UNIVERSIDAD
REY JUAN CARLOS

GRADO EN INGENIERIA EN TECNOLOGIAS DE LA TELECOMUNICACION

Curso Académico 2019/2020

Trabajo Fin de Grado

ENTORNO DE CREACIÓN DE PROGRAMAS EN REALIDAD VIRTUAL

Autor : Álvaro López García

Tutor : Dr. Jesús María González Barahona

Trabajo Fin de Grado

Entorno de Creación de Programas en Realidad Virtual

Autor : Álvaro López García

Tutor : Dr. Jesús María González Barahona

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2020, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2020

Dedicado a

mi familia / mis compañeros / mis amigos

II

Agradecimientos

La verdad tengo tantas cosas que deciros que no sabría por donde empezar. Está claro que esto no habría sido posible sin ninguno de vosotros. Me habéis apoyado muchísimo siempre y sin pedir nada a cambio. Por ello siempre os estaré agradecido.

En primer lugar me gustaría agradecer su apoyo a mis padres. Vosotros siempre habéis apostado por mí y nunca y en ningún momento habéis dejado de creer en mí. Aunque a veces la situación fuera muy complicada para mí, siempre me habéis dado fuerzas para seguir. Vosotros habeís sido el pilar de mi éxito siempre y en mis momentos más díficiles durante el grado habéis sido mi fuerza y mi motivo para seguir adelante. También se lo agradezco a mi familia. Muchas veces uno está tan ocupado que no es consciente del esfuerzo que hay detrás suyo para ayudarte a salir adelante. Quiero deciros que siempre he sentido vuestro calor, vuestro apoyo y vuestras ganas de verme triunfar.

En segundo lugar se lo quiero dedicar a mis compañeros. Lo que habéis hecho por mí nunca tendré palabras suficientes como para agradecéroslo. Nunca olvidaré esa sensación de verme sobrepasado por el tiempo y las asignaturas. Sin vosotros sé que hay muchas asignaturas que no hubiera sacado adelante o que me hubiera costado el triple de esfuerzo. Además de ayudarme académicamente, me habéis animado siempre a seguir adelante y eso significa mucho para mí. Por supuesto, vosotros sois mis amigos. Al final no todo ha sido estudiar. También hemos tenido momentos de risa dignos de recordar. Me habéis enseñado maneras de vivir y afrontar la vida que nadie más me ha enseñado. Siempre me acordaré de vosotros y siempre podréis contar conmigo. Mi éxito también es vuestro.

En tercer lugar se lo quiero dedicar a mis amigos. Vosotros siempre habéis estado cuando he necesitado desahogarme. Siempre me habéis dado ánimos y fuerza. Siempre habéis creído en mí. Recuerdo todas esas veces cuando estaba obsesionado con estudiar que me habéis sacado de casa o de la biblioteca para dejar de pensar tanto y disfrutar más al menos un rato. Gracias

por esos descansos porque me habéis dado vida, alegría, diversión y tranquilidad.

Resumen

El proyecto consiste en la creación de una biblioteca que permite generar interfaces de programación configurables en realidad virtual en el navegador. Una de las ventajas que tiene dicha interfaz es que no requiere conocimientos técnicos de programación, pues está orientada a un alto nivel de abstracción. De manera que, es amigable y puede ser usada de un modo divulgativo. Esto es posible gracias al uso del framework A-Frame y otras tecnologías como HTML, JavaScript y WebVR fundamentalmente.

Mediante A-Frame generamos el entorno de realidad virtual, el cual utiliza la biblioteca Three.js y es renderizado con WebGL. Con HTML generamos la estructura principal de la escena y después con JavaScript la completamos y la integramos con funcionalidad específica. EL uso de WebVR nos es muy útil para la sincronización del proyecto con dispositivos VR. Aunque no es estrictamente necesario usar dispositivos VR para probar la biblioteca, ya que el proyecto tiene otro modo de uso donde con un ordenador podemos ejecutarlo.

Los recursos del proyecto están alojados en Github. De manera que podemos acceder y hacer uso de ellos de manera gratuita.

Summary

This project creates a library that allows us to generate configurable programming interfaces in virtual reality in a browser. One of the advantages of this interface is that technical programming knowledge is no required for using it, as it is oriented towards a high level of abstraction. So, the interface has been designed to be user friendly and for informational use. This is possible thanks to the use of A-Frame framework and other technologies such as HTML, JavaScript and WebVR, fundamentally.

We generate the virtual reality environment in A-Frame, which uses the Three.js library and is rendered with WebGL. HTML is used to generate the main structure of the scene and JavaScript to complete and integrate it with the specific functionality. The use of WebVR is very useful for synchronizing the project with VR devices. However, it is not strictly necessary to use VR devices to test the library due to the project has another way to be run directly on a PC.

Project resources are hosted on Github. Thereby, we can manage them for free.

Índice general

Índice de figuras

Capítulo 1

Introducción

Con este proyecto perseguimos la elaboración de una interfaz de programación en realidad virtual de uso en el navegador. La idea de este proyecto vino por el afán de crear una herramienta en relidad virtual que pudiera ayudar a introducir a otras personas a la programación, donde los programas en sí son objetos con forma tridimensional. Además, se buscaba que dicha biblioteca permitiese generar nuevas interfaces fácilmente configurables.

