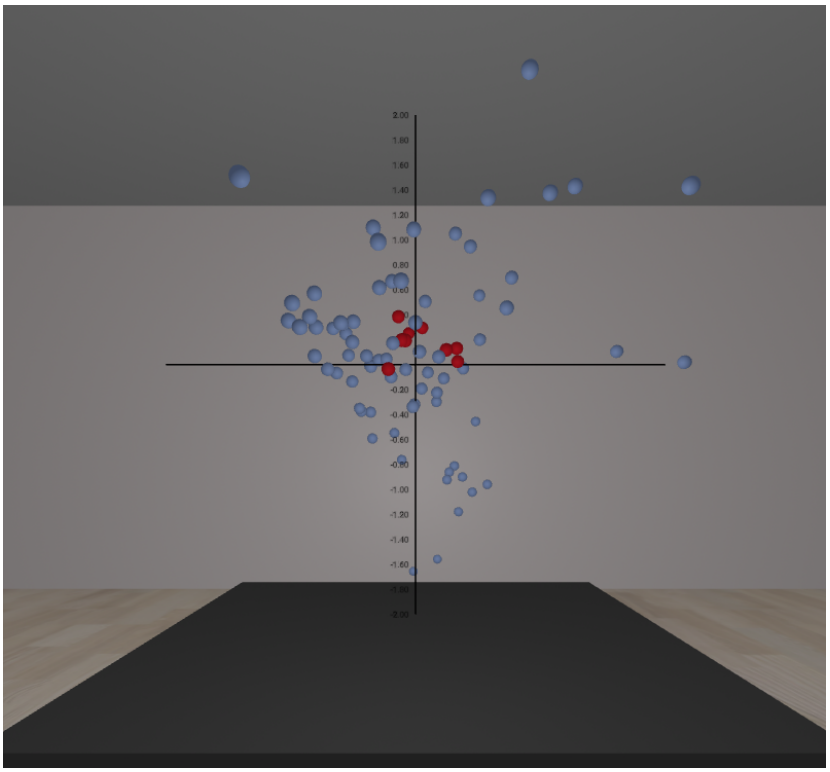

Figura 5.1: Demostración básica.

En esta demo se puede ver el gráfico con las esferas azules que representan al atributo data y las rojas que representan al dataMatrix, esas esferas rojas, que representan los ejes vectoriales, son las que pueden ser agarradas con el componente grabbable de super-hands y que tienen la función de recalcular los valores.

Demo con varios gráficos

Como se puede ver en la imagen, el funcionamiento de esta prueba es muy similar a la anterior, pero con la característica especial del uso de varios gráficos al mismo tiempo. Esto implica que se debe tener en cuenta cada uno de los gráficos por separado para que se recalcule solo el gráfico del que pertenece la esfera movida, no se modifiquen los otros, los archivos utilizados sean siempre los mismos, y no se solapen unos con otros. También se ha probado el cambiar los colores de los gráficos para poder diferenciarlos claramente.

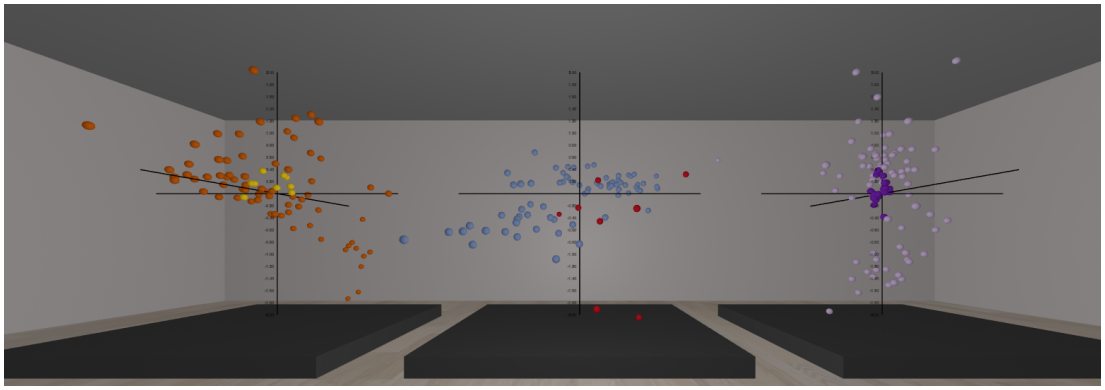

Figura 5.2: Demostración con varios gráficos.

