UNIVERSIDAD
REY JUAN CARLOS

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Curso Académico 2019/2020

Trabajo Fin de Grado

VISUALIZACIÓN DE DATOS EN REALIDAD VIRTUAL: EVOLUCIÓN DE SISTEMAS

Autor : Álvaro Villalba Cabañas

Tutor : Dr. Jesús María González Barahona

Trabajo Fin de Grado

Visualización de datos en Realidad Virtual: Evolución de Sistemas

Autor : Álvaro Villalba Cabañas

Tutor : Dr. Jesús María González Barahona

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 20XX, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 20XX

Dedicado a

todos los que han hecho esto posible

II

Agradecimientos

Quiero agradecer especialmente a mis padres por darme todo el entorno necesario para poder desarrollar todo lo que quisiera, a mi hermano por ser una ayuda constante, mi pareja por aguantarme y apoyarme día a día con todo lo que ello conlleva y a toda mi familia en general que siempre me han mostrado su ilusión y su alegría por que terminara la carrera. Quiero agradecer a todas las personas que siempre han confiado en mí y han estado apoyándome en cada momento, mostrándome lo afortunado que soy de tener a tanta gente buena cerca.

También quiero agradecer a todos los profesores que he tenido tanto en la universidad como fuera de ella por ayudarme a ir abriendo los ojos y despertar esa curiosidad que estaba en mi interior.

Resumen

El objetivo de este trabajo es explorar los diferentes algoritmos de empaquetamiento, es decir, algoritmos que pretenden compactar lo máximo posible sin que se solapen los elementos dentro de la escena dejando el mínimo hueco entre los elementos. Una vez exploremos cuál es el algoritmo que más se ajuste a los requisitos y prioridades tendremos que diseñarlo y desarrollarlo desde cero sin partir de ningún algoritmo previo y sin basarnos en código de terceros. Por último desarrollaremos una prueba de concepto creando un algoritmo de atracción y repulsión que atraiga a los elementos de la escena entre si, para que sirva de base para futuros proyectos. Como resultado de este trabajo hemos desarrollado una biblioteca de representación de datos en realidad virtual con un algoritmo que coloca los elementos que le pasamos como parámetro en forma de espiral y una prueba de concepto para futuros trabajos que atrae a los elementos de la escena hacia el de mayor tamaño.

Para el desarrollo del proyecto hemos utilizado A-Frame, framework de javascript que se suele utilizar para crear escenas en realidad virtual. Además como hemos trabajado dentro del navegador hemos trabajado con HTML5 y con JavaScript. Para el mantenimiento del proyecto hemos creado un repositorio de control de versiones y lo hemos alojado tanto en GitHub como en GitLab, asegurándonos una correcta sincronización entre los dos servicios.

Summary

The objective of this project is explore the packaging algorithms, that is, algorithms that want to compact as much as posible the elements in the scene taking care in the collisions between the elements. The elements should have the minimum space between them. When we explore all the algorithms that we think should be fine, we have to adjust that algorithm to the requirements and priorities defined before starting the project. When we have all fine we have to design and develop the algorithm starting from zero without looking at the code of other algorithms. By last we have to develop one proof of concept based in one algorithm of atract and repulsion, that is, we have to atract the elements in the scene to the bigger one.

For the development of the project we use A-Frame, a JavaScript framework that create virtual reality scenes. As we have worked at the browser, we have to work with HTML5 and JavaScript too. For the project maintenance, we have used the GitHub and Gitlab tools synchronized.

Índice general

Índice de figuras

Capítulo 1

Introducción

Antes de comenzar con la descripción del proyecto, creemos conveniente hacer una breve introducción del contexto de este trabajo y de las causas que nos han llevado a hacerlo. Al final el mundo de la tecnología está evolucionando y todo lo que habíamos avanzado a lo largo de los últimos 10 años cada vez se queda antes desfasado, la velocidad que está tomando la tecnología es vertiginosa y nos tenemos que adaptar a ella. La tecnología ahora se está enfocando cada vez más al análisis de datos [1], tanto a la obtención como al tratamiento y la posterior visualización de ellos. En este proyecto queríamos enfocarnos sobre todo en la última fase, en la fase de la visualización de los resultados ya que queríamos trabajar desde un principio con el framework A-Frame para crear escenas en realidad virtual.

Además de centrarnos en la fase de visualización de los datos en realidad virtual lo vamos a tener que hacer con una serie de requisitos previos, uno de ellos será el de trabajar con el framework de JavaScript [2] A-Frame [3], otro de ellos será que debe ser visualizado tanto en un navegador web como en unas gafas de realidad virtual y por último el código debe cumplir tres requisitos, uno de ellos es que no tiene que ser de dificil comprensión para futuras personas, no debe ser elevado el tiempo de cómputo y el resultado tiene que ser relativamente bueno.

1.1. ¿Por qué A-Frame?

Queremos realizar un trabajo que nos sirva para la visualización de datos en realidad virtual tanto en el navegador como en dispositivos orientados a la realidad virtual como son las gafas de realidad virtual. Dentro de las tecnologías que nos ofrecían esto nos decidimos por la que más

se asemeja con el desarrollo frontend, parte del desarrollo encargada de trabajar desde el lado del navegador con tecnologías como HTML, CSS y JavaScript. Para que fuera lo más parecido a este desarrollo frontend se puso como requisito del proyecto la utilización de A-Frame.

A-Frame nos permite crear escenas en realidad virtual y poder visualizarlas tanto en el navegador como en las gafas de realidad virtual, utilizando tecnologías conocidas en el desarrollo frontend [5] por lo que nos será de fácil adaptación dado que ya conocemos gran parte de esos lenguajes.

A la hora de representar los elementos en la escena hicimos una búsqueda muy superficial en la que al final decidimos hacer los algoritmos desde cero dado que queríamos explorar las dificultades de diseñar y desarrollar un algoritmo y no basarnos en uno que ya ha pasado y solucionado esos problemas.

A la hora de modelar cómo queremos representar los elementos en la escena vimos que había varios algoritmos en dos dimensiones que hacían cosas parecidas a lo que queremos hacer, pero de nuevo decidimos partir de cero e ir explorando todos los procesos por nosotros mismos.

1.2. Problema de 2D y por qué usamos 3D

Si nosotros queremos representar datos en dos dimensiones por ejemplo el número de alumnos que tenemos en una clase en un eje y las clases en otro eje podemos utilizar gráficos de barras, donde la altura de cada barra significa el número de alumnos y las clases las vamos a tener en el otro eje, en la base de la barra.

Figura 1.1: Diagrama de barras que representa el número de alumnos en cada clase

Nosotros queremos representar datos con tres propiedades, para ello tenemos que utilizar tres dimensiones. La anchura de la figura determinará una de las propiedades, la profundidad nos dará otra y, por último, la altura nos dará la tercera propiedad de interés del elemento en

cuestión. Además de querer representar los elementos en tres dimensiones, para poder tener tres valores de interés a representar queremos tener los elementos colocados dentro de la escena de la forma más compacta entre ellos, es decir, dejando el menor espacio posible entre los elementos de la escena.

1.3. Empaquetamiento

No queremos que los elementos representen lo que queremos pero estén muy dispersos por la escena si no que queremos que se encuentren todos en torno a un elemento central. Por ejemplo, si colocamos los elementos de mayor a menor, suponiendo que son archivos dentro de un directorio y que el volumen del elemento es el dato relevante, los colocaremos todos en torno al fichero que más ocupa y tendremos una clara visualización de cuales son los elementos de nuestro directorio que son los más pesados, y actuar o tomar alguna decisión en función de eso. A esta forma de representar los elementos en torno al elemento central lo llamaremos empaquetamiento ya que queremos que la figura resultante al representar todos los elementos esté lo más compacta posible.

Cabe destacar también que la finalidad de este trabajo no es encontrar la forma óptima de empaquetamiento si no diseñar y desarrollar desde cero los algoritmos necesarios para el posicionamiento dentro de la escena de la forma más empaquetada posible siguiendo tres principios o claves que debemos cumplir: tenemos que desarrollar el código sin comprometer la velocidad de cómputo y asegurando una fácil comprensión para futuros contribuyentes además de ser un código relativamente simple. Para conseguir este objetivo tendremos que investigar los diferentes algoritmos que se nos ocurran y, posteriormente desarrollarlos, teniendo especial cuidado en no olvidarnos de los tres principios que eran requisitos del proyecto.

En este proyecto se pretende resolver el problema del empaquetamiento en realidad virtual utilizando técnicas de 2 dimensiones y diseñando y desarrollando todos los algoritmos desde cero. Hemos detectado que hay diversas ocasiones en la que resulta de utilidad tener una muestra de datos y agruparlos en base a alguna propiedad formando una figura geométrica. Entre las diferentes posibilidades, vimos que hay algoritmos que dado un lugar geométrico, bien sea un cuadrado, un círculo o cualquier polígono regular, agrupan los datos dentro del lugar geométrico pero no conseguimos encontrar ningún algoritmo que dado unos datos los agrupara formando

un lugar geométrico deseado por lo que encontramos de especial interés explorar esta segunda opción e investigar todas las posibilidades que se nos estaban mostrando.

1.4. Nuestro objetivo

El objetivo es desarrollar una biblioteca para complementar los componentes ya existentes en BabiaXR [13], proyecto en el cual se agrupan diferentes formas de representación de datos. El problema a resolver es agrupar en forma circular y de la forma más compacta posible los distintos elementos que le pasamos al componente como parámetro y así poder tener una visión mucho más clara de los datos que estamos tratando. Usaremos técnicas pensando en un espacio de dos dimensiones (2D) para posteriormente representarlo en realidad virtual. Usaremos técnicas de dos dimensiones ya que todos los elementos los vamos a mostrar sobre el mismo plano y lo representaremos en realidad virtual utilizando escenas de A-Frame para ello.

Pese a haber diferentes soluciones para la resolución del problema, decidimos escribir el código desde cero por varios motivos, el primero fue el interés por explorar el diseño y desarrollo de los algoritmos desde cero, sintiendo cuales son los problemas a los que podríamos enfrentarnos y ver de qué manera se podrían solucionar. Otro de los motivos por el cual decidimos escribir los algoritmos desde cero sin utilizar código o algoritmos ya existentes es que no estábamos muy seguros de que el código existente resolviera nuestro problema por completo y creímos que era más conveniente conocer perfectamente nuestro código para diseñarlo y desarrollarlo de manera que pueda ser escalable y pueda ser mejorado siguiendo la tendencia que hemos iniciado.

El inconveniente de diseñar y desarrollar los algoritmos partiendo desde cero es que tenemos que cumplir ciertas propiedades como la de que el código sea relativamente sencillo de entender por otras personas, tiene que ser eficiente y no gastar demasiados recursos de la máquina donde se ejecute y tiene que resolver el problema que hemos planteado de una manera relativamente buena.

En este proyecto vamos a investigar y desarrollar diferentes algoritmos que nos permitan agrupar los elementos de varias maneras, decidiendo cuál es el que más se ajusta al objetivo buscado y sacarle el máximo provecho. Para sacarle el máximo provecho probaremos diferentes formas de agrupación pensando las maneras más eficientes a la hora de empaquetar, es

decir, dejando el mínimo hueco posible entre los elementos de la escena. También probaremos a representar los elementos de la escena con diferentes formas geométricas, como cubos y cilindros, observando cuál es la forma geométrica que mejor se ajusta a nuestros objetivos y además cumple mejor el requisito de dejar el mínimo espacio entre los elementos de la escena.

Al desarrollar estos componentes e incorporarlos al paquete de componentes de BabiaXR, el objetivo es que la gente que quiera desarrollar una aplicación para visualización de datos pueda utilizar los componentes a su gusto, siendo mucho más fácil conseguir lo que se proponga sobre la base proporcionada por nuestra biblioteca. Por ejemplo alguien que quiera desarrollar una aplicación que represente los diferentes directorios del ordenador, con cada archivo siendo un bloque, podemos tener una visión gráfica de cuáles son los directorios que más ocupan dentro de nuestro ordenador y si los ficheros no nos interesan incluso eliminarlos al ver que el bloque es demasiado grande.

Para dos dimensiones existen una gran variedad de bibliotecas que agrupan los distintos elementos según la forma que queramos pero para las escenas de tres dimensiones o escenas de realidad virtual no disponemos de ninguna biblioteca, o al menos no la conocemos, que nos agrupe los distintos elementos generando nuevas formas geométricas por lo que puede ser de utilidad para la comunidad.

Los objetivos a seguir a lo largo del diseño y desarrollo del proyecto van a ser muy diferentes unos de otros pero en cada uno de ellos encontraremos ciertas dificultades que nos servirán de aprendizaje para el futuro. En primer lugar tendremos que familiarizarnos con A-Frame, saber qué es un componente y como podemos construir escenas dentro de A-Frame, por otro lugar tendremos que centrarnos en el diseño de los algoritmos para lo que tendremos que dedicar un trabajo de investigación para conocer cuál puede ser la forma de colocación que mejor se ajuste a nuestros requisitos teniendo en cuenta las limitaciones que tenemos. Al finalizar cada paso tendremos que comprobar el correcto funcionamiento y que el código sigue nuestros tres principios de que sea entendible, eficiente y resuelva el problema.

1.5. Tecnologías utilizadas

Para desarrollar este proyecto se va a utilizar la tecnología A-Frame., que es un framework web de código abierto de Three.js, librería de JavaScript, para crear escenas de realidad virtual.

Era uno de los objetivos principales junto con la idea de trabajar dentro del navegador por el cual decidimos abordar este proyecto. A-Frame es uno de los frameworks que creemos con mayor potencial en cuanto a la creación de escenas en realidad virtual ya que está escrito en el lenguaje más popular según StackOverflow el pasado año [15].. La actual popularidad del lenguaje de programación junto con la creciente tendencia a la realidad virtual hace de A-Frame un gran framework para trabajar con él en el trabajo de fin de grado. Podemos observar esta creciente tendencia a la realidad virtual observando un artículo [8] publicado por el departamento de innovación de la empresa Iberdrola, en el cual aseguran que ”la inversión en RV [6] y RA [7] se multiplicará por 21 en los próximos cuatro años”

Para mantener un control de versiones del código desarrollado utilizaremos Git y alojaremos el repositorio en los servicios web GitHub y GitLab, habilitando un repositorio compartido para que el código esté disponible en ambas plataformas simultáneamente. Creemos que esto nos facilita el trabajo a la hora de que el tutor pueda conocer los avances a lo largo del proyecto, también tenemos la posibilidad de crear issues o tareas para saber claramente las tareas que tenemos que desarrollar en cada sprint.