El proyecto se realizó principalmente en el framework, A-Frame. Tambíen se hace uso de otras tecnologías como HTML, JavaScript y más.

En este capítulo trataremos el objetivo principal del proyecto, los objetivos específicos, la estructura de la memoria y la disponibilidad del software.

1.1. Estructura de la memoria

En este apartado definiremos la estructura de la memoria, que es la siguiente:

- En el primer capítulo se hace una introducción al proyecto. En él hablamos de manera breve en qué consiste el proyecto, las principales tecnologías utilizadas y la motivación de esta idea.

- En el capítulo 2 se muestran los objetivos del proyecto. Hablaremos acerca del objetivo principal del proyecto y de los objetivos específicos. Esto nos permitirá dar una visión general de porqué en el capítulo 4 buscamos llevar a cabo determinadas funcionalidades y propiedades del trabajo. También hablaremos acerca de la planificación temporal.

- En el capítulo 3 trataremos el estado del arte. En este capítulo hablaremos de las tecnologías utilizadas y de cómo se integran en el proyecto.
- En el capítulo 4 hablaremos del proceso de desarrollo donde lo veremos desde una perspectiva cronológica para analizar el progreso y la dirección que iba tomando el proyecto.
- En el capítulo 5 hablaremos del resultado definitivo del proyecto. Trataremos la guía de usuario, la estructura del proyecto y analizaremos un amplio ejemplo para aprender el uso completo de dicha interfaz.
- En el capítulo 6 trataremos las conclusiones del proyecto. Analizaremos si hemos logrado alcanzar los objetivos marcados, las lecciones aprendidas y los problemas que han surgido durante el proyecto.

Capítulo 2

Objetivos

2.1. Objetivo general

El objetivo de este trabajo fin de grado ha consistido en desarrollar una biblioteca, que basado en técnicas de realidad virtual, permitiese a personas alejadas del mundo del desarrollo de sofware llegar a comprender el proceso de programación de sistemas informáticos complejos mediante su uso y de un modo amigable y divulgativo.

2.2. Objetivos específicos

A continuación, explicaremos cuáles son los objetivos específicos:

1. La creación de dos módulos claros e independientes como lo son el módulo de objetos/móviles o creación de objetos y nuestro IDE o entorno de creación de programas. Programas que se podrán aplicar a los objetos.

2. La creación de un sistema que fuera intuitivo y amigable para que mediante la visualización 3D se puedan adquirir conocimientos básicos sobre los procesos y las formas en las que se puede aplicar la informática.

3. Dar la posibilidad de poder moverse por la escena.

4. Dar la posibilidad de poder interactuar con el entorno mediante herramientas de realidad virtual haciendo uso del navegador.

5. Poder hacer uso de recursos alojados en Github y que estos fueran accesibles a todo el mundo.

6. La creación de un menú de programa o interfaz de programación que permitiera desarrollar, modificar y eliminar programas en tiempo real. Además de que el programa en sí fuera visible en formato 3D.

7. La creación de un panel de configuración de móvil que permitiera eliminar objetos en tiempo real, volverlos a su posición original, elegir un programa y ejecutarlo.

8. La creación de una biblioteca que permitiese modificar la interfaz y generar nuevas interfaces de manera sencilla.

2.3. Planificación temporal

Este proyecto se ha desarrollado a la vez que finalizaba mis prácticas e iniciaba mi vida laboral en la misma empresa.

Durante el proyecto se mantenían reuniones entre el tutor y el alumno ya fueran presenciales en la universidad o mediante videollamada. En esas reuniones se planeaban los objetivos a alcanzar a corto/medio plazo debido a las ocupaciones del alumno, se trataban problemas durante la iteración y se analizaban los resultados alcanzados. Normalmente trabajaba en el proyecto durante las tardes y/o fines de semana.

Capítulo 3

Estado del arte y tecnologías

3.1. A-Frame

A-Frame[6] es un framework de JavaScript el cuál es usado para generar un entorno de realidad virtual (VR). En A-Frame hay una comunidad donde se puede contribuir con desarrollo propio y hacer uso de recursos.

A-Frame está basado en la arquitectura ECS (Entidad, Componente, Sistema). Esta arquitectura permite hacer uso de HTML, JavaScript, WebVR, three.js y WebGL. Las entidades son objetos que tienen asociadas una serie de propiedades. Los componentes son elementos con un comportamiento asociado y si es necesario se le pueden añadir propiedades para su correcto funcionamiento. De manera que, si queremos dotar a una entidad de un comportamiento específico, enlazamos la entidad con el componente. Los sistemas se encargan de gestionar la lógica para el correcto funcionamiento del entorno.