Demo con botones

En esta demostración se habla de la figura 4.13, una demo básica al igual que la figura 5.2, pero con la singularidad del uso de botones como selectores, al igual que se requiere de tener en cuenta todos los archivos y gráficos para la hora de seleccionar y mover esferas, en este es importante tener guardado el archivo del gráfico actual de la escena.

Con la prueba que se realizó se utilizaron tres bloques de datos, pero no hay un límite necesario de gráficos, siempre y cuando en la escena se represente de una manera eficiente y accesible.

Demo con botones y mirror

Al igual que en la demo anterior, se utiliza el mismo display de botones como selector de gráficos, pero en este caso se prueba por primera vez el componente mirror. Este debe tener en cuenta en cada momento el gráfico que se representa en escena.

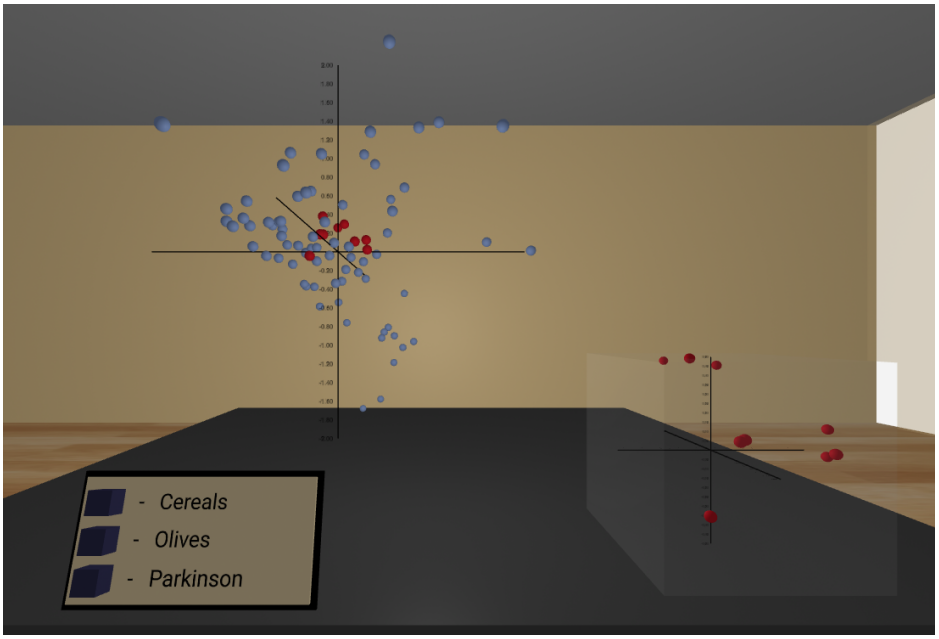

Figura 5.3: Demostración con selectores y mirror.

Demo con varios gráficos y mirror

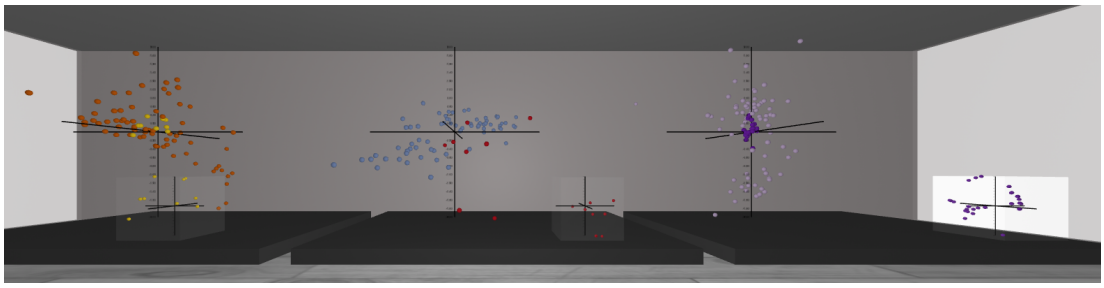

Figura 5.4: Demostración con varios gráficos y espejo.

En este caso solo se realiza una prueba similar a la de la demo de la figura 5.2, con la característica especial del uso de mirror para cada gráfico.