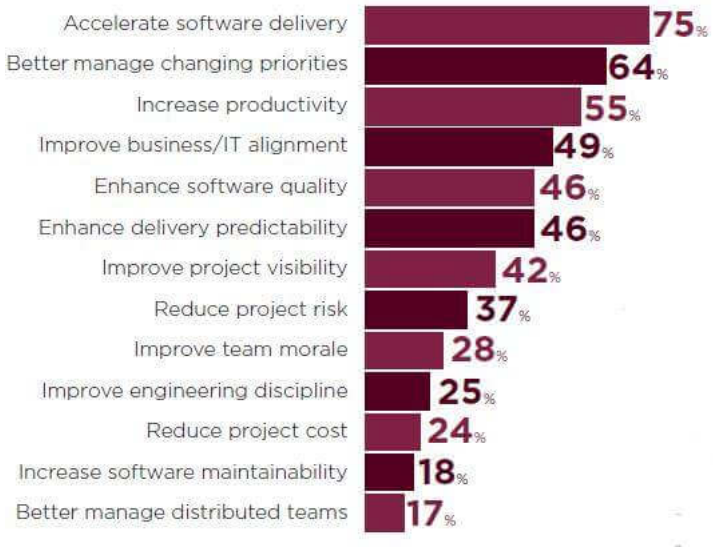
Como uno de los requisitos para el proyecto que nos marcamos al iniciarlo, trabajaremos en el navegador por tanto utilizaremos HTML5 y JavaScript. JavaScript será nuestro lenguaje de programación y con el desarrollaremos todos nuestros algoritmos que darán lugar a los componentes que vamos a utilizar. Por otro lado HTML nos va a permitir ver en el navegador las escenas que estamos creando con los diferentes elementos dentro de la escena.

Como vamos a trabajar dentro del navegador vamos a tener que trabajar con los estándares web WebVR y WebGL lo que nos va a permitir tener una sensación de realidad virtual dentro del navegador y observar nuestra escena tal y como lo haríamos con las gafas de realidad virtual.

1.6. Metodología de trabajo

Para el desarrollo del trabajo vamos a seguir metodologías ágiles [4], muy utilizadas en empresas de desarrollo de software. Las metodologías ágiles han demostrado ser más eficientes y hacen llegar a los equipos de trabajo que las llevan a cabo a tener unos resultados mucho más productivos y llegar mucho más rápido a la meta. A continuación adjunto una imagen de la empresa tecnológica Adeva [16] en la que se pueden ver algunos porcentajes en los que se

vieron aumentados algunos factores en los equipos que trabajaban con metodologías ágiles.



1.7. Estructura de la memoria

En este apartado vamos a resumir los diferentes capítulos para una mejor comprensión de la memoria:

- En el primer capítulo se hace una introducción al proyecto, describiendo brevemente el problema que queremos solucionar y posteriormente unos pasos u objetivos que debemos ir siguiendo para conseguir solucionar el problema.
- En el capítulo 2 describimos todos los objetivos del proyecto más profundamente, tanto el objetivo principal como los objetivos específicos, en este apartado nos centraremos un poco más en cada uno de los objetivos que nos hemos marcado para conseguir solucionar el problema.
- A continuación se presenta el estado del arte y las tecnologías utilizadas en el capítulo 3. En él describiremos todas las tecnologías que hemos utilizado para la resolución del problema y los diferentes componentes similares que hay en la actualidad y pueden servir como referencia para el desarrollo del proyecto.
- En el capítulo 4 vamos a describir el diseño y la implementación del proyecto, enfocándonos en la arquitectura seguida, la metodología de trabajo y describiendo los componentes que vamos a crear

- En el capítulo 5 vamos a analizar los resultados obtenidos y vamos a describir los algoritmos utilizados para poder llegar a esos objetivos
- Por último, en el capítulo 6 procederemos con las conclusiones del proyecto, describiendo los puntos en los que más hemos aprendido y ofreciendo nuevas ideas para posibles próximos proyectos.

Capítulo 2

Objetivos

2.1. Objetivo general

En este proyecto tenemos dos grandes objetivos entre manos, el primero de ellos es experimentar con diferentes algoritmos de empaquetamientos para ver cual de ellos cumple de una mejor manera los requisitos previos que nos hemos propuesto. En segundo lugar nos propusimos el objetivo de diseñar y desarrollar los algoritmos desde cero.

Los requisitos propuestos para este proyecto son varios y entre ellos están el de trabajar con el framework A-Frame, que funcione tanto en dispositivos de realidad virtual como en el propio navegador, el funcionamiento tiene que ser rápido y el código ha de ser de fácil comprensión para que futuros contribuyentes a la librería puedan aportar de una manera más sencilla.

Los requisitos que debe cumplir el algoritmo es que una vez experimentados los diferentes algoritmos de empaquetamiento que se nos ocurran, el resultado debe mostrar los elementos de la escena relativamente cerca unos de otros ya que si no no solventaríamos el problema del empaquetamiento.

Para la obtención de este objetivo trataremos de desarrollar los algoritmos desde cero para experimentar y enfrentar las dificultades y los problemas que nos podamos encontrar al diseñar los algoritmos y posteriormente desarrollarlos.

Para diseñar los algoritmos tendremos en cuenta diferentes restricciones en cuanto al número de bloques, tamaño de los bloques, etc. También mantendremos una localización de los elementos dentro de la escena relativamente constante, lo que nos permitirá identificar con facilidad cuales son los bloques situados en cada lugar.

Por otro lado tenemos el objetivo de crear una prueba de concepto que simule una atracción dentro de la escena hacia el elemento central.

2.2. Objetivos específicos

2.2.1. Trabajar con el framework A-Frame

Uno de los incentivos que nos hizo decantarnos por este proyecto fue la utilización de realidad virtual dentro del navegador pero que luego se pudiera ver también con gafas de realidad virtual. Para lograr este objetivo elegimos A-Frame como el encargado de crear nuestras escenas de realidad virtual ya que trabaja con JavaScript y eso es un aliciente a la hora de escogerlo. Una vez escogido el framework nos marcamos el objetivo de trabajar con él.

2.2.2. Conseguir un algoritmo de empaquetamiento en dos dimensiones

De los objetivos más importantes en este trabajo es conseguir un algoritmo que nos agrupe los elementos en la escena en torno a un elemento central siguiendo un algoritmo que diseñemos y desarrollemos nosotros desde cero. Para poder conseguir este objetivo tendremos que explorar los algoritmos ya existentes y ver si nos valen como base o tendremos que partir totalmente desde cero, asimismo tendremos que probar algoritmos que creamos que cumplen con nuestros requisitos de que sean de fácil comprensión para terceras personas, sea rápido en tiempo de cómputo y genere un resultado relativamente bueno.

2.2.3. Conservar el orden lógico de los elementos dentro de la escena

Otro de los objetivos que tenemos que lograr cuando diseñemos el algoritmo es que los elementos no se podrán colocar en posiciones indistintas a lo largo de la escena sin que tengamos el control de dónde se colocará el siguiente elemento. Tenemos que diseñar un algoritmo que sea capaz de colocar los elementos conservando un orden lógico para que si sabemos la posición del anterior elemento sepamos aproximadamente dónde irá situado nuestro próximo elemento en la escena.

2.2.4. Diseño de algoritmos

Como ya hemos mencionado anteriormente, pese a haber soluciones que más o menos podían resolver nuestro problema, no dimos cuenta de que ninguna fuera lo suficientemente buena como para hacer todo el proyecto en base a ese algoritmo por lo que decidimos desarrollar y diseñar los algoritmos que necesitamos desde cero.

En primer lugar tenemos que diseñar un algoritmo que nos permita colocar los bloques de la forma más empaquetada posible en torno a un elemento central, tendremos que probar con diferentes algoritmos y observar los resultados.

2.2.5. Encontrar la forma de colocación

Una vez tengamos desarrollado el primer componente que nos represente un bloque en la escena con los datos obtenidos de las propiedades del objeto JSON, tenemos que ponernos a probar las diferentes formas de colocación que se nos ocurran, diseñando los algoritmos previamente para que nos queden los menores espacios posibles entre los bloques y de una sensación de empaquetamiento en torno al elemento central.

2.2.6. Probar con diferentes formas geométicas

Durante el desarrollo de los algoritmos, tendremos que hacerlo con la idea de que sean genéricos, es decir, que nos puedan servir para poder representar diferentes formas geométricas sin apenas modificaciones.

Deberemos probar con diferentes formas geométricas para estudiar el posicionamiento y con qué forma geométrica obtenemos una mejor representación de los elementos de una forma más compacta.

2.2.7. Comprobar el resultado de los algoritmos

Una vez tenemos el diseño y el posterior desarrollo de los algoritmos, tenemos que comprobar que cumple con todos los requerimientos que nos pusimos al iniciar el diseño, es decir, tiene que ser de simple comprensión, puede ser escalable posteriormente y resuelve nuestro problema. En el caso que nuestro algoritmo cumpla con todas esas premisas podemos darlo por

válido y continuar con los siguientes objetivos.

2.2.8. Algoritmo de atracción-repulsión

Por último trataremos de desarrollar una prueba de concepto en la cual un elemento centrar atraerá hacia sí mismo a otros elementos en la escena, tratando de que la distancia que hay entre todos ellos disminuya y que la distancia entre el elemento central y cada uno de los elementos también disminuya y todo esto sin que se produzca ningún solapamiento en la escena.

Capítulo 3

Tecnologías utilizadas y Estado del arte

3.1. Tecnologías utilizadas

Para el desarrollo del proyecto se han utilizado las tecnologías que se describen a continuación:

3.1.1. A-Frame

A-Frame es un framework web de código abierto de three.js, que a su vez es una librería de JavaScript, uno de los lenguajes con más popularidad dentro del entorno de desarrollo de software tanto de parte del navegador como de parte del servidor debido a sus numerosos frameworks que ayudan a esa versatilidad. El objetivo de A-Frame es poder crear escenas en realidad virtual dentro del navegador.

Al ser un framework de three.js vamos a poder utilizar todas las funciones existentes en three.js, entre las cuales vamos a encontrar especial utilidad las relacionadas con el modelado 3D, el cálculo de distancias entre elementos geométricos y todas las funciones relacionadas con la geometría.

A-Frame tiene muchos beneficios entre los que está la compatibilidad con los frameworks de JavaScript más utilizados: AngularJS, React y Vue.js. Otro de sus beneficios es que no es necesario instalar nada, simplemente basta con abrir una etiqueta `<a-scene>` dentro del body de nuestro fichero HTML, esta etiqueta nos creará una escena de realidad virtual y posteriormente podremos crear componentes para que cuando le pasemos determinados parámetros nos

muestre lo que hemos programado.

Cuenta con una excelente documentación en su página web, además de una comunidad muy amplia y con multitud de componentes, tanto de la comunidad [17] como de los propios creadores de A-Frame [18] que se pueden usar gratuitamente debido a que es software libre. Para empezar a utilizar A-Frame es recomendable hacer el tutorial desde cero [20] , luego como vamos querer crear nuestros propios componentes podemos ver el tutorial de como crear un componente para A-Frame [**?**]

La arquitectura de A-Frame es una arquitectura entidad-componente, la más utilizada en el desarrollo de videojuegos, posee una gran flexibilidad ya que todos los elementos presentes en la escena de realidad virtual pueden ser modelados como entities (entidades). Cada una de las entidades posee uno o varios componentes que contienen datos o estados.

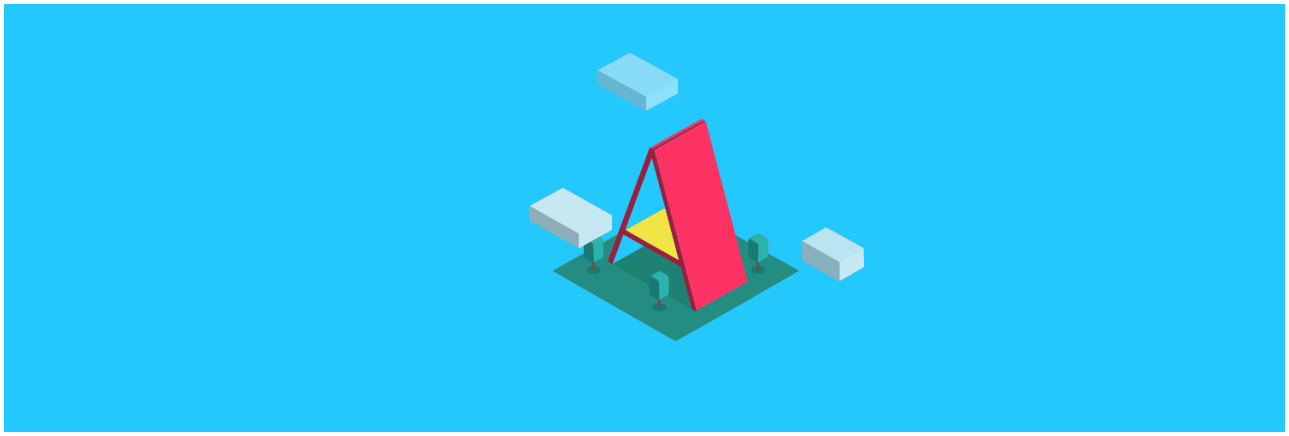


Figura 3.1: Ejemplo de escena en a-frame.

En esta figura podemos observar una escena en A-Frame compuesta por varios planos para formar la letra A, el suelo y las nubes, con diferentes propiedades de altura, anchura y profundidad y varios tetraedros para formar los árboles.

3.1.2. HTML

HTML [9], siglas de HyperText Markup Language en inglés, en español lo podríamos traducir como lenguaje de marcado de hipertexto. HTML es un lenguaje de marcado que determina la estructura que va a tener nuestra página web mediante el uso de etiquetas. En el caso de nuestro proyecto va a ser la punta del iceberg, va a ser nuestra parte visible en la que vamos a incluir nuestra etiqueta `<a-scene>` para generar la escena de realidad virtual y nuestra etiqueta para

llamar al componente. Luego vamos a poder utilizar todas las etiquetas propias de HTML para modificar las propiedades de la escena, por lo tanto mediante HTML vamos a hacer que nuestra escena tome la apariencia que deseamos.

El marcado en HTML incluye elementos especiales tales como `<a-scene>`, `<head>`, `<title>`, `<body>`, `<header>`, `<footer>`, `<article>`, `<section>`, `<p>`, `<div>`, `<span>`, `<img>`, `<aside>`, `<audio>`, `<canvas>`, `<datalist>`, `<details>`, `<embed>`, `<nav>`, `<output>`, `<progress>`, `<video>`, `<ul>`, `<ol>`, `<li>`, y muchos otros más.

Además de elementos, HTML tiene atributos por los cuales vamos a definir propiedades especiales de los elementos. De esta forma una línea de código en HTML podría parecerse a esto : `<nombre-de-elemento atributo=''valor''>Contenido</nombre-de-elemento>` Cabe destacar que el valor es una variable que afecta al atributo, pudiendo conseguir efectos diferentes. Por ejemplo el atributo puede ser un `href` y la variable valor una ruta hacia una imagen dentro de un elemento `img`.

3.1.3. JavaScript

JavaScript es un lenguaje de programación interpretado, lo que tiene sus cosas buenas y sus cosas malas. Como cosas buenas es que es fácilmente modificable módulos muy grandes del código y como desventajas frente a los lenguajes de programación compilados es que los interpretados al tener un intérprete va a ser más lento en la ejecución que los compilados ya que los compilados traducen a código máquina las instrucciones que le hemos dicho y los interpretados traducen a medida que es necesario.