A continuación, veremos un ejemplo[1] de uso de A-Frame:

```
<html>
  <head>
    <script src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
  </head>
  <body>
    <a-scene>
      <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
      <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>
      <a-cylinder position="1 0.75 -3" radius="0.5" height="1.5" color="#FFC65D"></a-cylinder>
      <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4" color="#7BC8A4"></a-plane>
```

[1] https://aframe.io/docs/1.0.0/introduction/

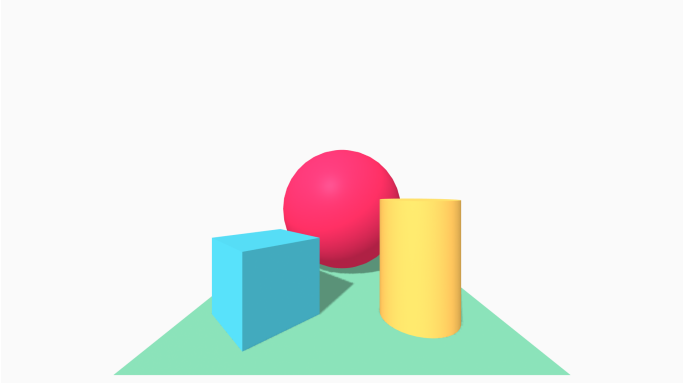

Figura 3.1: Ejemplo de introducción de A-Frame

```
      <a-sky color="#ECECEC"></a-sky>
    </a-scene>
  </body>
</html>
```

Este código es un HTML de un ejemplo de introducción de la página web de referencia de A-Frame. En este ejemplo podemos observar como insertar entidades en nuestra escena.

Primero, en la cabecera (head) del HTML indicamos los scripts que vamos a usar para poder utilizar A-Frame.

Después, en el cuerpo (body) del HTML insertamos la escena con la etiqueta 'a-scene' y dentro de ella todas aquellas entidades que queramos añadir a la escena. En este ejemplo se usan tipos primitivos ya definidos en A-Frame, como 'a-box', 'a-sphere', 'a-plane', etc. Además, para añadir cualquier tipo de entidad podemos hacer uso de la etiqueta 'a-entity'.

Finalmente, si observamos la figura 3.1, podemos ver la escena generada con todas las entidades que ponen el documento HTML.

Por supuesto, al igual que en las etiquetas HTML, podemos añadir atributos específicos a la entidades.

En cuanto a los componentes, son creados mediante A-Frame haciendo uso del lenguaje de programación JavaScript. Los componentes tienen una estructura. Los componentes desarrollados en el proyecto se componen de 3 partes: esquema (schema), inicilización (init) y actualización (update). El esquema sirve para definir las propiedades del componente. La inicilización sirve para inicializar el componente cuando acaba de ser asignado a una entidad. La actualización se accede a ella cuando ha sido modificada una propiedad del esquema del componente. En el apartado 4 veremos algunos ejemplos de desarrollo de componentes y de cómo enlazar

un componente con una entidad.

3.2. JavaScript

JavaScript[5] es un lenguaje de programación interpretado el cuál es usado en el lado del cliente para realizar páginas web dinámicas, pues HTML muestra páginas estáticas, y en el lado del servidor (Node.js). Al ser un lenguaje interpretado no necesita ser compilado antes de ejecutarlo. La principal ventaja de esto es que es menos restrictivo para el desarrollador. La principales desventajas es que introduce errores silenciosos, a diferencia de un lenguaje compilado que es más restrictivo, y que en ejecución es más lento. El DOM transforma el HTML en un árbol de nodos, donde los elementos se transforman en nodos. De este modo, mediante JavaScript podemos manipular el DOM ya sea para acceder a él, crear o eliminar nodos.

Además de programación secuencial, JavaScript permite programar en base a eventos que definimos nosotros en nuestra página web, por ejemplo, hacer click en un determinado botón o simplemente pulsar un botón estando el cursor en un input. Se definen manejadores de eventos que se ejecutan cuando se da lugar el evento asociado. Este modelo de programación se convitió en un estándar a partir de HTML4 para todos los navegadores a excepción de Intenet Explorer que tiene su propio modelo.

También, como cualquier otro lenguaje de programación, te permite hacer uso de variables, realizar operaciones con números, strings, etc.

3.3. HTML5

HTML5[2] es actualmente la última versión de HTML. HTML, en español Lenguaje de Marcado de Hipertextos, es un lenguaje de marcado usado en el desarrollo del lado del cliente (frontend) para realizar la estructura del contenido del sitio web. Es de marcado porque utiliza etiquetas para insertar contenido e hipertexto porque enlaza unas páginas web con otras mediante enlaces. Como veremos más adelante, gracias a HTML hemos podido aplicar otras tecnologías compatibles como A-Frame y JavaScript haciendo uso de la etiqueta ”script”. A diferencia de HTML4, introduce una serie de nuevas características. Algunas de ellas son:

- Elementos de audio y multimedia.
- Nuevas etiquetas para estructurar el DOM.
- Visores.

También introduce una serie de nuevas APIs, como por ejemplo:

- Geolocalización.
- Web Storage.
- Web Workers.
- HTML Canvas.

3.4. GitHub

Github[1] es una red social y una plataforma para desarrolladores y equipos de trabajo. Está basado en Git, un sistema de control de versiones distribuida el cual permite tener el historial de todos los cambios realizados y de los usuarios autores de dichos cambios.

También, permite estar informado sobre la interacción de los usuarios propios de GitHub y de los proyectos llevados a cabo por otros usuarios.

3.5. CSS3

CSS3[3] pertenece a la familia de versiones de CSS. Siendo así la última versión desarrollada de este lenguaje de hojas de estilo, el cual es usado en el desarrollo del lado del cliente (frontend). Este lenguaje se encarga de la presentación del documento. De este modo, queda separada la presentación del contenido.

Algunas novedades de CSS3 son:

- Uso de diferentes tipografías.
- Efectos aplicables al texto.
- Colores RGBA.