Demo con varios gráficos y espejo reubicado

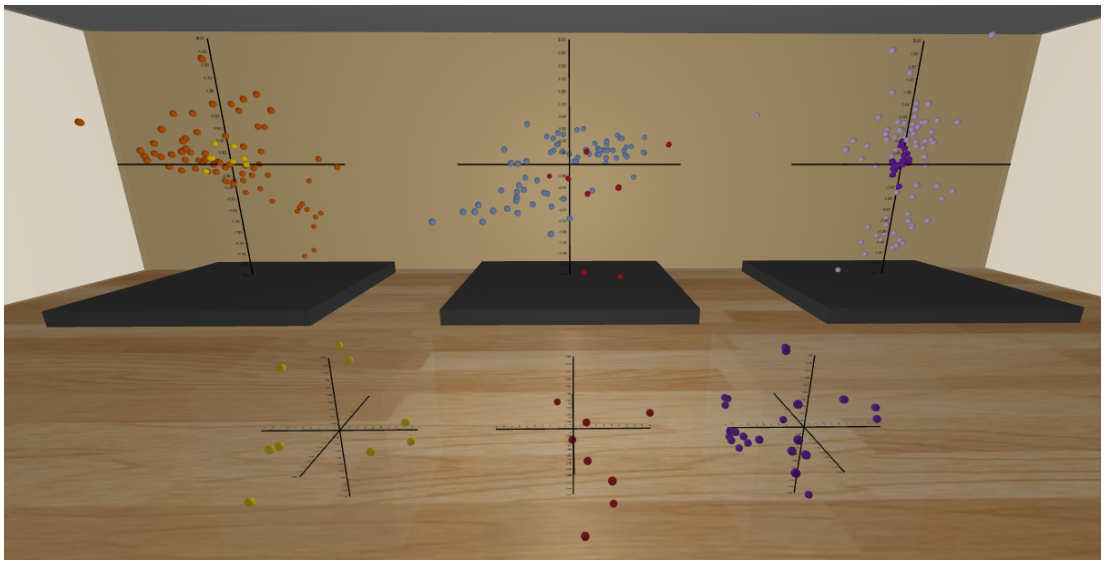

Figura 5.5: Demostración con varios gráficos y espejo modificando ubicación.

Esta es la última prueba realizada como demostración del funcionamiento del proyecto. En este caso, al igual que el anterior, se utilizan varios gráficos de manera simultánea en la escena, pero con los mirror ubicados a petición del usuario, por lo que es necesario añadir más contexto al componente toprint-config sobre cuál es la posición de la entidad a la que se quiere agarrar al mirror y cuál es el gráfico al que representa el mirror.

6

Conclusiones y trabajos futuros

6.1. Conclusiones

Este trabajo de fin de grado concluye cumpliendo los objetivos marcados desde el principio. Tanto la propuesta inicial de poder crear una escena en realidad virtual para la visualización y manipulación de datos, como las propuestas hechas conforme avanzaba el desarrollo, como es la de crear el componente mirror que nos permite manipular los datos desde una posición más cómoda, han sido completados y evaluados mediante las pruebas previamente analizadas.

Se han superado todas las complicaciones encontradas durante el desarrollo. Al comienzo, se planteó utilizar como soporte el componente de BabiaXR explicado en la sección Estudio de alternativas, pero se concluyó que sería mucho mejor crear un componente propio, al que se le pudiese modificar el comportamiento en cualquier momento, siendo esto un nuevo objetivo.

Más adelante, con el uso de las gafas de realidad virtual, se encontraron problemas para poder utilizarlas con una ejecución en local sin el uso de GitHub Pages, ya que necesitan un protocolo de seguridad con conexión HTTPS, para ello se tuvo que configurar un certificado SSL propio.

En resumen, este proyecto no solo ha contribuido a mi propio crecimiento académico, sino que también puede tener un impacto más amplio al abrir nuevas perspectivas en el ámbito de la visualización de datos y la realidad virtual.

6.2. Trabajos futuros

Aunque han sido cumplidos estos objetivos, aún quedan pendientes algunas otras propuestas de mejora que pueden añadir valor al proyecto. Por ejemplo, sería posible valorar los datos dando mayor o menor importancia a los mismos utilizando un gradiente de color, lo que requiere de modificaciones extensas en el código para añadir también otro componente que permita elegir cuál de las variables es la que marcará el gradiente.