Para trabajar con navegadores web utilizar lenguajes de programación interpretados es un acierto ya que así no dependes de la plataforma donde se ejecute el código si no del propio navegador.

Todos los navegadores modernos interpretan código JavaScript integrado en sus páginas web, de este modo es muy normal trabajar con HTML para definir la estructura de la web, CSS para darle los estilos que prefieras y tome un aspecto más moderno y JavaScript para dotar de funcionalidad los distintos elementos de la página web.

Para poder interactuar con la página web se provee al lenguaje JavaScript de una implementación del DOM, Document Object Model, a través del cual los programas pueden acceder y

modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente.

Cabe destacar la versatilidad del lenguaje JavaScript, siendo uno de los más populares en la programación web debido a que puede ser utilizado tanto del lado del navegador como del lado del servidor. Desde el lado del navegador lo podemos ver a través de los frameworks AngularJS, Vue.js y React. Desde el lado del servidor lo podemos encontrar con Node.js.

3.1.4. Three.js

Three.js [10] es una liviana biblioteca de JavaScript utilizada para representar gráficos en 3D y crear escenas de realidad virtual en un navegador web. Esta biblioteca fue creada y liberada en GitHub por el español Ricardo Cabello en abril de 2010, conocido por su seudónimo de Mr.doob.

Para renderizar los gráficos en 3D podemos utilizar WebGL, API de JavaScript para renderizar gráficos en 3D y 2D. Three.js es una librería creada sobre WebGL, lo que garantiza la compatibilidad de todos los navegadores modernos. Three.js es a WebGL lo que JQuery es a JavaScript, ofrece una abstracción de las complejidades y ameniza el uso de los gráficos 3D a los programadores.

Three.js posee muchas funciones para el tratamiento con elementos geométricos y escenas de realidad virtual, habiendo una función ya creada para casi todo lo que puedas imaginar. Todas las operaciones con matrices, vectores en el espacio... También encontramos funciones para representar elementos geométricos como cajas, cilindros, esferas... Hay funciones para calcular las distancias entre ellos, los vectores de los centros y muchas más.

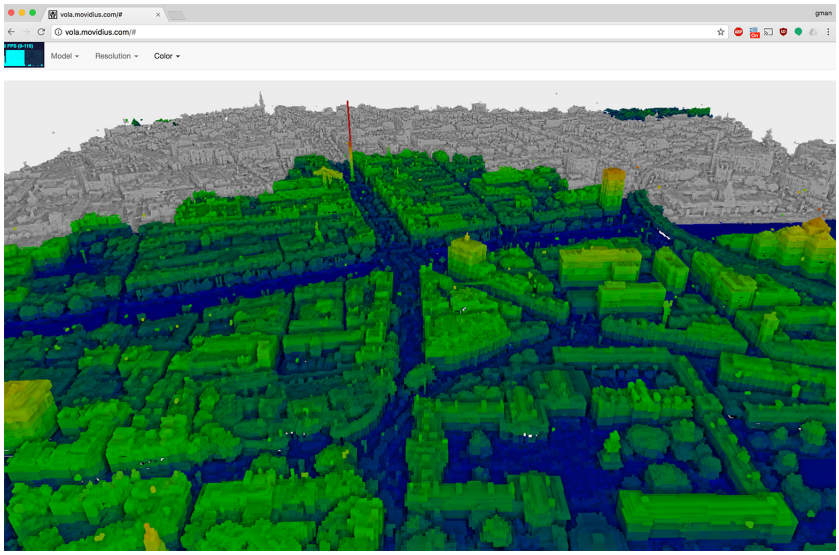


Figura 3.2: Ejemplo de escena con Three.js

En la figura anterior podemos ver un ejemplo de una escena creada con el lenguaje Three.js. En la escena podemos apreciar numerosos elementos formando una especie de ciudad con sus calles y sus edificios, si nos detenemos a observar la escena podemos apreciar la gran variedad de figuras geométricas que forman la escena y le dan un aspecto de ciudad.

3.2. Trabajos relacionados

En la sección de trabajos relacionados describiremos diferentes representaciones que nos encontramos en el tiempo de investigación y que, en cierto modo, sirvieron de inspiración para la realización del proyecto.

3.2.1. Treemaps

Desde el punto de vista de los algoritmos, uno muy similar es el utilizado en los treemaps [11] ya que son una forma de empaquetamiento de la información y de un vistazo puedes ver mucha información agrupada.

Los treemaps son representaciones muy apropiadas para representar un gran número de datos. La representación es en forma jerárquica y el espacio de la visualización está dividido en rectángulos a los que se les asigna un tamaño y un orden determinado dependiendo de los datos que queramos representar.

En la figura a continuación se muestran los datos recogidos en Florida en 2016 para las elecciones presidenciales y podemos ver todos los estados divididos en cuadriláteros de diferente color. El color azul representa los demócratas y el rojo los republicanos. De esta manera con la intensidad del color podemos ver la diferencia entre la intención de voto, siendo un rojo muy intenso una intención muy grande por votar a Trump y los azules muy intenso por votar a Clinton, a medida que los colores se suavizan, sigue habiendo más intención por votar al bando del color dominante pero la diferencia con el otro bando disminuye.

Estas representaciones muestran una gran cantidad de datos en una misma representación por lo que nos vimos muy inspirados para hacerlo de una forma más visual.

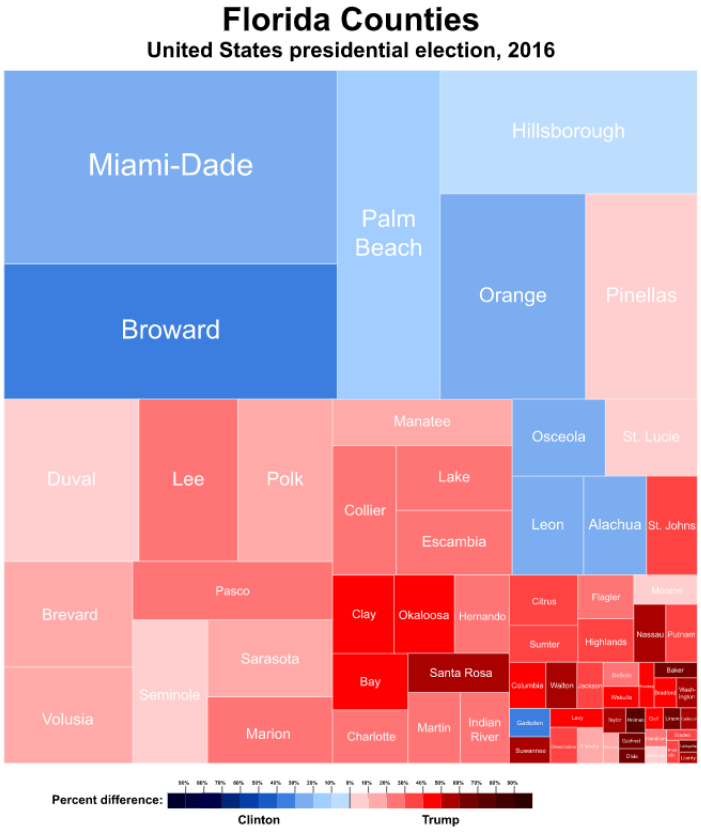


Figura 3.3: Ejemplo de Treemap

Una librería de JavaScript para el uso de treemaps puede ser por ejemplo:

https://observablehq.com/@d3/treemap.

3.2.2. Empaquetamiento en círculos

Otro algoritmo parecido al nuestro puede ser el algoritmo de los círculos que se les puede hacer zoom, es decir, hay círculos grandes que al seleccionarlos se abren y te muestran otros dentro, este algoritmo además de empaquetar clasifica en diferentes secciones que serán las circunferencias grandes. Para nuestro algoritmo podemos hacer algo parecido si primero filtramos la información y se la pasamos filtrada para representarla.

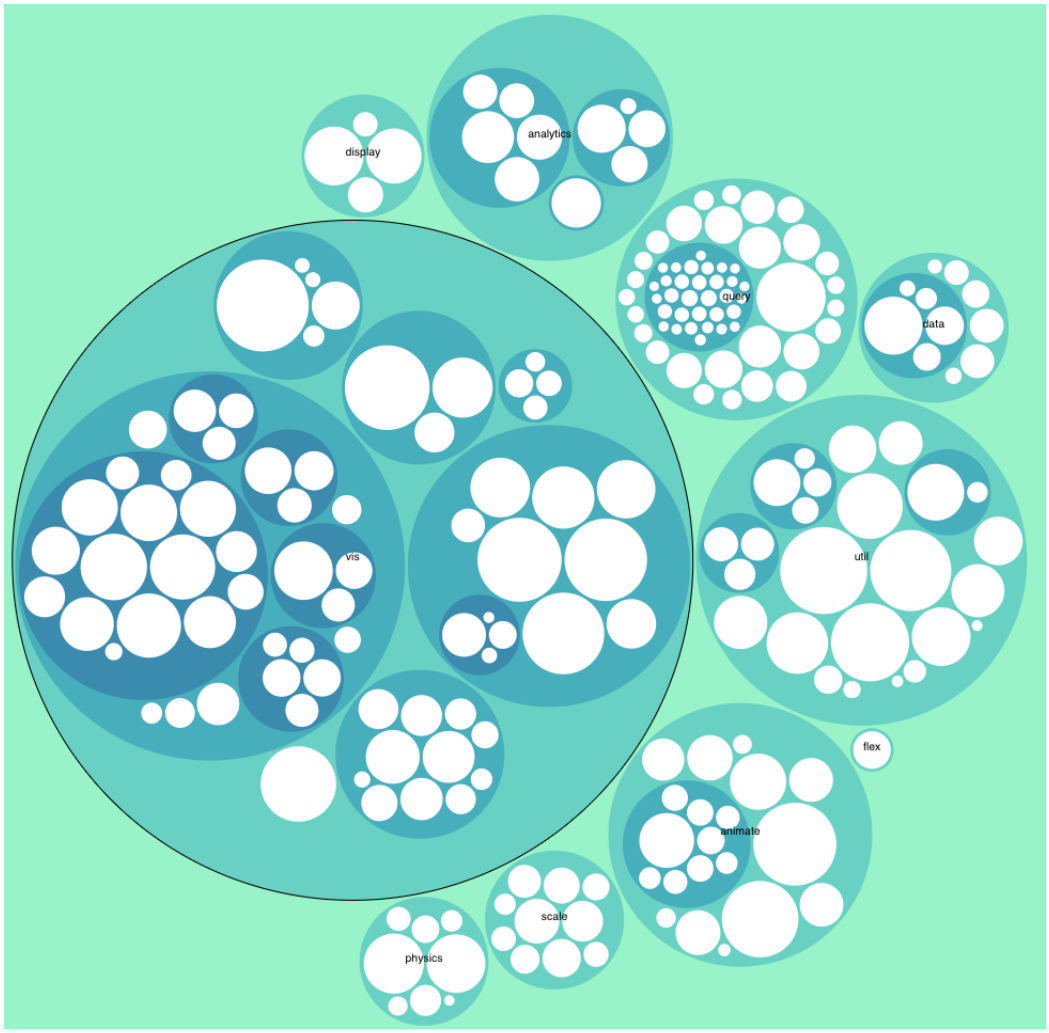


Figura 3.4: Ejemplo de zoomable-circle

En la figura anterior podemos ver el resultado de aplicar el algoritmo de ”zoomable-circle”que podemos ver en https://observablehq.com/@d3/zoomable-circle-packing

3.2.3. Algoritmos de atracción y repulsión

Para la prueba de concepto de atracción y repulsión nos hemos basado en las fuerzas gravitacionales, siendo la distancia que recorren los elementos directamente proporcional al tamaño del elemento al que se acercan.

En dos dimensiones hemos visto algún algoritmo que acerca un nodo a un grafo pero no creo que nuestro algoritmo sea parecido o tenga que parecerse ya que nuestro algoritmo está destinado a acercar todavía más los elementos dentro de la escena.

Capítulo 4

Desarrollo del proyecto

A continuación describo la arquitectura general del proyecto y describo los procesos por los que pasamos en el desarrollo, entrando en los detalles concretos del proyecto y comentando el porqué de cada una de las decisiones que tomamos a medida que se nos iban presentando problemas.

Como metodología de trabajo vamos a seguir una metodología ágil, entre todas ellas escogeremos scrum, una metodología de trabajo muy utilizada en las empresas de software que permite llevar un control del proyecto bastante cercano por parte de los jefes de proyecto, en este caso el tutor del TFG, cuya función es dividir las tareas principales en tareas más pequeñas y dividir las tareas más pequeñas en periodos de tiempo denominados sprints.

Los sprints suelen ser espacios de tiempo de una semana o dos semanas dependiendo de las tareas o funcionalidades que se hayan previsto realizar y el tiempo que se haya estimado para cada tarea. Las tareas a realizar van en consonancia con los objetivos previstos en el capítulo de objetivos, por eso esta forma de trabajar nos ayuda a la hora de realizar la memoria del TFG ya que hemos tenido que realizar el trabajo previo de separar en tareas el proyecto mediante los diferentes objetivos que nos hemos planteado. Las tareas las vamos a colocar en el backlog.

El backlog es un espacio reservado donde debemos anotar todas las tareas que vayamos identificando a lo largo del desarrollo. En un principio en el backlog estarán las tareas principales y algunas secundarias pero a medida que vayamos avanzando con el desarrollo irán surgiendo problemas que deberemos solucionar con nuevas tareas que apuntaremos en el backlog. Las tareas que estén en el backlog las resolveremos en un sprint que agrupe varias tareas con la misma temática o que resuelvan en conjunto el mismo problema.

A lo largo del proyecto, y siguiendo con lo estipulado en las metodologías ágiles, tendremos diferentes tipos de reuniones con el tutor: en primer lugar las llamadas reuniones de planificación de sprint, donde nos reunimos para decidir cual es el objetivo del sprint y cuales son las diferentes tareas a realizar en el sprint. Este tipo de reuniones es muy útil para mantenerse enfocado en un objetivo y no trabajar sin un objetivo claro, de esta manera permanecemos enfocados y ponemos todas las energías en llegar a conseguir el objetivo del sprint. En segundo lugar tenemos las reuniones de seguimiento, normalmente una a la semana como mínimo, en la cual mostramos los avances que hemos tenido y si hemos tenido algún bloqueo podemos contarle al tutor los problemas con los que nos hemos encontrado y nuestro punto de vista para que él nos de el suyo y así poder llegar a una solución más asequible. Estas reuniones también sirven para corregir el rumbo si el tutor ve que nos estamos desviando del foco y son de gran utilidad también para obtener feedback antes de tener el resultado final, de este modo si no es lo que buscábamos podemos redirigir el rumbo más fácilmente. Por último tenemos las reuniones de final de sprint, las cuales sirven para mostrar el resultado final del sprint y ver si hemos conseguido con éxito el objetivo que nos planteamos en la reunión de planificación del sprint. En la reunión de final de sprint solemos hacer una demo con los resultados y los avances del sprint, obteniendo feedback por parte del tutor y de esa reunión saldrán nuevas tareas a realizar que irán al backlog si hay cosas a mejorar.