- Múltiples imágenes de fondo.
- Bordes redondeados.

3.6. Sublime Text

Sublime Text[4] es un editor de código fuente y de texto desarrollado por Jon Skinner en C++. Además, para el uso de plugins se puede hacer uso de la API de Python. Está disponible para Windows, GNU Linux y Mac Os X.

Algunas características de Sublime Text son:

- Auto-completado y marcado de llaves.
- Paleta de Comandos.
- Búsqueda avanzada.
- Acceso a plugins que aportan nueva funcionalidad.

3.7. Three.js

Three.js[7] es una biblioteca 3D en la cual está basada A-Frame. Actualmente está desarrollada en JavaScript.

Gracias a Three.js, se pueden inyectar en nuestro sitio web elementos 3D como un escenario, luminosidad, efectos, objetos 3D, elementos o funciones matemáticas. Además, hace uso de renderizadores, materiales y texturas.

También, siguiendo una serie de reglas, se puede agregar nueva funcionalidad y contribuir a Three.js para que otros desarrolladores hagan uso de ello.

3.8. WebGL

WebGL[9] es una API basada en JavaScript la cual fue creada para poder generar gráficos 3D sin la necesidad de usar plugins.

Fue creada por Mozilla y posteriormente se creó el Grupo de Trabajo de WebGL, donde inicialmente paticiparon Apple, Google, Mozilla y Opera. Por ello, WebGL se incluyó en Safari, FireFox, Google Chrome y Google Maps.

Se ejecuta en Canvas (HTML5) y al ser un API DOM es compatible con aquellos lenguajes de programación que trabajen sobre el DOM. Por tanto, es compatible con frameworks como jQuery, Three.js y Babylon.js.

3.9. WebVR

WebVR[8] es una API desarrollada en JavaScript la cual fue creada para poder tener un experiencia 3D en el navegador. Para tener una experiencia completa hay que hacer uso de otras APIs y tecnologías como Web Audio y WebGL. Actualmente, está en desuso y será sustituida por WebXR.

WebVR tiene una interfaz (VRDisplay), la cual contiene todos los métodos necesarios para obtener las referencias a los dispositivos, obtener los fotogramas y representarlos con una velocidad de fotograma.

3.10. GIMP

GIMP[10] es un software libre diseñado para edición de imágenes. Actualmente, está disponible para GNU/Linux, Windows y Mac OS X. Se caracteriza por algunas de las siguientes propiedades:

- Trabaja con formatos de imagen como gif, jpg, png, xcf, etc.
- Permite trabajar con diferentes capas.
- Herramientas de selección, como selección rectangular, selección elíptica.
- Herramientas de pintura, como relleno, degradado, lápiz, pincel.
- Herramientas transformación, como mover, recortar, inclinar, voltear.
- Permite trabajar con filtros, como difuminar, realzar, luz y sombra.

3.11. LaTeX

LaTeX[11] es un software libre de elaboración de documentos de texto procedente de TeX. Se asemeja a Word con la ventaja de que LaTex separa las reglas de estilo del contenido. Por tanto, una vez estén definidas esas reglas estas se aplicarán al documento de manera automática, lo que permite centrarse en el contenido.

Además, está disponible para diferentes sistemas operativos y es posible exportar en diferentes formatos como HTML, PDF, SGML, etc.

Capítulo 4

Proceso de desarrollo

En este apartado, trataremos el modelo empleado para el desarrollo de este proyecto. Para ello se ha hecho uso de la metodología Scrum[12], en la que el cliente es el tutor del proyecto y el equipo de desarrollo lo forma únicamente el autor.

Scrum es una metodología creada por Ken Schwaber y Jeff Sutherland, la cuál permite organizar el trabajo y el personal haciendo uso de diferentes procesos y técnicas para desarrollar el proyecto de forma ágil y eficaz. El trabajo a realizar puede ser de cualquier tipo. No es exclusivamente aplicable a ingeniería de software.

El equipo está formado por el Product Owner, el Equipo de Desarrollo y el Scrum Master. El Product Owner es aquella persona encargada de que el proyecto salga adelante. El Scrum Master se encarga de ayudar al Product Owner a que todo se desarrolle de manera correcta acorde a los principios de Scrum de transparencia, inspección y adaptación. El Equipo de Desarrollo se encarga de llevar a cabo las tareas definidas en cada Sprint aplicando los principios de Scrum.

Los Sprints son espacios de tiempo definidos con una duración específica en los cuales hay que llevar a cabo las tareas definidas. Por cada Sprint se hace un seguimiento y una vez finalizado se obtiene una versión funcional la cual podrá seguir siendo mejorada de manera incremental en Sprints posteriores. Antes de empezar con el Sprint, se hace una planificación. Durante el Sprint, se hacen un serie de reuniones para llevar un seguimiento del avance y del trabajo aportado por cada miembro. Por último, una vez finalizado, se hace un repaso de todo aquello sucedido en relación al Sprint para localizar puntos en los que se puede mejorar eficiencia de cara al próximo Sprint.