El campo de la realidad virtual está en constante evolución, y la contribución a través de este trabajo de fin de grado podría tener aplicaciones prácticas y significativas en el ámbito matemático. La capacidad de visualizar representaciones de datos de manera inmersiva podría tener un impacto positivo en la comprensión y el análisis de conceptos como el álgebra lineal o la geometría.

Las posibilidades para trabajos y mejoras futuras son muy amplias. Es posible añadir nuevas funcionalidades al código sin modificar la estructura del mismo, con conceptos como:

Seguimiento del gráfico con la posición del usuario

Una posible mejora para optimizar la usabilidad del usuario sería implementar un seguimiento dinámico de la posición del gráfico en función de la dirección hacia la cual el usuario esté mirando mientras utiliza el espejo. Esto garantizaría que el espejo siempre esté alineado frente al gráfico, sin importar los movimientos del usuario alrededor de la representación de datos.

Sin embargo, es importante destacar que esta mejora podría presentar algunas limitaciones. En primer lugar, la opción de mostrar varios gráficos simultáneamente en la escena podría ser restringida, ya que se requeriría un único gráfico que pueda ajustarse y seguir la vista del usuario de manera coherente.

Además, modificar continuamente la posición y rotación del gráfico podría generar complicaciones al intentar interactuar con las esferas, especialmente cuando se intenta mover una de ellas al agarrarla sobre el espejo. Esta situación podría afectar la experiencia del usuario y la precisión en la manipulación de elementos en la representación de datos.

Es crucial considerar también que la posición del gráfico podría resultar molesta para el usuario, dependiendo de su ubicación y la perspectiva desde la cual desee visualizarlo. Esta limitación podría dificultar la capacidad del usuario para rodear y analizar detenidamente los resultados obtenidos, ya que estaría restringido a una única perspectiva.

Es decir, mientras que la implementación de un seguimiento dinámico del gráfico puede mejorar la alineación con el espejo, es esencial abordar cuidadosamente las posibles limitaciones, como la interacción con elementos específicos y las restricciones en la visualización desde diferentes perspectivas, para garantizar una experiencia de usuario fluida y efectiva.

Conferencias con gafas VR

Uno de los proyectos futuros con mayor potencial reside en la implementación de una opción que permita, a través de realidad aumentada, analizar representaciones de datos de manera colaborativa entre varios usuarios en un entorno compartido. Imagina la posibilidad de llevar a cabo reuniones virtuales donde los participantes, equipados con gafas de realidad virtual, puedan interactuar y discutir sobre visualizaciones de datos en tiempo real.

Esta innovadora propuesta tiene el propósito de simplificar el análisis de datos al ofrecer múltiples perspectivas que pueden interactuar simultáneamente con la misma representación. Esto no solo agilizaría el proceso de toma de decisiones, sino que también eliminaría la necesidad de organizar reuniones presenciales, que conllevan una mayor planificación y complejidad logística al coordinar calendarios y ubicaciones.

Al aprovechar las ventajas de la realidad aumentada y la realidad virtual, este proyecto podría revolucionar la forma en que los equipos colaboran y comparten información, superando las barreras de la distancia física. Además, la capacidad de analizar datos de manera conjunta en un entorno virtual podría conducir a descubrimientos más profundos y a una toma de decisiones más informada.

En resumen, la integración de la realidad aumentada en reuniones virtuales con gafas de realidad virtual abriría un nuevo horizonte para la colaboración y el análisis de datos, ofreciendo eficiencia y flexibilidad sin comprometer la calidad de la interacción entre los usuarios. Este proyecto tiene el potencial de transformar la manera en que los equipos trabajan juntos, haciendo avanzar significativamente la experiencia de trabajo en entornos virtuales colaborativos.

Bibliografía

[1] M. Rubio-Sánchez, A. Sanchez, and D. J. Lehmann, "Adaptable radial axes plots for improved multivariate data visualization," *Eurographics Conference on Visualization (EuroVis) 2017*, 2017.