Una vez aclarada la metodología que vamos a utilizar a lo largo de todo el proyecto vamos a adentrarnos un poco más en él:

4.1. Arquitectura general

La arquitectura seguida en el proyecto es la propia de A-Frame: entidad- componente - sistema (ECS). Es una arquitectura utilizada habitualmente en el desarrollo de videojuegos o escenas 3D. La arquitectura entidad-componente-sistema sigue la composición sobre la herencia o principio de reutilización del material compuesto, lo que nos ahorra mucho tiempo y esfuerzo a la hora de desarrollar escenas en realidad virtual ya que podemos modelar cada uno de los elementos de nuestra escena como una entidad, por ejemplo, podemos modelar una escena de una ciudad creando una entidad ciudad que esta formada por varios componentes: edificio, carretera, persona, vehículo, árbol... Cada entidad puede estar formada por uno o varios componentes que

van a hacer que se comporte de una manera determinada o tenga la apariencia que deseemos.

Nuestro proyecto se va a dividir en tres grandes componentes. El primero de ellos será el encargado de crear un elemento y posicionarlo dentro de la escena dados unos parámetros. El segundo componente es el encargado de leer de la entrada datos, procesarlos y pasárselos al primer componente para que este cree los elementos y los posicione dentro de la escena. El tercer componente es una prueba de concepto que hemos avanzado para posibles trabajos que se puedan realizar en el futuro y consiste en crear atracción entre los elementos de la escena para que queden más empaquetados todavía.

El desarrollo de nuestro trabajo de fin de grado lo hemos dividido en cuatro sprints, es decir, en cuatro grandes periodos de tiempo en los cuales nos dedicamos a realizar diferentes funcionalidades. En el primer sprint nos centramos en aprender el lenguaje y todo el ecosistema que íbamos a utilizar, en el segundo comenzamos a desarrollar el primer algoritmo, en el tercero perfeccionamos nuestro algoritmo y obtuvimos el algoritmo definitivo de colocación, en el cuarto desarrollamos ese algoritmo definitivo para que fuera genérico y nos funcionara correctamente para cualquier caso de entrada y en el quinto sprint nos dedicamos a realizar la prueba de concepto del algoritmo de atracción y repulsión.

A continuación describiremos los diferentes sprints, entrando al detalle del diseño y el desarrollo de los algoritmos.

4.2. Sprint 0 : Introducción y conocimiento de las tecnologías

El primer sprint fue el encargado de hacer de introducción a las tecnologías y a la metodología de trabajo, en él tuvimos que aprender el funcionamiento de las tecnologías, tanto A-Frame como la manera de trabajar con GitHub y Gitlab. En este sprint nos explicaron la metodología de trabajo que íbamos a seguir y realizamos un tutorial a modo de introducción. En el tutorial pudimos entender más a fondo de qué se trataba A-Frame, pudimos realizar alguna escena en realidad virtual y probar nuestra propia escena en las gafas de realidad virtual del laboratorio. Es una gran motivación ver los resultados con las gafas de realidad virtual y ver la escena nada más terminar de desarrollarlo.

En el primer sprint desarrollamos los ejemplos del tutorial y pudimos tener una visión más amplia de qué es realmente lo que íbamos a hacer. Además de los ejemplos del tutorial también

hicimos una escena la cual trataba de asemejarse a un videojuego en el cual debes ir moviéndote y esquivando los obstáculos presentes en la escena para conseguir llegar a la meta. Fue un juego bastante sencillo pero nos permitió familiarizarnos con las diferentes figuras y con el posicionamiento dentro de la escena de los diferentes elementos, además de manejar el movimiento, el fondo de la escena, las texturas y demás entresijos característicos de A-Frame.

4.3. Sprint I: Exploración con diferentes algoritmos

Siguiendo la metodología ágil mencionada anteriormente cerramos el primer sprint con una reunión en la que acordamos pasar al siguiente sprint dado que habíamos conseguido familiarizarnos lo suficiente con el nuevo framework y ya habíamos conseguido superar todos los desafíos que se presentaron en el tutorial. En nuestro caso las reuniones de final de sprint y de planificación se convertían en la misma ya que aprovechábamos la misma reunión para cerrar el sprint anterior y comenzar el siguiente haciendo una estimación de los tiempos que nos llevaría cada tarea y viendo las tareas a realizar en el nuevo sprint .

Para poder avanzar al sprint II tuvimos que entender cómo funciona A-Frame y construir algún pequeño componente, a continuación, para demostrar lo aprendido, desarrollamos un componente al cual le pasamos un parámetro y nos representa en la escena un cubo perfecto con todas las dimensiones iguales, tanto altura como anchura como profundidad en el centro de la escena. Una vez hecho ese componente, el siguiente paso fue modificarlo para que pudiera admitir varios parámetros de entrada, la primera modificación consistía en que pudiera admitir un ancho, alto y profundidad en un determinado orden y si no le pasamos ningún parámetro nos asignaba por defecto los parámetros que consideramos oportunos en ese momento que fue de 1.

Después de superar el objetivo de desarrollar un componente al que le podemos pasar tres argumentos y nos representa un cubo con esas propiedades de altura, anchura y profundidad, el siguiente paso fue poderle pasar la posición en la que queremos que nos lo represente, pudiendo así colocarlo en cualquier posición de nuestra escena.

Una vez superada esta última tarea ya tenemos un componente al que le podemos pasar como argumento sus propiedades geométricas de altura, anchura y profundidad además de la posición en la que queremos colocar el elemento dentro de la escena. El siguiente paso es diseñar el algoritmo para poder colocar los elementos dentro de la escena de la manera mas em-

paquetada posible. Para ello tuvimos que pensar unos cuantos algoritmos y ponerlos en práctica para ver cuál era el que más se adaptaba a lo que estábamos buscando.

En primer lugar pensamos en colocar un elemento en el centro y en torno a él colocar los siguientes elementos en los ejes de coordenadas x e y, todo esto suponiendo que estamos trabajando sobre una escena de realidad virtual pero nos estamos fijando en un plano de dos dimensiones. Este fue el más sencillo de implementar pero dejaba demasiados espacios y no nos convenció a la hora de ser el elegido.

El procedimiento de este algoritmo es muy simple, nos guardamos la posición más alejada de cada eje y, llevando un control del eje en el que estamos y cual es el siguiente, siempre siguiendo el sentido que siguen las agujas del reloj, colocamos el siguiente elemento en el eje que le corresponda a partir de la posición más alejada, de esta manera nos aseguramos que no se produzcan solapamientos o superposiciones entre los bloques del mismo eje y como los ejes están separados 90º es muy difícil que llegue a haber superposiciones entre bloques de diferentes ejes, aunque no imposible.

Ante la imposibilidad de conocer si se van a producir superposiciones o no dado que puede haber bloques de muy diferente tamaño y si los bloques son de tamaño reducido se generarían demasiados espacios entre los elementos decidimos optar por otras opciones que nos proporcionaban una solución que consideramos más apropiada.

El siguiente algoritmo que pensamos en la posibilidad de implementar consistía en colocar los elementos en círculos concéntricos, colocando un número de elementos ya determinado con anterioridad y teniendo en cuenta el radio de los elementos. El procedimiento en este algoritmo era algo más complejo que el anterior. Vamos a describir brevemente como planeamos el algoritmo, es decir, la fase de diseño y las modificaciones que tuvimos que realizar a medida que nos fueron surgiendo dificultades.

Idealmente el algoritmo debería de ser capaz de saber cuantos elementos van a poder colocarse dentro de la circunferencia en la que estamos representando en cada iteración y, posteriormente colocarlos sin que se solapen unos con otros dentro de la misma circunferencia ni con la circunferencia anterior. Para poder colocarlos en circunferencias concéntricas, tenemos que calcular dos parámetros, el número de elementos que pueden entrar en cada iteración y el radio máximo de los elementos de la circunferencia. Llegados a este punto del diseño del algoritmo, nos dimos cuenta que estaba llegando a un nivel de complejidad bastante elevado y que,

dados unos elementos con dimensiones muy dispares, el resultado no sería para nada óptimo, dejando demasiado hueco sin rellenar entre las circunferencias, por lo que nos concentramos en diseñar otro algoritmo que nos pudiera dejar un mejor resultado para dimensiones variables de los elementos.

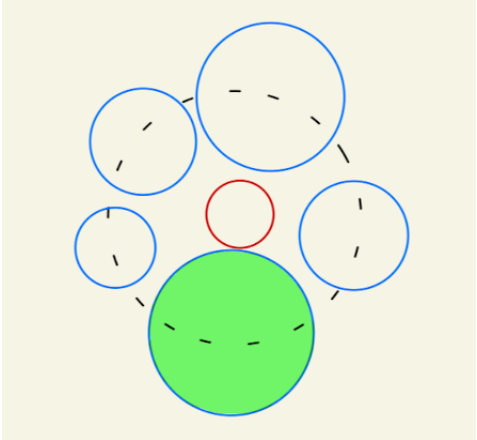


Figura 4.1: Ejemplo en 2D del resultado del algoritmo de circulos concéntricos

En esta figura podemos ver a lo que nos hemos referido anteriormente. El elemento con el contorno en rojo se situará el primero en el centro de la escena. A partir de él tenemos que calcular los radios de los elementos siguientes, escogiendo el mayor radio de los que entren en una circunferencia alrededor suyo. En este caso el radio mayor es el del elemento representado en color verde y debido a ese radio se genera la circunferencia con línea discontinua en la cual tienen que ir situados los centros de los elementos siguientes. A partir de este ejemplo nos dimos cuenta de la complejidad de este algoritmo y que no nos retornaba unos resultados lo suficientemente buenos por lo que decidimos descartarlo.

4.4. Sprint II: Algoritmo seleccionado: primera vuelta al elemento central

Tras la reunión de final de sprint donde decidimos dejar a un lado el diseño del algoritmo de los círculos concéntricos dada la dificultad que estaba alcanzando el algoritmo y la poca eficiencia que el algoritmo nos iba a ofrecer en cuanto al nivel de empaquetamiento de los elementos decidimos dar por cerrado el diseño de ese algoritmo y centrarnos en el diseño de

otro un poco más complejo todavía pero que nos iba a asegurar unos resultados mejores, este algoritmo consistía en representar los elementos dentro de la escena asemejando el proceso de creación al de una espiral, es decir, empezaremos por el centro y seguiremos colocando siguiendo las agujas del reloj, cada elemento lo más cerca posible del centro sin solaparse con ningún otro elemento de la escena y siguiendo la forma de espiral.

El algoritmo puede parecer algo complejo pero fue lo que pensamos que podía tener el mejor resultado y que diera la sensación de empaquetamiento que tanto estamos buscando. El diseño de este algoritmo también se tuvo que dividir en varias fases o tareas, la primera fue la de conseguir colocar el elemento central y, posteriormente, elementos hasta que completen la primera vuelta al elemento central, estando en contacto por todos los laterales con más elementos y todos en contacto con el elemento central. A continuación tuvimos que diseñar el algoritmo para poder colocar elementos en la parte superior, en la parte derecha, en la parte inferior y en la parte izquierda, respectivamente.

Una vez tengamos el algoritmo capaz de representar un elemento en cada uno de los sectores, tenemos que ser capaces de identificar cuando tenemos que cambiar de sector y, por tanto, cambiar la forma de representar nuestros elementos. Una vez tengamos todo eso controlado tendremos un algoritmo capaz de representar todos los elementos que queramos ya que el algoritmo será genérico y será capaz de dar todas las vueltas que queramos en forma de espiral y sin importar el tamaño de las cajas.

Para la primera vuelta es muy sencillo controlar todo ya que sólo nos tenemos que fijar en el elemento central y tomar como referencia su posición, de esta manera, comenzaremos colocando el segundo elemento (el primero es el que está en el centro) haciendo que coincida su esquina inferior izquierda con la esquina superior izquierda del elemento central y calcularemos la posición del centro del segundo elemento para que se cumpla eso, una vez tenemos la posición del elemento se la pasamos al primer componente que habíamos desarrollado el cual dadas las propiedades geométricas del cubo y la posición nos representa un cubo con esas dimensiones en la posición indicada.

Para calcular la posición lo que haremos será pensar que estamos en un plano de dos dimensiones con las coordenadas x e y. Colocamos el elemento central en la posición (0,0), por lo tanto, llegará hasta la mitad de la longitud de su lado en todos los ejes.

Para calcular la posición del nuevo elemento tendremos que calcular su posición en el eje

x como la menor del elemento central, la que esté situada más a la izquierda y por tanto será negativa, más la mitad de su longitud para que así puedan coincidir las esquinas anteriormente citadas.

Para calcular la posición del eje y tenemos que hacer que no se solapen por lo que tendremos que sumar la mitad de la longitud del elemento central mas la mitad de la longitud del elemento que queremos colocar. Las propiedades geométricas del elemento vendrán determinadas por un fichero en formato JSON en el cual vamos a tener los datos geométricos junto con un id de todos los elementos que queremos representar en nuestra escena.