En cuanto al método seguido en el proyecto, estamos ante un caso en el que somos dos

integrantes. Teníamos una reunión previa al comienzo da cada iteración donde definíamos los objetivos y algunas cuestiones técnicas. Los Sprints o iteraciones solían durar alrededor de dos semanas. En caso de que surgiera algún problema o cuestión se producían reuniones durante el Sprint. Una vez finalizado se procedía a una revisión de resultados para comprobar si se han cumplido los objetivos y comprobar la forma y el modo en el que se han llevado a cabo las tareas.

A continuación, analizaremos las iteraciones que se han dado lugar durante el proyecto.

4.1. Iteración 0. Primera toma de contacto

En esta primera iteración tuve la primera toma de contacto con el uso de A-Frame. Como experimento, se me ocurrió desarrollar un breve juego con mis primera nociones en A-Frame, que explicaré a continuación en el siguiente subapartado. Puedes consultar y realizar las pruebas necesarias del juego[1].

4.1.1. Objetivos

Los objetivos de esta iteración eran aprender a crear un marco de simulación virtual en el navegador, aprender a definir entidades en el HTML, aprender a desarrollar componentes de A-Frame en JavaScript, aprender a vincular un componente a un evento, aprender a asignar un componente a una entidad, aprender a insertar texturas en el HTML y aprender a usar texturas sobre las entidades.

4.1.2. Desarrollo

Esta escena consiste en un juego sencillo en el cual ganas si llegas a la última habitación y lees el texto ”THE END” que hay en la pared. Para ello, hay que atravesar un camino, entrar en la casa y llegar a la última habitación. Si en algún momento del juego caes del camino o del jardín, caes a la plataforma donde pone ”GAME OVER”; por tanto, pierdes.

[1] `https://alvarolopezgarcia.github.io/a-frame-pruebas/Scene-House/camera-2.html`

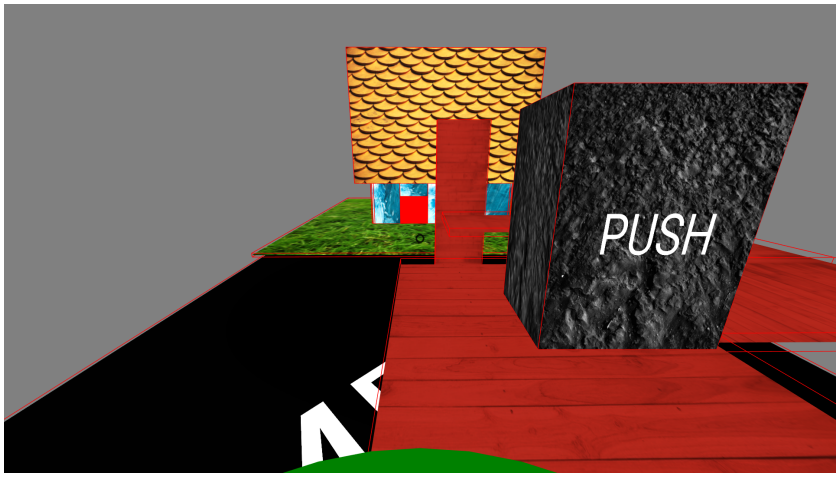

Figura 4.1: Inicio de la escena

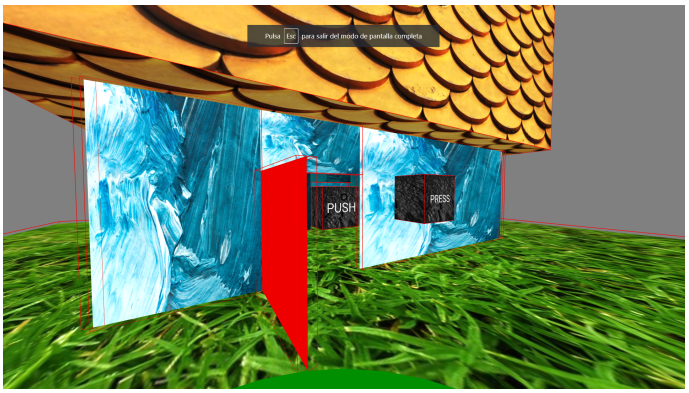

Figura 4.2: Casa

Como se puede apreciar en la figura 4.1, para atravesar el camino y saltar al jardín, hay que llevar a cabo una serie de pasos:

1. Hay que empujar la caja con el texto ”PUSH”para poder continuar por el camino sin caer a la plataforma ”GAME OVER”.

2. Hay que conseguir que la plataforma, que tiene una inclinación de 90º, se incline a 0º para poder saltar al jardín. Esto se consigue tocando con el cursor la plataforma. De el mismo modo, si vuelves a tocar la plataforma con el cursor cambia la inclinación a 90º. En la figura el cursor es el círculo negro que hay sobre el césped.

A continuación, como se puede apreciar en la figura 4.3, para acceder a la casa hay que seguir una serie de pasos:

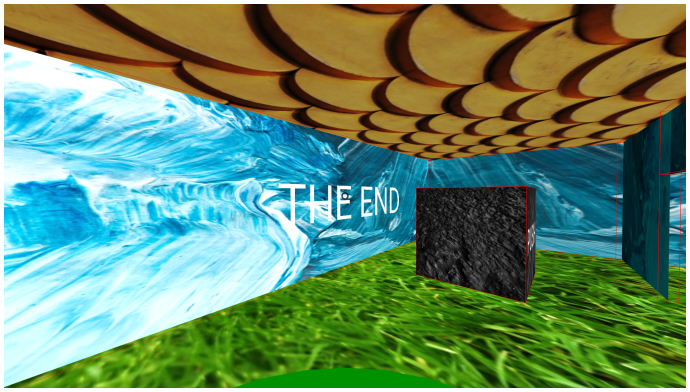

Figura 4.3: Final del juego

1. Hay que presionar el botón ”PRESS”, para que se abra la puerta. Para ello hay que situar el cursor encima del botón y realizar click.