[2] R. Cabello. (2010) Three.js. [Online]. Available: https://en.wikipedia.org/wiki/Three.js

[3] S. ang Google and W. comunnity. (2015) A-frame. [Online]. Available: https://es.wikipedia.org/wiki/A-Frame

[4] Y. Holtz. (2018) Scatterplot matrix with the native plot() function. [Online]. Available: https://r-graph-gallery.com/98-basic-scatterplot-matrix.html

[5] J. Chill. (2014) Common charts and graphs. [Online]. Available: https://justinchill.com/charts.html

[6] N. Gogia. (2020) Data analyst interview questions(part-1). [Online]. Available: https://medium.com/@nishesh.kumar/data-science-interview-questions-on-dimensional-reduction-fb8df2b239dd

[7] figshare. Mca dge data. [Online]. Available: https://figshare.com/articles/dataset/MCA_DGE_Data/5435866

[8] R. Satija, S. Lab, and Collaborators. (2019) Guided clustering of the microwell-seq mouse cell atlas. [Online]. Available: https://satijalab.org/seurat/archive/v3.0/mca

[9] Y. Wang, J. Li, F. Nie, H. Theisel, M. Gong, and D. J. Lehmann, "Linear discriminative star coordinates for exploring class and cluster separation of high dimensional data," *Computer Graphics Forum (Volume 36, Number 3)*, 2017.

[10] E. Kandogan, "Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions," *In Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics*, 2000.

[11] S. Team. (2015) A-frame documentation. [Online]. Available: https://aframe.io/docs/1.5.0/introduction/

[12] R. Cabello. (2010) three.js documentation. [Online]. Available: https://threejs.org

[13] J. Steele and N. Iliinsky, Eds., *Beautiful Visualization: Looking at Data through the Eyes of Experts*. O'Reilly, 2010.

[14] J. Hartigan, "Printer graphics for clustering," *Journal of Statistical Computation and Simulation*, 1975.

[15] A. Inselberg, "Don't panic ... just do it in parallel!" *Computational Statistics 14*, 01 1999.

[16] G. H. Laurens van der Maaten, "Visualizing data using t-sne," *Journal of Machine Learning Research 9*, 11 2008.

[17] J. M. Leland McInnes, John Healy, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv:1802.03426v3 [stat.ML]*, 09 2020.

[18] Meta. (2020) Meta quest 2. [Online]. Available: https://www.meta.com/es/quest/products/quest-2/

[19] M. Company. (2004) Meta. [Online]. Available: https://about.meta.com/es/

[20] N. Communications. (1995) Javascript. [Online]. Available: https://es.wikipedia.org/wiki/JavaScript

[21] W. W. W. Consortium. (1993) Html. [Online]. Available: https://es.wikipedia.org/wiki/HTML

[22] C. W. Group. (1996) Css. [Online]. Available: https://es.wikipedia.org/wiki/CSS

[23] M. DesLauriers. (2015) three bmfont text. [Online]. Available: http://experience-monks.github.io/three-bmfont-text/

[24] D. M. Lumbreras. (2020) Babiaxr. [Online]. Available: https://babiaxr.gitlab.io

[25] c frame. (2017) super-hands-component. [Online]. Available: https://c-frame.github.io/aframe-super-hands-component/

Apéndices

A

Manual de usuario

En este anexo se pretende explicar el funcionamiento del desarrollo realizado con tres maneras de uso, una para el usuario que pretende utilizar una demostración ya preparada, otra para un diseñador que pretenda crear demostraciones, pero sin interés por comprender como funciona el código de una manera más profunda y la última, para un desarrollador que pretenda aprender el funcionamiento de todo el código para posteriormente hacer modificaciones o añadir funcionalidades.

A.1. Usuario final

En este primer manual se deben conocer pocos conceptos, algunos ya explicados en la memoria de este trabajo.

Si nos presentan una escena ya creada, muy posiblemente lo primero con lo que nos enfrentemos sea con la adaptación al uso de gafas de realidad virtual y al entorno que nos rodea. Este tipo de adaptaciones requieren de un periodo corto de tiempo, pero que depende de cada persona. Para algunas puede hacerse complicado debido a que puede provocar hasta mareos. Si ya se han utilizado gafas de realidad virtual previamente, este periodo de adaptación no será necesario.

Para imaginar como podría ser una escena que nos proporcionen, podemos mirar las imágenes proporcionadas en las figuras 5.2 o 4.13. Dentro de esa escena podremos movernos de varias maneras, dependiendo de la configuración que se haya proporcionado, pero como base, siempre podremos hacerlo moviéndonos en el mundo real, de manera que en la escena realizaremos los mismos movimientos.