Tras haber descrito el procedimiento para colocar en la escena el segundo elemento, el siguiente elemento irá a continuación, es decir, siguiendo el sentido de las agujas del reloj, a la derecha del anterior, siendo su posición x la posición x del anterior mas la mitad de su longitud y la posición y se calculará de la misma manera que la del segundo elemento, es decir, sumando la mitad de su longitud con la mitad de la longitud del elemento central, consiguiendo así que no se solape el elemento que queremos colocar con el elemento central. Una vez vistos tres elementos de ejemplo podemos describir el algoritmo para colocar los elementos de la parte superior de la siguiente manera:

Para calcular la posición del eje x hay que partir siempre del elemento central, si es el segundo elemento calcularemos su posición x para que su borde izquierdo quede alineado con el borde izquierdo del elemento central y, a partir de este segundo elemento, iremos colocando a su derecha elementos hasta que superemos el borde derecho del elemento central

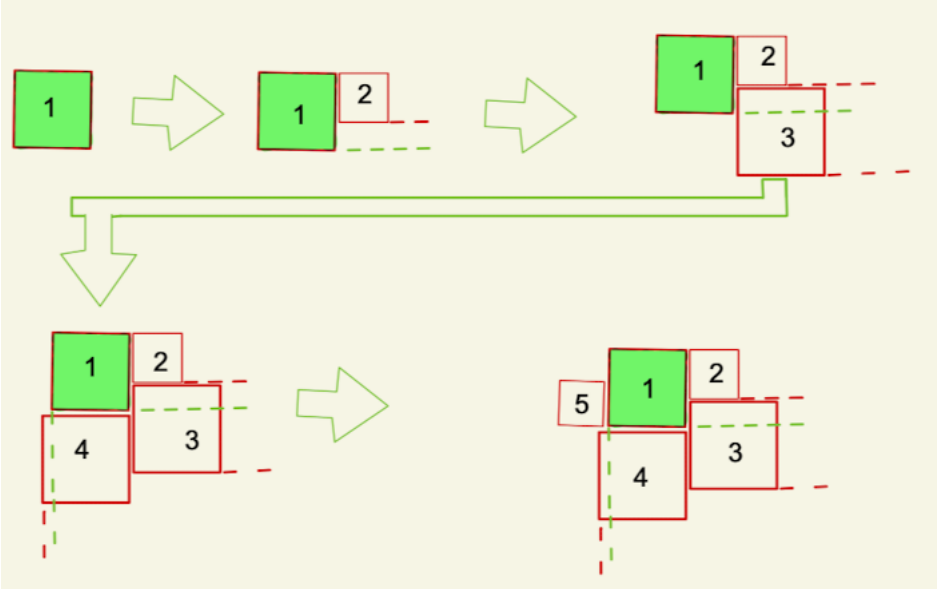


Figura 4.2: Secuencia de colocación de 5 elementos

En la figura anterior podemos ver los cinco pasos de las cinco iteraciones que haremos a la hora de colocar los elementos. en primer lugar colocaremos nuestro elemento central en el centro de la escena. En segundo lugar tendremos que colocar el segundo elemento haciendo coincidir los bordes superiores del primer elemento y el que coloquemos a continuación. En tercer lugar colocaremos el tercer elemento, para saber dónde colocarlo tenemos que fijarnos en las líneas discontinuas tanto la roja como la verde. La verde representa el borde inferior del elemento central y la roja el borde inferior del anterior elemento colocado, en el caso de que el borde el elemento anteriormente colocado supere el borde inferior del elemento central tendremos que colocarlo en la siguiente región, es decir, en la parte inferior del elemento central y si el borde inferior del elemento anteriormente colocado no supera la parte inferior del elemento central colocaremos el siguiente elemento a continuación suya. El caso de la imagen es este último en el que el borde del elemento anterior, señalado con una línea discontinua roja, no supera el borde inferior, señalado con la línea discontinua verde (podemos apreciarlo en la segunda figura, en la que hay dos elementos). Como podemos apreciar en la siguiente figura, la última de la primera fila, el tercer elemento se ha colocado a continuación del segundo elemento y el borde inferior del tercer elemento ya supera el borde inferior del elemento central por lo que el siguiente elemento tendrá que ir en la parte inferior del elemento central. En la primera figura de la segunda línea apreciamos como el cuarto elemento ha sido colocado en la parte

inferior y su borde izquierdo, señalado con una línea discontinua roja, supera el borde izquierdo del elemento central, señalado con una línea discontinua verde, por lo que el siguiente elemento deberá ir en la siguiente zona, la zona de la izquierda del elemento central. Como vemos en la última escena ya hemos colocado el quinto elemento en la parte izquierda y el borde superior del elemento no supera el borde superior del elemento central, por lo que el siguiente elemento lo colocaríamos justo encima del quinto elemento.

Para calcular la posición del eje y de los elementos de la parte superior tenemos que tener en cuenta en todo momento el elemento central, porque siempre van a estar tocando en esta primera iteración la arista inferior de los elementos que coloquemos en la parte superior con la arista superior del elemento central. Para conseguir que se cumpla esta premisa tenemos que calcular la posición y de los elementos de la parte superior en la primera iteración como la mitad de la longitud del elemento central más la mitad de la longitud del elemento que queremos colocar.

Para poder hacer que el algoritmo sea recursivo tenemos que poner una condición para la cual se sigan calculando las posiciones y se coloquen los elementos en la posición calculada en la parte actual o por el contrario tengamos que pasar a la siguiente. Esta condición es que por ejemplo en la parte superior el borde derecho del elemento que estemos colocando sobrepase el borde derecho anterior, es decir, si en la iteración en la que estamos el borde derecho del elemento tiene una posición cuya componente x es mayor que la componente x del borde anterior entonces no debemos colocar más elementos en ese sector y debemos pasar al sector derecho ya que estamos siguiendo el sentido de rotación de las agujas del reloj.

Una vez tenemos diseñada la manera de colocar los elementos de la parte superior tenemos que pensar en la forma de ir rotando la forma de colocar los elementos dentro de nuestra escena, dado este problema surgen dos posibles soluciones, una sería el rotar imaginariamente la figura para estar continuamente colocando en la parte superior de la figura y la segunda es colocar cada una de las partes de la figura, dividiendo la figura en cuatro partes, la parte superior, la parte derecha, la parte inferior y la parte izquierda, siguiendo una rotación como hacen las agujas del reloj y repitiendo hasta que se terminen los elementos a colocar en la escena la rotación de las cuatro partes.

Decidimos adoptar la segunda manera de afrontar el algoritmo por lo tanto tendremos que guardarnos todos los elementos más alejados del centro de cada parte de la figura. De esta manera podemos tener controladas las cuatro posibles zonas de colocación y dónde tenemos

que colocar el siguiente elemento en el caso de que cambiemos de zona.

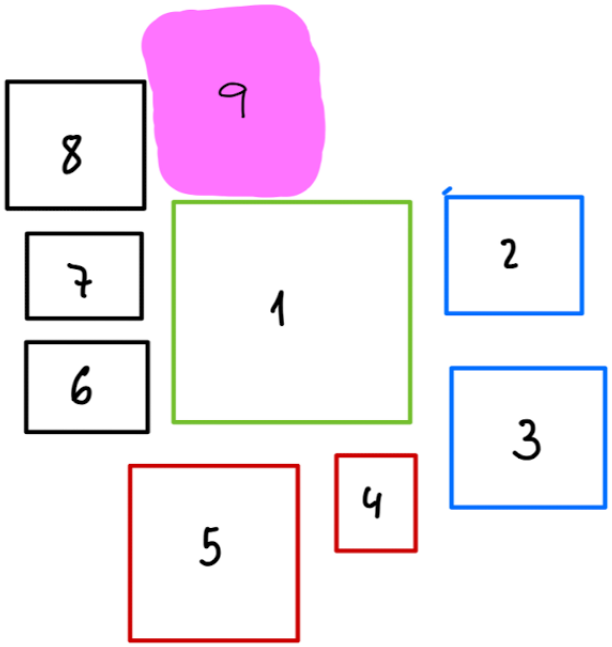


Figura 4.3: Ejemplo de colocación de 8 elementos

En la figura anterior, los números del interior de los elementos significan el orden de posicionamiento, comenzando en el 1 con el elemento central y siguiendo el algoritmo, a continuación del elemento central colocaremos los elementos de la zona lateral derecha, representados en azul. Cuando sobrepasamos el borde inferior del elemento central pasamos a la zona inferior, representada en roja. Cuando sobrepasamos el borde izquierdo del elemento central pasamos a la zona izquierda, representada en negro y cuando los elementos de la zona izquierda sobrepasan el borde superior del elemento central deberemos colocar el elemento noveno en la zona marcada de color rosa.

A continuación describiremos la manera de colocar los elementos en la parte derecha teniendo en cuenta que seguimos dentro de la primera iteración. La base es la misma que la de la parte superior lo único que cambia es la manera de calcular la posición del eje x y la del eje y. Para calcular la posición del eje x tenemos que sumar la mitad de las longitudes del elemento central y del elemento a colocar. La posición del eje y tenemos que calcularla de manera similar a la x de la parte superior y es que si es el primer elemento que vamos a colocar en esta zona tenemos que colocarlo de manera que nos queden alineadas las aristas superiores del elemento central y del elemento que queremos colocar, de esta manera nos evitamos superposiciones en-

tre el último elemento de la zona superior y la zona positiva del eje de las abscisas, es decir, la zona colocada a la derecha del elemento central.

A continuación pasaremos a la zona inferior si el borde inferior del elemento anterior ha sobrepasado el borde inferior del elemento central. Para comenzar a colocar los elementos de la zona inferior será muy similar a los de la zona superior pero invirtiendo el sentido de los ejes, es decir, iremos hacia la izquierda y hacia abajo. Empezaremos colocando el primer elemento de la zona inferior haciendo coincidir los bordes derechos del elemento central con el elemento actual que vamos a colocar. La posición del eje y la calcularemos sumando la mitad de las longitudes de los elementos central y actual y la posición del eje x la calcularemos de tal manera que vayamos hacia la izquierda desde el primer elemento, sumando la mitad de la longitud del elemento anterior con la del elemento actual, pero desplazándonos hacia la izquierda por lo que hay que restar las longitudes para que la posición x del elemento actual salga menor que la del elemento anterior.

Una vez sobrepasemos el borde izquierdo del elemento central tenemos que pasar al sector negativo del eje de las abscisas, es decir, la zona izquierda del elemento central, siendo muy parecida a la zona derecha la única diferencia es que tenemos que invertir los signos. la posición x se calculará restando las mitades de los elementos central y actual y la posición y se calcula sumando la mitad de las longitudes del elemento anterior y el actual salvo que sea el primer elemento, en ese caso tenemos que hacer que coincida la posición y del borde inferior con la posición del eje y del borde inferior del elemento central.

Una vez el borde superior del elemento del sector izquierdo supere el borde superior del elemento central habríamos completado una iteración sobre el elemento central y sólo nos quedaría repetir el proceso pero sin tener como referencia el elemento central si no que varíe en función de la posición en la que vayamos a colocar el elemento en cuestión.

4.5. Sprint III: Algoritmo definitivo: después de la primera vuelta

Después de haber conseguido superar el problema de dar una vuelta alrededor del elemento central surge un nuevo reto que tenemos que afrontar, el reto consiste en colocar los elementos siguiendo el orden lógico que seguiría una espiral, es decir, justo a continuación del anterior

pero sin solaparse con ningún otro elemento de la escena y siempre siguiendo el sentido que marcan las agujas del reloj. Siempre seguiremos el mismo ciclo de zonas descritas en el proceso por el cual rodeábamos al elemento central.

Para afrontar este problema tenemos que primero imaginarnos como queremos que quede y hacernos un pequeño esquema para luego poder llevarlo a la práctica. Para el diseño de este otro algoritmo seguimos los mismos principios que para el que rodeaba al elemento central, es decir, tendremos guardadas tanto las posiciones como las longitudes de los elementos más alejados de las cuatro zonas, por tanto cuanto más grande sea nuestra figura, más elementos tendremos guardados.

Teniendo guardados los elementos más alejados de cada una de las zonas ya tenemos una visión de dónde tenemos que colocar los elementos y tan solo tenemos que encontrar el elemento que nos sirva de referencia. Dependiendo de dónde hayamos colocado el elemento anterior nos corresponderá colocar el siguiente en una posición determinada y el elemento que tenga inmediatamente debajo si estamos en la zona superior, a la izquierda si estamos en la zona derecha, arriba si estamos en la zona inferior o en la derecha si estamos en la zona izquierda será el elemento que nos sirva de referencia para calcular la posición y que no se solape con ningún otro elemento.

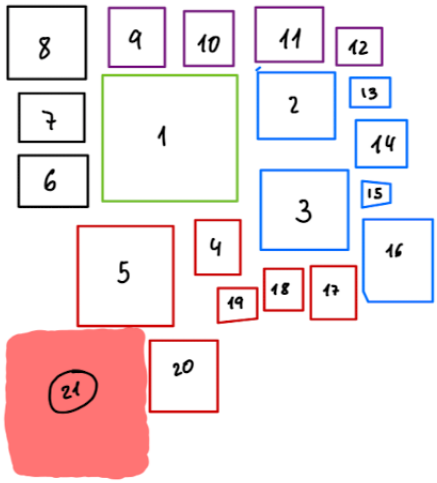


Figura 4.4: Ejemplo de colocación de 20 elementos

En la figura anterior podemos observar un ejemplo de cómo continuaría el algoritmo una vez completada la primera vuelta y, marcado en color rojo, el lugar donde debería ir el siguiente elemento, siguiendo la lógica de nuestro algoritmo.

Una vez tenemos el diseño del algoritmo claro, lo único que cambia con respecto al que toma como referencia al elemento central es que ya no tenemos un elemento de referencia fijo si no que tenemos que buscar cuál es el nuevo elemento de referencia dependiendo de la zona en la que nos encontremos y del elemento anteriormente colocado.

Una vez tengamos colocado el elemento tenemos que actualizar los elementos más alejados de la zona, eliminando el que nos había servido de referencia y colocando en esos elementos que en un futuro será de referencia el nuevo elemento que hemos colocado, colocaremos su posición final y la longitud de sus lados para poder hacer una colocación correcta de los siguientes elementos en las posteriores iteraciones.

Como os podeis imaginar comenzaremos por la parte superior justo encima del primer elemento que colocamos después del elemento central y, aplicando el algoritmo de colocación explicado en el primer paso para la zona superior iremos colocando elementos a la derecha del anterior hasta que sobrepasen el borde derecho del último elemento guardado para esa zona.

Una vez coloquemos cada elemento tenemos que actualizarlo en la zona apropiada para que sirva como referencia para futuras iteraciones. Una vez consigamos completar la zona superior continuaremos con la derecha, a continuación la inferior y por último la izquierda, siempre teniendo en cuenta los algoritmos descritos para cada una de las zonas en el paso uno.

Una vez tengamos completada la zona izquierda pasaremos de nuevo a la parte superior y, dado que se actualizan los elementos de referencia automáticamente al colocarse un nuevo elemento, ya tenemos nuestro algoritmo iterativo, el cual nos sirve para colocar n elementos dentro de la escena.

4.6. Sprint IV: Algoritmo de atracción-repulsión

Ya hemos completado la mayoría de los objetivos que nos propusimos al principio pero aún nos queda por hacer una prueba de concepto para que sea la semilla de nuevos trabajos que puedan surgir a partir de ella. Esta semilla consistirá en el diseño y el desarrollo de un algoritmo de atracción y repulsión para algunos elementos dentro de la escena hacia el elemento central.

Para el desarrollo del algoritmo tenemos que tener ya posicionados los elementos dentro de la escena y los tenemos que tener identificados y etiquetados. Para conseguir tener los elementos de la escena localizados y etiquetados lo hacemos a medida que los vamos colocando. Cuando

colocamos un elemento en la escena por el método desarrollado en los anteriores sprints, lo vamos guardando en un array con un identificador único junto con su posición para posteriormente poder acceder a todos sus datos y hacer los cálculos necesarios.

El proceso de guardar y obtener los datos de una manera sencilla y rápida es algo fundamental ya que es uno de nuestros requisitos el no ocupar mucho tiempo el cargar la escena.

Una vez tenemos colocados todos los elementos en la escena y etiquetados correctamente obtendremos una escena normal pero el objetivo de esta prueba de concepto es hacer que los elementos se vayan acercando al elemento central, para ello iremos calculando los vectores que unen los centros de los diferentes elementos con el elemento central y los iremos haciendo cada vez más pequeños, cuidando de que no se produzca ningún tipo de solapamiento ni entre los diferentes elementos ni entre el elemento en cuestión y el elemento central.

Para que no ocurra solapamiento calcularemos las distancias entre los centros de todos los elementos y comprobaremos que es mayor a la suma de sus radios. Si se cumple que la distancia entre los centros es mayor a la suma de los radios entonces moveremos los elementos a la nueva posición y si no se cumple entonces ya hemos llegado al límite y no los moveremos pues se produciría un solapamiento.