2. Por último, hay que empujar la caja ”PUSH”, para acceder a la última habitación de la casa donde podremos ver el texto ”THE END”. Obsérvese la figura 4.2.

Para llevar a cabo este juego he tenido que hacer uso de HTML5, JavaScript y A-Frame. En el HTML he definido las entidades necesarias para componer el escenario. El camino, las plataformas, las paredes, la puerta y el techo son planos o composiciones de planos en posiciones determinadas. Todo ello son entidades. De las cuales algunas tienen un comportamiento asociado o componente. Los componentes están definidos en components.js. Dichos componentes son creados mediante A-Frame, el cual hace uso de JavaScript.

Veamos un breve ejemplo[2] de un componente.

```
AFRAME.registerComponent('clickable', {
  schema: {
    event: {type: 'string', default: 'click'},
  },

  init: function () {
    var self = this;

    this.eventHandlerClick = function () {
        let door = document.getElementById('door');
        let rotation_Y = door.getAttribute("rotation").y;
```

[2] https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Scene-House/components.js

```
        if(rotation_Y === 0){
            door.setAttribute("position",{x:-3,y:2,z:-5});
            door.setAttribute("rotation",{x:0,y:90,z:0});
            door.setAttribute('door_component', {door_state: 'Opened'});
        }else{
            door.setAttribute("position",{x:-2,y:2,z:-6});
            door.setAttribute("rotation",{x:0,y:0,z:0});
            door.setAttribute('door_component', {door_state: 'Closed'});
        }
    };
  },

  update: function(oldData) {
   var el = this.el;
   var data = this.data;

   //The first time we call update, oldData hasn't got any attribute
   if(! oldData.event){
        el.addEventListener(data.event, this.eventHandlerClick);
   }
  },
});
```

En este fragmento de código podemos observar el componente creado para que al pulsar el botón "PRESS"se abra o se cierre la puerta de la casa. Este componente ha sido llamado 'clickable'. En 'schema' definimos los atributos que necesitemos utilizar para el correcto funcionamiento del componente y de dicha entidad. En 'init' inicializamos el componente, es decir, inicializamos el manejador de evento 'eventHandlerClick'. En 'update' únicamente se accede cuando inicializamos ese componente y cuando se produce algún cambio en los atributos propios del componente. En este caso, al acceder por primera vez, 'oldData' no tiene los atributos del componente pues este acaba de ser asignado a la entidad, por tanto, se le asigna a esa entidad el manejador de evento 'eventHandlerClick' junto con el evento 'click' asociado. Así, cuando en la escena se haga 'click' sobre el botón 'PRESS', el manejador abrirá o cerrará la puerta.

Veamos un ejemplo[3] como es asignado dicho componente a su entidad.

```
 <a-box clickable static-body position="0.5 2.5 -6" width="1" height="1" depth="1.7" src="#caja">
            <a-text value="PRESS" color="white" position="-0.35 0 0.85" rotation="0 0 0"
                scale="1 1 1"></a-text>
</a-box>
```

[3] https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Scene-House/camera-2.html

En este ejemplo, se puede ver como se asigna el componente 'clickable' a la entidad 'a-box' (botón 'PRESS').

4.2. Iteración 1

En esta iteración hemos asentado las bases de la estructura y de las primeras funciones del proyecto. En ella trabajamos con un programador y un móvil. El usuario, a partir del programador, podrá crear instrucciones para después ejecurtarlas y provocar que se mueva el móvil. A continuación, explicaremos los resultados y el proceso de desarrollo de la iteración. Trataremos las 3 fases seguidas en esta primera iteración.

4.2.1. Objetivos

El objetivo de esta iteración era establecer una versión funcional donde teniendo un móvil y un programador pudieramos crear un programa simple y ejecutar sus instrucciones. Para ello había que definir una buena estructura en el DOM y crear los componentes necesarios para lograr la funcionaliadad buscada. Para lograr el objetivo tuve que aprender a crear e insertar elementos en el DOM mediante JavaScript para poder inicializar la forma del programador y generar las instrucciones. También tuve que profundizar más acerca del uso de eventos y sobre los problemas y soluciones que se pueden llevar a cabo.

4.2.2. Primera Fase

En la primera fase se creó una estructura inicial y unos componentes para dar comienzo con los objetivos del proyecto. Todo acerca de la primera fase lo podemos encontrar en el siguiente enlace[4]. Hablaremos de los archivos:

- components.js
- exercise-1.html

[4] `https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Exercises/exercise1`

En esta primera fase hemos definido 3 componentes en el script (components.js). Para el programador (PROGRAMMER), para el móvil y para cada instrucción. En el HTML (exercise-1.html) hemos definido como cajas (a-box) al programador y al móvil; cada uno con su componente correspondiente. En efecto, en la figura 4.4 podemos ver, que al inicializar la simulación de realidad virtual en el navegador, solamente están el programador y el móvil. Las instrucciones serán insertadas en el DOM, como último hijo del nodo 'a-scene' (nodo padre), cuando el usuario mueva el cursor sobre el programador. De modo que se creará una caja blanca que representa una instrucción. Este comportamiento se observa en la figura 4.5. Para ejecutar todas las instrucciones hay que realizar click sobre el programador. Cada instrucción desplaza una posición más arriba al móvil con respecto a su posición en ese momento.