En caso de que nos hayan ofrecido otras maneras de movernos, la más normal viene dado por el movimiento con los joysticks de los mandos, que simularán el movimiento que se haría en un videojuego en dos dimensiones, moviendo la cámara para posicionarlos con la rotación de nuestra cabeza.

Después de conocer como nos podemos mover, solo queda conocer como interactuar con el entorno. Y eso se realizará gracias a las manos que podemos ver en escena. Apretando uno de los gatillos, veremos como la mano realiza la acción de agarrar. Lo que nos permitirá, al acercarnos a una esfera que represente los ejes vectoriales, agarrarla y moverla a nuestro antojo.

Como aclaración y para no crear confusión al usuario, a continuación de cada movimiento de la esfera, al soltarla veremos como se recalculan los valores y se vuelven a ubicar dentro del espacio designado a cada gráfico.

Para terminar, si nuestra escena dispone de botones y/o mirror, como en la figura 5.3, podremos agarrar de igual manera los botones con las manos, para cambiar de gráfico, y también podremos mover las esferas del mirror igual que hicimos con las del gráfico principal. Viendo de esta manera como se mueve su espejo.

A.2. Diseñador de entornos/demostraciones

Este apartado está dirigido a usuarios que busquen crear escenas utilizando los componentes.

Para poder utilizar lo implementado en este trabajo, deberemos conocer el funcionamiento de dos componentes únicamente. *toprint-config* y *bubbles-star-coordinates*. En este apartado 4.4 podremos ver como funcionan ambos componentes con detalle, haciendo uso de las tablas 4.3 y 4.4 donde encontraremos todos y cada uno de los atributos que podemos utilizar para personalizar los componentes y sus valores por defecto.

Junto a esto, siempre podremos utilizar algunos componentes para hacer más bonita nuestra escena, o añadir algo más de contexto para comprender las dimensiones de cada componente, como pueden ser paredes, suelo, estantes o lo que consideremos oportuno con el espacio del que dispongamos.

El único componente de uso obligatorio es *super-hands-component* que nos permitirá utilizar las manos para agarrar objetos.

Código A.1: Declaración de componente *super-hands-component*.

```
1 <a-entity sphere-collider="objects:a-sphere" super-hands hand-
    controls="hand: left"></a-entity>
2 <a-entity sphere-collider="objects:a-sphere" super-hands hand-
    controls="hand: right"></a-entity>
```

Esta sería una manera de implementarlo, es algo muy sencillo que nos tomará solo dos líneas de código, en la que añadimos cada una de las manos y con el sphere-collider, asignamos el tipo de objeto que queremos agarrar. En caso de tener botones como a-box, tendremos la opción de cambiar una de las manos para que agarre a-box y la otra a-sphere.

Esto se suele añadir dentro de la entidad que controla el movimiento del usuario .

Quedando algo como esto finalmente, donde también se ha añadido funcionalidades para el 'debugging', dándole al ratón las mismas características que a las manos, para poder probar en un entorno de dos dimensiones como es la pantalla de nuestros ordenadores.

Código A.2: Declaración de componente *super-hands-component* con rayCaster.

```
<a-entity position="0 1 2" movement-controls="fly:true; speed
    :0.1">
    <a-entity sphere-collider="objects:a-sphere" super-hands
       hand-controls="hand: left"></a-entity>
    <a-entity sphere-collider="objects:a-sphere" super-hands
       hand-controls="hand: right"></a-entity>
    <a-entity position="0 2 0" camera look-controls
              capture-mouse
              cursor="rayOrigin:mouse"
              static-body="shape: sphere; sphereRadius: 0.001"
              super-hands="colliderEvent: raycaster-
                 intersection;
                         colliderEventProperty: els;
                         colliderEndEvent:raycaster-
                            intersection-cleared;
                         colliderEndEventProperty: clearedEls;
                            ">
    </a-entity>
</a-entity>
```

A.3. Desarrollador

Por último, para cualquier desarrollador que quiera modificar el código de este proyecto, o simplemente quiera conocer su funcionamiento de una manera más profunda, podrá revisar el apartado 4.4 de igual manera, aunque no solo necesita pararse a ver los dos componentes, sino que puede analizar los diagramas de secuencia como ejemplos, y el resto de la explicación dada en ese apartado.

Como información complementaria, siempre puede acceder a este repositorio de GitHub donde se encuentra el código:

- Repositorio GitHub