Con este algoritmo conseguimos que los elementos en la escena queden lo más compactados posible, siempre teniendo en cuenta que los podemos acercar más o menos en cada iteración pero nos va a afectar inversamente al tiempo de cómputo, es decir, cuanto menor sea la distancia que empleemos para acercar los elementos en la escena, mayor será el tiempo que tardaremos en calcular todas las iteraciones y el resultado final de la escena. Por ello hay que llegar a un compromiso para que sea lo suficientemente válido y que la distancia entre los elementos sea lo menor posible pero sin comprometer el tiempo de cómputo.

Capítulo 5

Resultados

A medida que hemos ido avanzando en el proyecto hemos ido cumpliendo una serie de objetivos, mencionados en el Capítulo 2. A continuación describiremos los resultados obtenidos en cada uno de los objetivos planteados.

5.1. Comprender y aprender a utilizar A-Frame

Antes de ponernos propiamente a diseñar y desarrollar el componente final del proyecto, el primer paso fue conocer la tecnología, estudiar las particularidades que tiene el framework y desarrollar algunas escenas de prueba muy simples. También tendremos que leer bastante documentación para entender bien con qué tecnologías estamos tratando y qué es posible hacer y qué no es posible. Además tendremos que hacer bastantes pruebas de concepto para verificar que todo lo que hayamos leído sabemos aplicarlo posteriormente por nuestra cuenta para conseguir las figuras que queremos en las posiciones que queremos.

Este objetivo fue uno de los primeros que tuvimos que cumplir para poder continuar con los siguientes, al comenzar el proyecto tuvimos que entender el funcionamiento de A-Frame, entender con qué tecnologías teníamos que trabajar y, posteriormente, poner todo lo aprendido en práctica desarrollando alguna escena que demostrara que todo lo que habíamos aprendido con los tutoriales y por nuestra cuenta lo habíamos entendido y sabíamos ponerlo en práctica. En primer lugar construí una escena formada por varias figuras, todas las posibles dentro de A-Frame, fue una escena muy sencilla pero ya fuimos entendiendo cómo se podían colocar los elementos dentro de una escena.

La segunda escena que desarrollamos fue una especie de juego muy básico formado por planos para formar el suelo y distintas formas geométricas como cilindros y cubos para construir todos los elementos. El juego era un circuito en el que el objetivo era llegar al final sin caerse al vacío, que ocurría cuando te salías de los planos que formaban el suelo.

Para trabajar con A-Frame necesitamos desarrollar algún componente al que se le pasan diferentes parámetros desde el HTML y, en base a esos parámetros, podemos observar comportamientos diferentes dentro de la escena. Para comenzar desarrollaremos un componente muy básico al que le pasaremos como argumento un fichero JSON, formado por un objeto con las propiedades de altura, anchura y profundidad, y nos coloque un elemento con forma de cubo con las propiedades de altura, anchura y profundidad extraídas del fichero JSON.

La escena que nos sirvió para conocer el lenguaje y asimilar de una manera práctica los conceptos y las escenas del tutorial se puede ver en la siguiente dirección web: https://villalba5.github.io/aframe/ph
02/pruebajuego.html

A continuación adjunto dos capturas para describir de forma gráfica su funcionamiento:

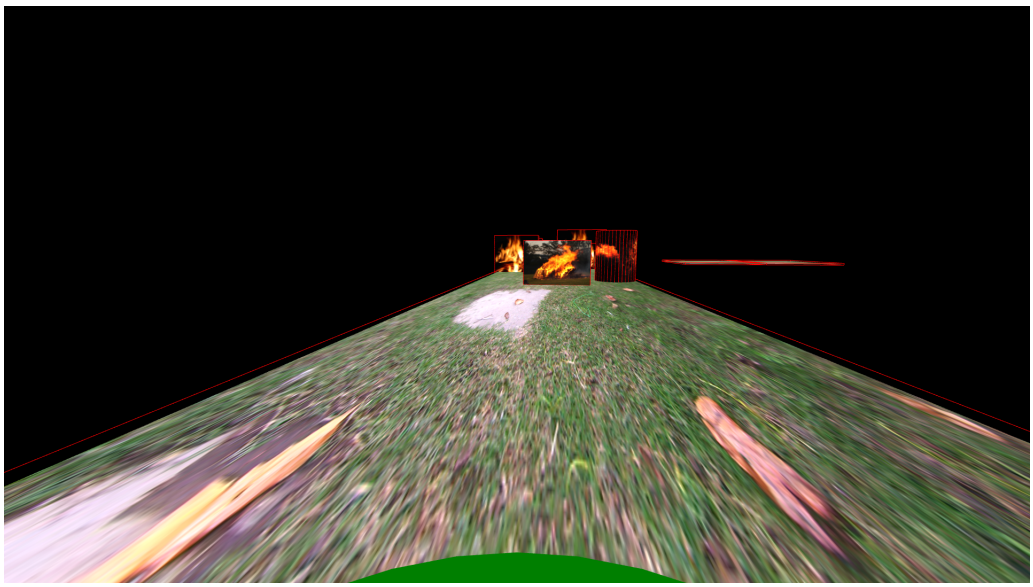


Figura 5.1: Captura de pantalla de la primera escena creada en A-Frame desde el navegador

En esta captura de pantalla podemos observar cómo se vería la escena si la abriéramos con las gafas de realidad virtual. También podemos probar el funcionamiento desde el navegador utilizando las flechas de dirección para movernos por la escena y el ratón para mover la cámara. El cilindro verde representa nuestra posición y tendremos diferentes obstáculos a superar.

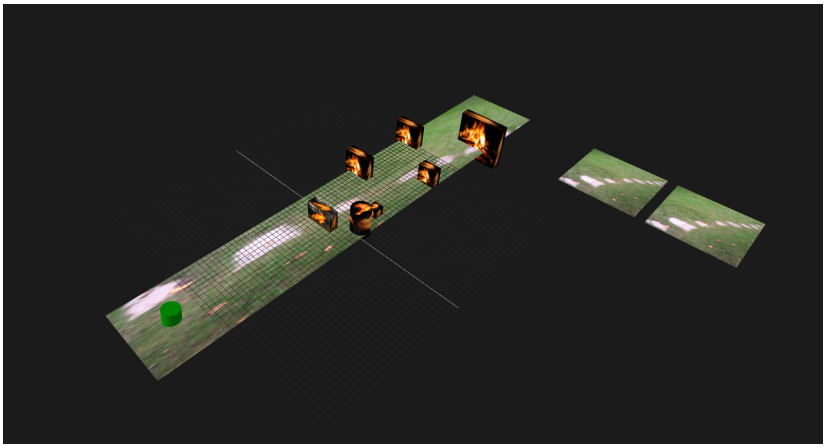


Figura 5.2: Captura de pantalla de la primera escena creada en A-Frame desde el inspector

En esta segunda captura tenemos una visión mucho mejor de la escena en sí. En la captura de pantalla podemos ver el cilindro verde al inicio de la escena, situado sobre un plano con forma de rectángulo y con textura de césped que incluye varios elementos con textura de fuego que simbolizan los obstáculos. Estos obstáculos tienen textura además de sonido al desplazarse. Los obstáculos irán de derecha a izquierda para impedirnos el paso y nosotros tendremos que conseguir pasar sin que nos tiren al vacío. Una vez conseguimos superar los obstáculos llegamos a una plataforma movediza, un plano cuadrado con textura de césped, que se mueve desde el plano cuadrado que simboliza la meta hasta el plano rectangular donde empezamos. Para llegar a la meta tendremos que subir en el plano movedizo y bajarnos en el último plano.

Como se puede apreciar es una escena bastante simple pero la intención era familiarizarse con la creación de escenas y experimentar con diferentes propiedades como las texturas, sonidos y movimiento de los elementos ya que posteriormente utilizaremos algunas de estas propiedades.

Con esta escena y habiendo asimilado los principios y los fundamentos del lenguaje conseguimos superar este objetivo y pudimos ponernos manos a la obra con lo que sería ya parte de nuestro trabajo final.

5.2. Diseñar y desarrollar un componente de A-Frame

Otro de los objetivos que conseguimos superar fue el de diseñar y desarrollar un componente en A-Frame. Al principio parecía algo complicado pero tras entender el funcionamiento es la

mejor manera para hacer un código iterativo sin repetir código, pudiendo usar tu componente en varios proyectos sin necesidad de cambiar nada. El primer componente que diseñamos y desarrollamos fue el encargado de formar un elemento geométrico el cual pasamos como parámetro al componente acompañado de las correspondientes propiedades necesarias para la representación y colocarlo en la posición pasada como parámetro igualmente. De esta forma por ejemplo podemos decirle a nuestro componente que nos coloque un cubo de 4m de ancho, 6m de alto y 3m de profudo en la posición (4,1,0), tomando como referencia los ejes cartesianos (x, y, z).

El segundo componente que conseguimos desarrollar era el encargado de calcular la posición de los elementos dependiendo de un parámetro de entrada para colocarlos en espiral, en circulos concéntricos, de manera aleatoria por la escena o lo más cercanos al centro posible. Este fue el algoritmo más complicado de diseñar y desarrollar y en el cual en el proceso de diseño tuvimos que resolver un mayor número de problemas. Finalmente el resultado fue bastante vistoso y quedamos orgullosos de él.

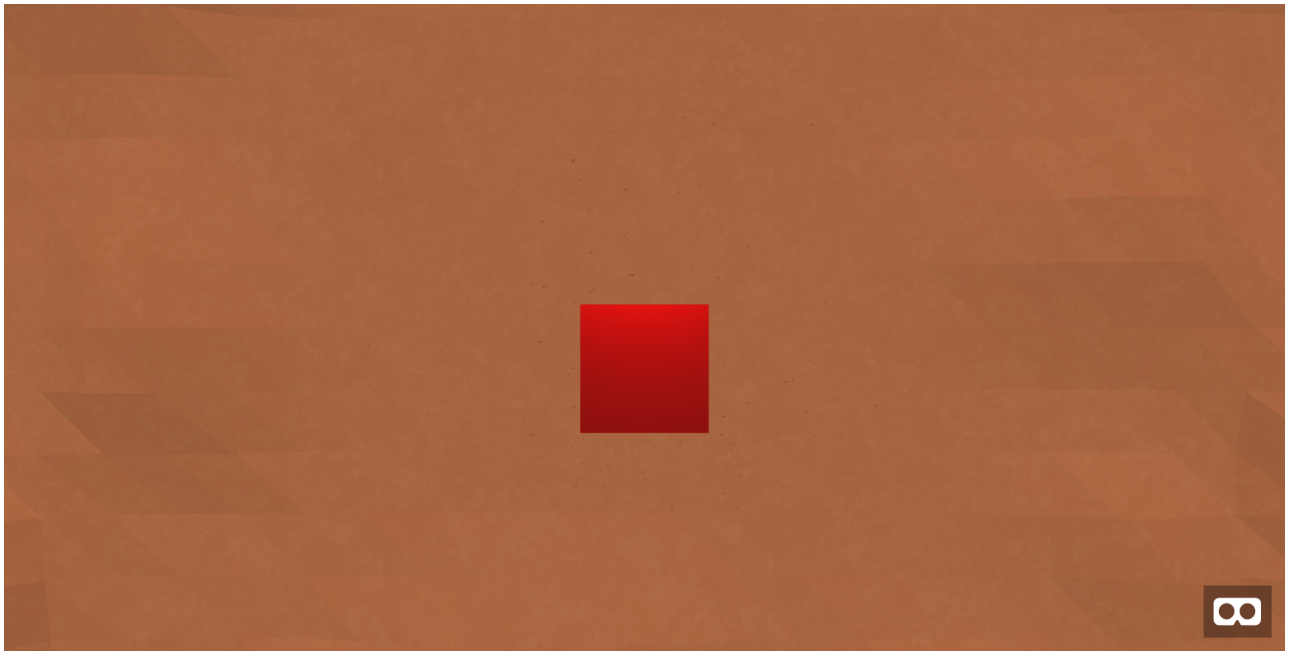


Figura 5.3: Captura de pantalla de la escena con un elemento

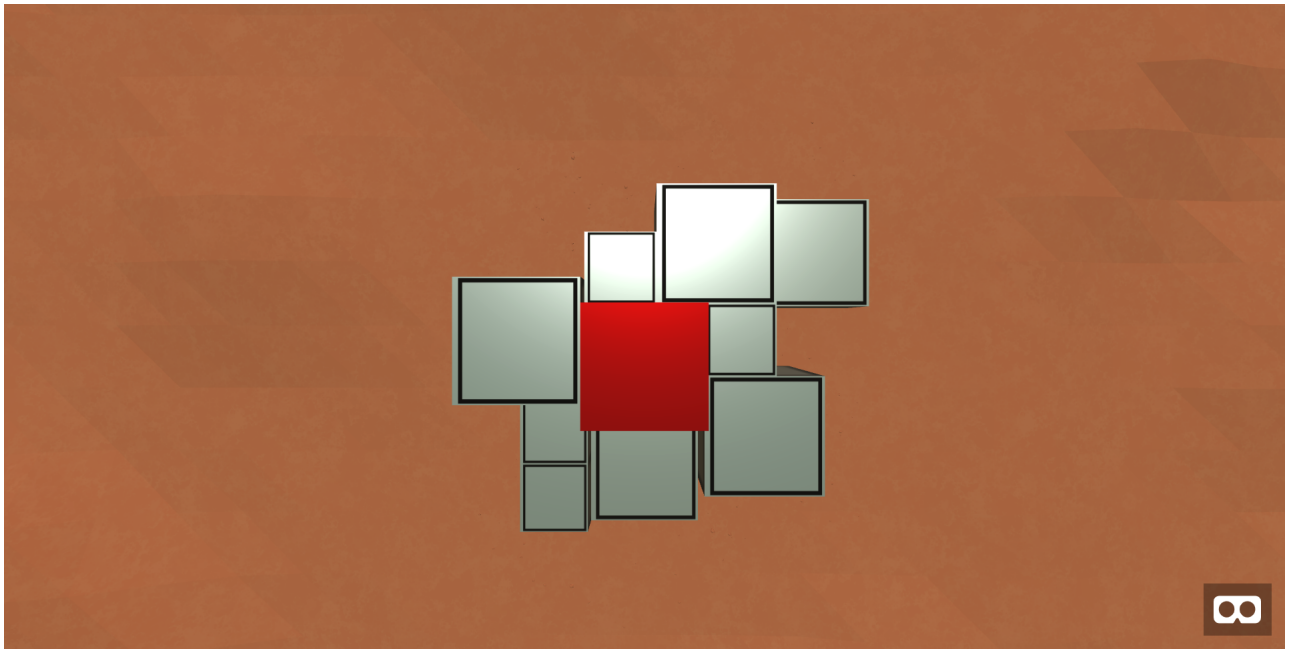


Figura 5.4: Captura de pantalla de la escena con diez elementos

5.3. Probar con diferentes formas geométricas

Como hemos expuesto en el anterior punto, el primer componente que diseñamos y desarrollamos es capaz de representar diferentes formas geométricas dentro de la escena, resolviendo así uno de los objetivos que nos habíamos propuesto. Para resolver este problema decidimos que las formas geométricas más adaptables a la representación serían los tetraedros, ya sean de base cuadrada o rectangular, y los cilindros, asemejando la vista final a la de la esfera ya que tenemos una vista desde arriba de la escena y el contorno de un cilindro y de una esfera será igualmente un círculo.