Las instrucciones utilizan el componente 'instruction_component'. El manejador de evento de este componente tiene asociado el evento 'click'. De modo que al emitirse este evento sobre la entidad que utiliza este componente se ejecutarán las sentencias definidas en el manejador. Cada instrucción para ser diferenciada del resto tiene definido un id, por ejemplo, la primera sería 'instruction0', la segunda 'instruction1' y así progresivamente. El id es un atributo HTML cuyo valor es único. Es utilizado por claridad y, para identificar y acceder a un elemento de manera rápida. El esquema del componente tiene definida la propiedad 'event' para asignar el evento a su manejador de evento.

El móvil utiliza el componente 'mobile_component'. En esta versión este componente no tiene definida ni funcionalidad ni esquema. En las siguientes versiones e iteraciones del proyecto veremos la evolución de este componente. El id del móvil es 'mobile'.

El programador utiliza el componente 'programmer_component'. Este componente tiene definidos dos manejadores de eventos. El primero asociado con el evento 'mouseenter', el cual se encarga de generar nuevas instrucciones. El segundo asociado con el evento 'click', se encarga de buscar las instrucciones en el DOM y ejecutarlas en el orden en el que fueron creadas. El id del programador es 'programmer' y tiene el texto 'PROGRAMMER', pues se ha definido como la entidad 'a-text' con ese valor en el atributo value. Al ser un hijo del programador (nodo padre) aporta la ventaja de que si, por ejemplo, cambiamos la posición del programador también cambiará la posición del texto. Esto se debe a que cualquier modificación realizada en el nodo padre repercute en sus hijos. El esquema del componente tiene definidas 3 propiedas: events, count y list. Events es un array el cual contiene los eventos que usarán los manejadores

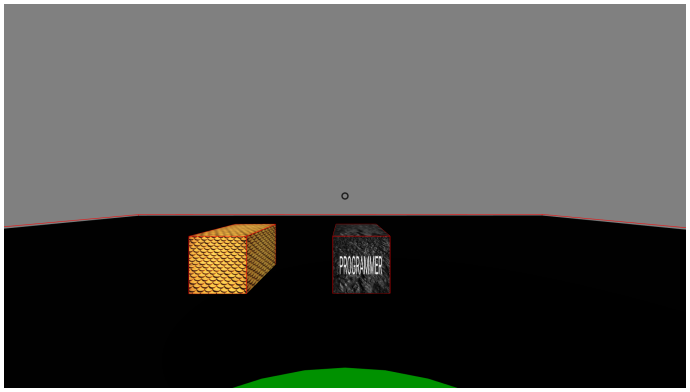

Figura 4.4: Inicio Ejercicio 1

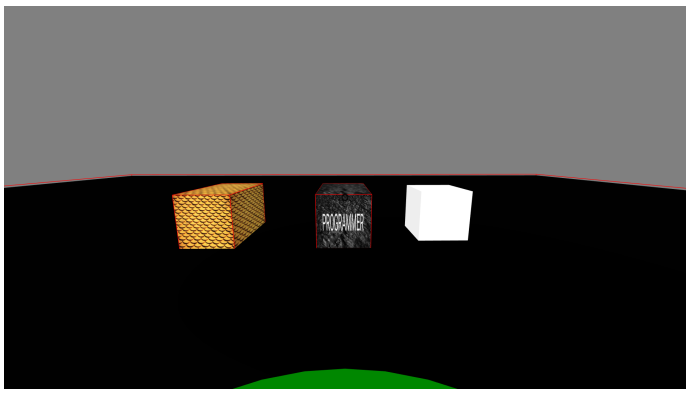

Figura 4.5: Nueva Instrucción Ejercicio 1

de evento. Count es un contador de instrucciones usado para definir el id de cada instrucción. List es un array usado para almacenar el id de cada instrucción.

4.2.3. Segunda Fase

En esta fase[5], se trabajó sobre dos archivos, que son los siguientes:

- components.js
- exercise-2.html

En esta fase de la iteración, se pensó en la idea de desarrollar un nuevo componente y crear una entidad padre como ámbito de programación que conteniera como nodos hijos todo aquello

[5] `https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Exercises/exercise2`

relacionado con el programador y lo que generase, es decir, el programador y las instrucciones. Esto provocó un problema de recursividad, ya que las instrucciones y uno de los manejadores del programador escuchaban el mismo evento. De manera que al emitirse el evento click en una instrucción, el hilo de ejecución volvía al programador, es decir, al nodo padre y así sucesivamente. Para solucionar este problema cambié el evento del componente instruction_component a mouseleave. Así padre e hijo escuchaban diferentes eventos.