En la siguiente captura podemos observar la manera de colocarse diez elementos con forma de cilindros. El primer elemento está representado en rojo y los demás elementos se colocarán siguiendo la forma de espiral tratando de no solaparse con ningún otro elemento en la escena según los algoritmos anteriormente explicados. Los cilindros tienen una textura especial para facilitar la detección de los bordes y, aunque tengan diferentes alturas y parezca que hay superposiciones es debido a la perspectiva ya que si nos acercamos al punto donde aparentemente se superponen están perfectamente colindantes.

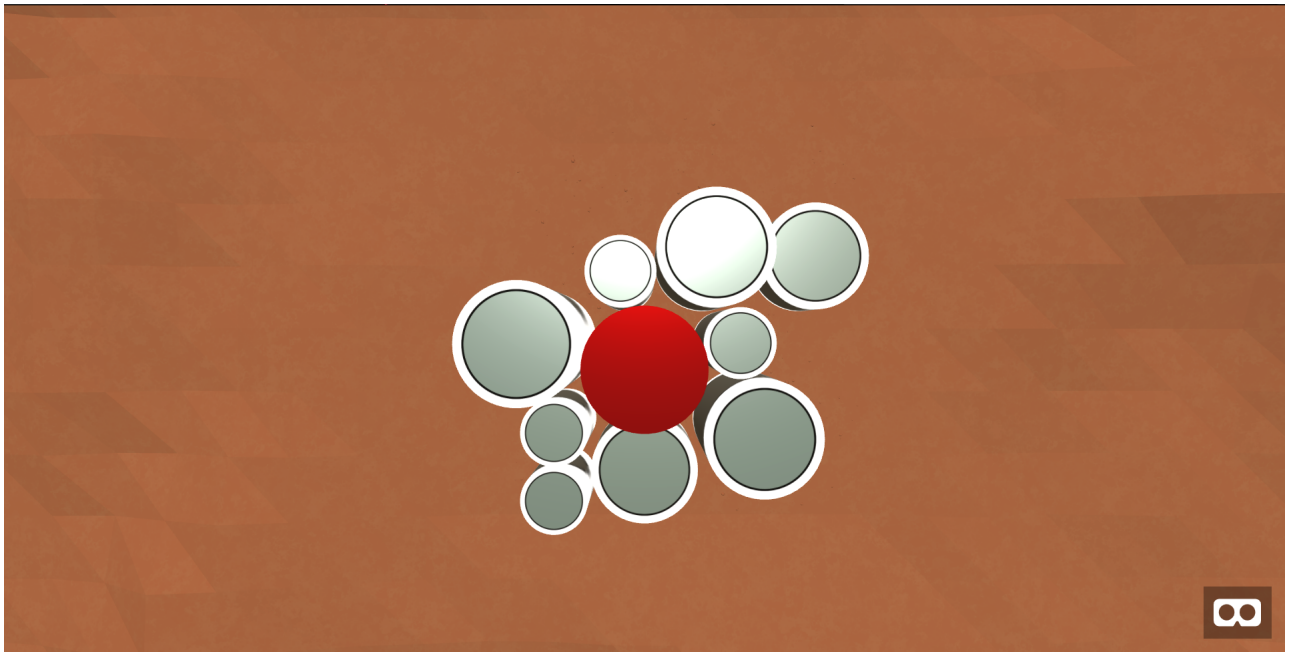


Figura 5.5: Captura de pantalla de la escena con diez cilindros

5.4. Encontrar la forma deseada de colocación

Para resolver el problema de la forma de colocación tuvimos que hacer varios ensayos de prueba y error, probando con diferentes algoritmos ya existentes pero tratando de adaptarlos a nuestro problema, para ello desarrollamos desde cero nuestra solución y observamos el resultado final, decidiendo si el algoritmo era lo suficientemente bueno para resolver el problema o debíamos buscar otra alternativa. Finalmente tras varios intentos decidimos adoptar el algoritmo que coloca los elementos en la escena en forma de espiral tratando de colocar los elementos lo más cercanos al centro posible.

Para representar la consecución de este objetivo adjuntaré dos capturas de pantalla, cada una de la captura será una escena con un número aleatoriamente grande para comprobar la correcta colocación de los elementos.

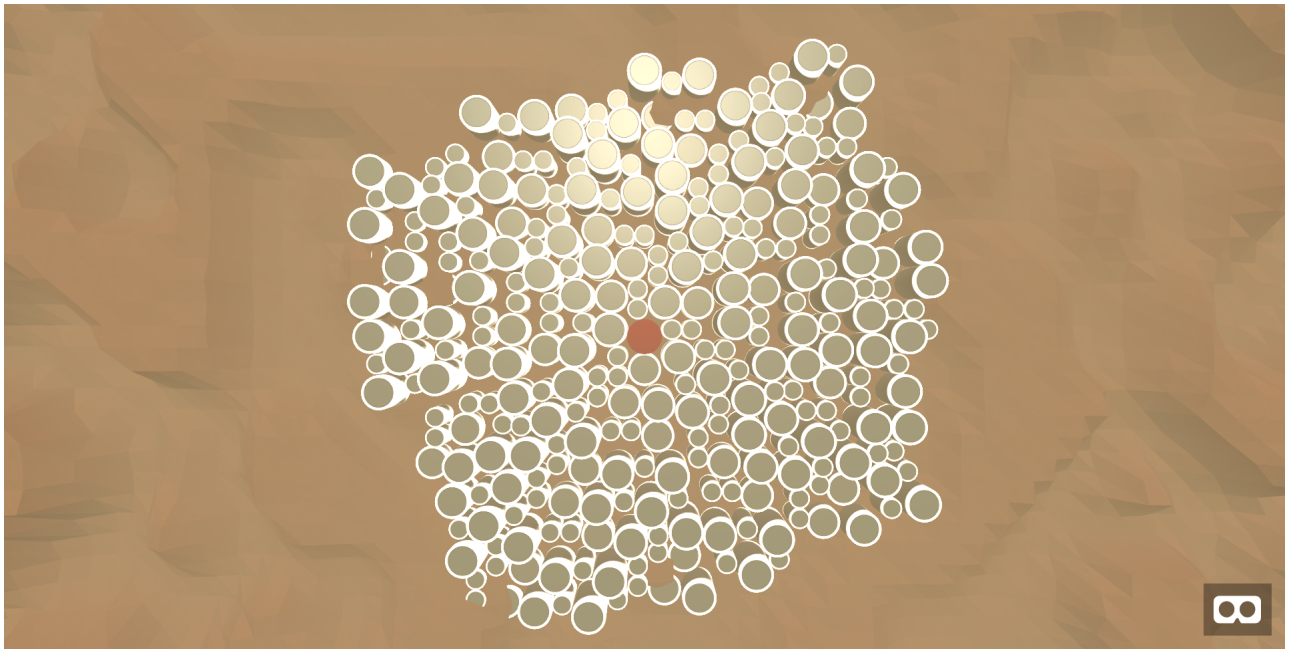


Figura 5.6: Captura de pantalla de la escena con cilindros

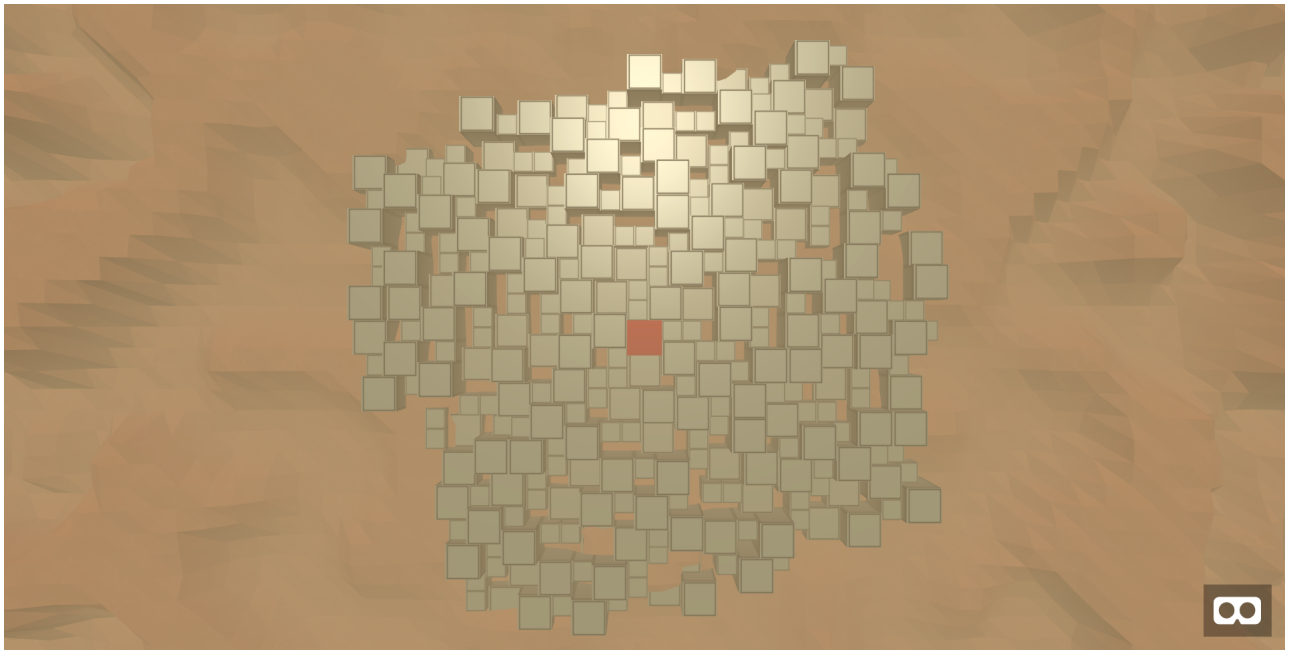


Figura 5.7: Captura de pantalla de la escena con cubos

5.5. Algoritmo de atracción y repulsión

Para dar por concluido el algoritmo de atracción y repulsión dejamos preparada una serie de dos escenas para que sirvan de precedente para futuros trabajos. En estas escenas podemos observar como un elemento central atrae otro elemento y cómo un elemento central atrae dos elementos más.

Con respecto a la prueba de concepto que hicimos primero lo desarrollamos para dos elementos en la escena que se colocaban siguiendo el algoritmo de empaquetamiento que seleccionamos en los primeros sprints. Una vez colocados los dos elementos nuestro objetivo es acercarlos lo máximo posible, siempre teniendo en cuenta que no se pueden solapar. Para ello establecimos una distancia mínima que era la que nos iba a acercar un elemento a otro en cada iteración y siempre teniendo en cuenta esa distancia mínima a recorrer íbamos acercando los elementos hasta que la distancia entre ellos fuera menor que la distancia mínima. Para establecer la distancia mínima tenemos que llegar a un compromiso entre lo que está suficientemente cerca y el tiempo de ejecución ya que cuanto menor sea esa distancia, más iteraciones tendremos que hacer en el algoritmo y eso puede penalizarnos en tiempos muy largos de espera cuando tengamos muchos elementos en la escena.

Una vez acercamos dos elementos en la escena uno a otro tuvimos que tener en cuenta que los elementos con mayor volumen tienen que atraer a los elementos con menor volumen y no al revés por lo que tuvimos que crear otro algoritmo que se encargara de hacer proporcional la distancia que tienen que acercarse respecto a su volumen, de ese modo podemos hacer que se asemeje a la gravitación, en la que los cuerpos con un mayor volumen atraen con mayor fuerza a los de menor volumen que a otros con mayor volumen que ellos.

5.6. Conclusiones del desarrollo del proyecto

Para concluir este gran apartado del desarrollo del proyecto cabe destacar varios puntos, en primer lugar, tras completar todo el desarrollo del proyecto, hemos conseguido un algoritmo capaz de mantener el orden de una lista ordenada para poder visualizarlo de una mejor manera en realidad virtual tanto en el navegador como en las gafas de realidad virtual.

Por otro lado si la lista que se pasa como argumento al componente no está ordenada no pasa

nada pues nos representará los datos manteniendo el orden de entrada.Además de la exploración con los diferentes algoritmos y el posterior desarrollo del algoritmo de empaquetamiento, hemos conseguido desarrollar una prueba de concepto que va a servir a futuros trabajos de base para poder mejorar todavía mucho más el resultado final de la visualización de los datos.

Los elementos que vemos en las escenas son elementos del DOM por lo que hemos tenido que crear primero esas entidades con los valores y características que se ajustaban a los parámetros de entrada y finalmente hemos tenido que meter esas entidades en el DOM para poder visualizar las escenas correctamente con todos nuestros elementos y las propiedades que se le pasaron como argumento.

Capítulo 6

Conclusiones

El objetivo de este trabajo de fin de grado era aprender, sobre todo en lo relacionado con el proceso de investigación, exploración, diseño y el posterior desarrollo que tiene un proyecto de software, para ello hemos utilizado metodologías ágiles y un control de versiones para mantener un control del código.

Este proyecto de fin de grado nos ha enseñado muchas cosas y hemos podido aprender mucho de él ya que se asemeja mucho todo el flujo de trabajo a lo que posteriormente voy a encontrarme en cualquier empresa de software. Ha sido muy gratificante ver como los resultados de nuestro esfuerzo se han plasmado en las escenas y el poder ver los resultados desde el navegador y desde las gafas de realidad virtual es todavía más gratificante.

Además de todo lo anteriormente citado, hemos aprendido a trabajar en realidad virtual, algo que parecía inpensable y hemos conseguido desarrollar algoritmos que pueden ayudar a quien lo necesite a conseguir un mejor resultado en su proyecto o sus desarrollos.

6.1. Consecución de objetivos

En primer lugar cabe destacar que conseguimos superar todos los objetivos importantes que nos propusimos antes de comenzar el proyecto, hemos conseguido hacer un trabajo de exploración bastante importante, topándonos en muchos casos con problemas a los que les hemos tenido que dar una solución eficiente y posteriormente poner en práctica esas soluciones para que quedara un resultado vistoso. Además de conseguir explorar los diferentes algoritmos de empaquetamiento que hicimos a lo largo de las primeras etapas luego supimos hacerlo lo

suficientemente modulable para que le podamos decir con qué figuras queremos que nos la represente bien sea con tetraedros o con cilindros.

A continuación describiremos cómo hicimos para conseguir superar los objetivos que nos planteamos además de los obstáculos que se nos interpusieron en el camino, haciendo hincapié en el por qué de cada una de las decisiones que nos llevaron a conseguir solventar los problemas y por consiguiente a superar los objetivos.

6.1.1. Entender el funcionamiento de A-Frame

El primer objetivo que logramos superar fue el de entender el framework de A-Frame, para ello seguimos los pasos de un tutorial [12], fue bastante más asequible la asimilación que si lo hubiéramos hecho por nuestra cuenta, en el tutorial vimos todas las posibilidades que teníamos a la hora de modelar una escena en realidad virtual, así como la creación de componentes, el manejo de las animaciones, etc.

6.1.2. Primer componente en A-Frame

Este fue el siguiente paso una vez revisamos toda la documentación de A-Frame en su sitio web oficial y nos hicimos el tutorial mencionado en el apartado anterior. El primer componente que desarrollamos fue bastante básico pero nos sirvió para entender el funcionamiento de los componentes, la manera en la que les podemos pasar parámetros como argumentos desde el html y también la estructura de ficheros que tenemos que tener para tener todo organizado y saber en cada momento en qué parte del código nos encontramos.

Para el primer componente desarrollamos uno que admitía como parámetros las propiedades geométricas de un cubo y la posición en la que queremos que se muestre dentro de la escena y el componente nos representaba el cubo con esas propiedades geométricas en la posición determinada.

6.1.3. Diseño de algoritmos

El objetivo de diseñar algoritmos fue un auténtico desafío para nosotros ya que en mi caso había diseñado algoritmos anteriormente pero en solitario, sin embargo esta vez fue algo

diferente ya que surgieron diversas ideas y tuvimos que ir viendo los beneficios que nos aportaban cada uno de los diseños propuestos y las desventajas que tenían para finalmente centrarnos en el definitivo. Una vez nos centramos en el algoritmo definitivo no todo fue sencillo ya que tuvimos que sortear varios obstáculos en forma de problemas inesperados que nos fueron surgiendo como algunas intersecciones entre los bloques que no habíamos previsto y debido a estos problemas intermedios y mediante las reuniones de seguimiento del sprint, tuvimos que ir modificando el algoritmo hasta que finalmente obtuvimos el resultado esperado.

Para el diseño del algoritmo utilizamos la idea de colocarlos en forma de espiral de forma que dividimos nuestra escena en cuatro posibles zonas donde podemos colocar los elementos que forman la escena. Llevando un seguimiento de la posición de cada uno de los elementos de la escena y en especial del último elemento que acabábamos de colocar, seguimos la forma de espiral en sentido horario, colocando en cada iteración el elemento lo más cerca posible del centro.

6.1.4. Encontrar la forma deseada de colocación

este objetivo fue una parte importante del diseño del algoritmo y también del desarrollo, por eso decidimos marcarlo como un objetivo diferenciado de estos dos otros, la importancia de encontrar la forma deseada de colocación es extrema, es la diferencia entre un resultado vulgar y un resultado vistoso, aun siguiendo la misma metodología de trabajo, trabajar en el diseño de los algoritmos y ponernos manos a la obra desarrollando los componentes necesarios y los algoritmos que los forman, el resultado final es muy diferente y, de hecho, tuvimos que diseñar y desarrollar varias formas de colocación para determinar cual era la que se adaptaba más a lo que realmente estábamos buscando.

6.2. Planificación temporal

El comienzo de este proyecto tuvo lugar en Septiembre de 2019, por lo que la duración en tiempo natural oscilará entre los 12 o 13 meses. El nivel de esfuerzo inicialmente no fue exigente, siendo principalmente los fines de semana cuando podía dedicarle algo de tiempo al proyecto.

A medida que fui aprobando el resto de asignaturas, el nivel de esfuerzo fue aumentando

Objetivos	Mes de inicio	Mes de finalización
Primeros pasos	Septiembre 2019	Octubre 2019
Sprint I	Noviembre 2019	Marzo 2020
Sprint II	Abril 2020	Junio 2020
Sprint III	Junio 2020	Agosto 2020
Sprint IV	Septiembre 2020	Septiembre 2020

Cuadro 6.1: Tabla de resumen de la duración de cada objetivo.

y pude estar algún día entre semana dedicado al proyecto pero nunca he tenido la dedicación completa debido a que lo he tenido que compaginar con otras asignaturas y con trabajo. Entre medias de estos meses he estado periodos de tiempo sin poder dedicarle apenas tiempo al proyecto debido a la alta carga de trabajo procedente de mi trabajo o de los exámenes de otras asignaturas.

Como podemos ver en el contenido de la tabla de la parte superior de la página, estuvimos durante los meses de Septiembre y Octubre de 2019 con los primeros pasos. Posteriormente a superar el objetivo avanzamos al sprint I, para el cual dedicamos los meses comprendidos entre Noviembre de 2019 y Marzo de 2020, podemos observar que fue el periodo más largo ya que fue el proceso de creación del componente y hubo que ajustar muchos detalles. Para finalizar el sprint II en los meses comprendidos entre Abril y Junio de 2020. Estuvimos dedicados al Sprint III entre Junio y Agosto de 2020. Para la prueba de concepto estuvimos durante el mes de Septiembre. Entre medias estuvimos escribiendo la memoria y preparando todo lo necesario para la defensa del proyecto.

6.3. Aplicación de lo aprendido

Para la resolución del objetivo principal del proyecto de fin de grado nos han sido de gran utilidad todas aquellas asignaturas que tenían alguna relación con la programación ya que sin esos principios básicos y sin tener la mente preparada hubiera sido imposible tanto el diseño como el desarrollo de los algoritmos y de los componentes desde cero, por otro lado también hemos necesitado tener conocimientos básicos de física y de matemáticas en general por lo que

en este proyecto nos hemos ayudado de los conceptos básicos de muchas de las asignaturas que hemos cursado durante el grado.

6.3.1. Asignaturas relacionadas con la programaación

1. FUNDAMENTOS DE LA PROGRAMACION
2. PROGRAMACION DE SISTEMAS DE TELECOMUNICACION
3. INGENIERIA DE SISTEMAS TELEMATICOS
4. SERVICIOS Y APLICACIONES TELEMATICAS
5. SISTEMAS OPERATIVOS
6. INGENIERIA DE SISTEMAS DE INFORMACION
7. DESARROLLO DE APLICACIONES TELEMATICAS
8. SERVICIOS Y APLICACIONES TELEMATICAS

Todas estas asignaturas nos sirvieron para ir formando la lógica de programación con los principios básicos de la programación necesarios para resolver problemas. Cada una de ellas nos aportó su granito de arena con lenguajes de programación distintos, distintos puntos de vista de los profesores a la hora de resolver los problemas y distintas metodologías de trabajo llevadas a cabo para poder superar las asignaturas, pero todas ellas aportaron para que pudiéramos conseguir llegar al objetivo final y conseguir terminar el proyecto.

6.3.2. Asignaturas relacionadas con las matemáticas o física

1. CALCULO Y ANALISIS DIFERENCIAL
2. ALGEBRA LINEAL Y MATEMATICA DISCRETA
3. SEÑALES Y SISTEMAS
4. FUNDAMENTOS FISICOS DE LA TELECOMUNICACION
5. RADIACION Y PROPAGACION

6. CAMPOS ELECTROMAGNETICOS

En este grupo de asignaturas agrupamos todas aquellas que nos hicieron capaces de resolver problemas matemáticos o físicos, en nuestro caso pese a no aplicar la gran cantidad de conceptos vistos en estas asignaturas, nos hicieron abrir los ojos y ser capaces de resolver problemas buscando entre las posibles soluciones que se nos ofrecen. Por todo ello creo que han sido de gran importancia por ejemplo para calcular las distancias entre los diferentes elementos en las escenas, calcular la posición en la que tiene que ir el siguiente elemento, transformar los datos de las propiedades de una figura geométrica en la propia figura geométrica o para el algoritmo de atracción y repulsión para poder asemejar nuestro problema al comportamiento de los planetas en el espacio y poder modelarlo con leyes físicas.

6.3.3. Escritura

Además de las anteriores asignaturas que fueron más técnicas aunque en la mayoría también nos vimos forzados a tener que describir nuestros algoritmos y pensar en la forma de escribir lo que estábamos haciendo, la asignatura de EXPRESION ORAL Y ESCRITA Y BUSQUEDA DE INFORMACION nos sirvió para aprender a escribir mejor de lo que ya hacíamos y expersarnos correctamente en diferentes ambientes o con diferentes tonos, por ello quería resaltar también su importancia.

6.4. Lecciones aprendidas

El trabajo de fin de grado ha sido una forma de aprendizaje muy productiva ya que en nuestro caso hemos llevado a cabo la gestión de un proyecto de software siguiendo metodologías ágiles lo que ha hecho que aprendamos no solo aspectos técnicos sino también de gestión.

Entre todos los aspectos técnicos que hemos aprendido llevando a cabo este proyecto han sido:

1. Aprender a trabajar con A-Frame. Aunque ya habíamos trabajado con JavaScript, A-Frame es un framework novedoso y que no habíamos trabajado nunca con él. Fue un reto muy grande el comenzar a trabajar con el, entender cuales eran sus particularidades y

leer toda la documentación para realmente aprender a trabajar con A-Frame. Una vez lo controlas es un framework muy útil para realizar numerosas escenas en realidad virtual.

2. Diseñar y desarrollar algoritmos de colocación desde cero. Pese a haber diseñado y desarrollado algoritmos durante la carrera, ninguno de ellos tenía la complejidad ni resolvía un problema tan grande como el del proyecto al que nos estamos enfrentado en el trabajo de fin de grado por ello fue un gran reto el diseñar y desarrollar estos algoritmos con una complejidad más elevada.

3. Aprender a escribir la memoria en LaTeX. Para poder presentar la memoria de una forma más profesional y con una calidad mayor decidimos optar por el sistema de composición de textos llamado LaTeX, para ello tuvimos que aprender los comandos que utilizaba el sistema para poder reproducir lo que queríamos en cada momento. Es una herramienta muy útil para escribir documentos profesionales con una calidad mayor de lo normal.

4. Diseño de páginas web. Aunque en la carrera ya vimos algunos conceptos básicos de diseño de páginas web, nunca realmente fue algo nuestro por lo que la realización de este proyecto nos ha servido para aprender a diseñar páginas web de una forma muy profesional para poder mostrar los resultados obtenidos en el transcurso del proyecto.

5. Desarrollo de páginas web. Una vez hicimos el diseño tuvimos que llevarlo a cabo por lo que este proyecto nos ha servido para aprender sobre desarrollo web, tuvimos que maquetar los diseños que habíamos hecho y darle la funcionalidad que tenía que tener para poder visualizar todos los ejemplos y las demostraciones perfectamente. Esta lección fue realmente reconfortante ya que pudimos ver el resultado resumido en una página web donde todo el que quisiera o tuviera curiosidad puede ver el resultado de nuestro trabajo.

6. Utilización de html, css y bootstrap. Para poder llevar a cabo el desarrollo de nuestra página web tuvimos que ayudarnos de los lenguajes de marcado html y css con la ayuda de bootstrap para que todo tenga un aspecto más vistoso y agradable para la gente.

7. Desarrollo de una web para vender el producto. La fase más importante de un proyecto es la de marketing ya que hace falta que la gente sepa que hemos resuelto un problema que a ellos les ocurre y decidan utilizar nuestro producto. Para la fase de marketing diseñamos y desarrollamos una página web para vender nuestro proyecto.

6.5. Trabajos futuros

Hemos dejado el proyecto preparado para que sea la base de futuros proyectos. Nosotros hemos diseñado y desarrollado los componentes y algoritmos necesarios para poder representar conjuntos de datos en dos dimensiones pero en realidad virtual, es decir, no representamos escenas con los elementos formando un elemento en tres dimensiones si no que todos los elementos están sobre un plano. Para un futuro proyecto se puede estudiar la manera de modificar el algoritmo de tal forma que sea capaz de colocar los elementos en tres dimensiones y no únicamente sobre un plano.

Otra idea para futuros trabajos puede ser la de hacer el algoritmo recursivo y jerárquico, por ejemplo nuestro algoritmo es capaz de representar los elementos como si fueran los ficheros que hay dentro de un directorio pero no es capaz de entrar en los subdirectorios. Para futuros trabajos se puede trabajar la recursividad y la jerarquía de los subdirectorios para poder representar los elementos del directorio actual pero también los de los subdirectorios.

La última idea de trabajo es terminar de completar la prueba de concepto de atracción y repulsión, es decir, dar un paso más para poder extrapolar ese algoritmo a todos los elementos de la escena e incluso llevar a cabo el algoritmo de atracción y repulsión en tres dimensiones, actualmente lo hace sobre un plano y tendría mucho sentido si se pudiera hacer en tres dimensiones.

Bibliografía

[1] VALUEXPERIENCE COMMUNITY MANAGER . Claves para un mayor crecimiento competitivo: el Big Data y el Análisis de Datos, 2017,
https://valuexperience.com/mayor-crecimiento-competitivo-el-big-data-y-

[2] MDN. JavaScript,
https://developer.mozilla.org/es/docs/Web/JavaScript

[3] A-FRAME . A web framework for building 3D/AR/VR experiences,
https://aframe.io/

[4] MARÍA TENA . ¿Qué es la metodología 'agile'?,
https://www.bbva.com/es/metodologia-agile-la-revolucion-las-formas-trab

[5] ALBER PEDRAZA . ¿Qué es desarrollo frontend?,
https://desarrollofrontend.com/que-es-desarrollo-frontend/

[6] REVISTA DE ROBOTS . Definición de qué es la Realidad Virtual,
https://revistaderobots.com/rv/definicion-de-que-es-la-realidad-virtual

[7] NEOSENTEC . ¿Qué es la Realidad Aumentada?,
https://www.neosentec.com/realidad-aumentada/

[8] IBERDROLA . Realidad Virtual: otro mundo al alcance de tus ojos,
https://www.iberdrola.com/innovacion/realidad-virtual

[9] MDN . HTML5,
https://developer.mozilla.org/es/docs/HTML/HTML5

[10] THREEJS. three.js,
https://threejs.org/

[11] TIBCO . ¿Qué es un treemap?,
https://docs.tibco.com/pub/spotfire_web_
player/6.0.0-november-2013/es-ES/WebHelp/
GUID-F3F4ABDF-8418-42D3-A1C4-60B7A8121C75.html

[12] JESÚS GONZALEZ BARAHONA . Some notes while learning about A-frame,
https://jgbarah.github.io/aframe-playground/

[13] BABIAXR . BabiaXR,
https://babiaxr.gitlab.io/

[14] STACKOVERFLOW . stackoverflow,
https://es.stackoverflow.com/

[15] STACKOVERFLOW . Most Popular Technologies,
https://insights.stackoverflow.com/survey/2020#
most-popular-technologies

[16] ADEVA . The simplest way to hire world-class developers,
https://adevait.com

[17] NPM . A-Frame packages,
https://www.npmjs.com/search?q=keywords:aframe&page=1&
ranking=optimal

[18] SUPERMEDIUM . Superframe,
https://supermedium.com/superframe/

[19] A-FRAME . Getting Started,
https://aframe.io/docs/master/introduction/

[20] A-FRAME . Example: box Component,
https://aframe.io/docs/master/introduction/
writing-a-component.html#example-box-component