También se hicieron otras modificaciones relacionadas con la funcionalidad del móvil y del programador a la hora de ejecutar las instrucciones. Se modificaron de manera que el programador buscara el móvil mediante su id y este fuera el que buscara y ejecutara las instrucciones de dicho programador. Por ello fue eliminada la propiedad list del esquema de programmer_component, pues era usada para formar una lista de instrucciones. Tampoco se implementó esta propiedad en el componente mobile_component, pues se podía acceder a todas las instrucciones a través del nodo padre, es decir, el programador. En el esquema de mobile_component se implementaron las propiedades 'event' para definir el evento usado en este componente y 'program' para indicar al móvil que programa es el que usa.

4.2.4. Tercera Fase

En esta fase[6], se trabajó sobre dos archivos, que son los siguientes:

- components.js
- exercise-3.html

Empezaremos hablando de las modificaciones realizadas en el HTML. A diferencia de la fase anterior el programador es una entidad o nodo padre que contiene como nodos hijos el texto, una entidad nueva que contendrá todas las instrucciones y la forma (a-box). En la fase anterior cuando se definía el programador se definía a su vez la geometría.

En cuanto al script, la forma del programador se inserta en el DOM al inicializar el componente programmer_component, se ha creado un nuevo componente llamado 'instructions' para la entidad que contendrá las instrucciones y el nuevo evento asociado al componente instruction_component se llama 'run'; por tanto, cuando el móvil quiere ejecutar las instrucciones

[6] https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Exercises/exercise3

emite el evento run sobre cada una de ellas. Esto se puede realizar debido a que no es obligatorio usar eventos predefinidos con una acción predefinida, como click, mouseleave, etc.

4.3. Iteración 2

En esta iteración seguimos trabajando con un móvil y un programador. Anteriormente las intrucciones generadas eran todas en una dirección. Veremos que ahora se pueden generar instrucciones para cuatro direcciones diferentes. Además, se ha añadido otra funcionalidad que permite realizar más operaciones con el programador. Trataremos las dos fases seguidas en esta iteración.

4.3.1. Objetivos

El objetivo de esta iteración era ampliar la funcionalidad del programador, es decir, realizar un menú para poder realizar programas más complejos. Además de otras funcionalidades como eliminar todas las instrucciones, ejecutar, eliminar el programa y volver el móvil a su posición original. Para ello ha sido necesario ampliar en el DOM la estructura del programa y crear los componentes necesarios. Otro objetivo era dar más visibilidad al menú y a las instrucciones para poder diferenciarlas. Para ello se hizo uso de GIMP.

4.3.2. Primera Fase

En esta fase[7], se trabajó sobre dos archivos, que son los siguientes:

- components.js
- exercise-4.html

En la primera fase de la iteración nos encargamos primero de desarrollar dos botones que fueran capaces de generar dos tipos de intrucciones diferentes, arriba y abajo. Como podemos ver en la figura 4.6, a diferencia de la anterior iteración, tenemos dos botones. El botón verde genera instrucciones 'up' y el botón rojo genera 'down'. De momento visualmente no podemos

[7] `https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Exercises/exercise4`

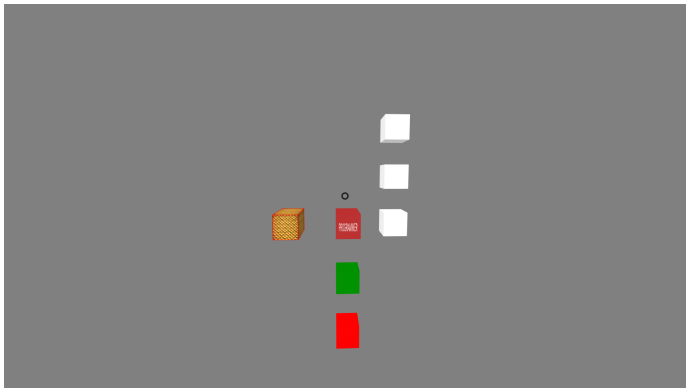

Figura 4.6: Ejercicio 4

distinguir que tipo de instrucción es cada una, pues tienen todas la misma textura. Esto no quiere decir que no tengan funcionalidades diferentes. En la siguiente fase tratamos dicha cuestión.

En el HTML se han añadido dos entidades que hacen referencia a los botones. Estas han sido añadidas como hijos del programador.

En el script se han hecho una serie de modificaciones. El esquema de instructions_component ha sido modificado. Tiene una nueva propiedad usada para diferenciar de que tipo es la instrucción generada, denominada 'type'. El programador ha sido modificado para que solo ejecute, pues ahora los botones son los encargados de generar las instrucciones. Se ha creado el componente 'button' para que hagan uso de él los botones. El esquema tiene la propiedad 'text', usada para definir el botón. Este componente tiene un manejador de evento por botón. Cada uno con una funcionalidad diferente pues generan instrucciones diferentes. Ambos manejadores siguen calculando la posición de las intrucciones para que no haya colisión entre ellas y estén unas encima de otras apiladas a la derecha del programador. Este componente escucha el evento click. Esto supuso otro problema de propagación de eventos pues tanto el programador como los botones escuchan el evento click. En la iteración anterior se optó por cambiar el evento a uno de los componentes pues no suponía ningún problema. En este caso, queremos que ambos escuchen el evento click para interactuar del mismo modo con ambos. De modo que la solución adoptada tenía que ser diferente. Para solucionar este problema se hizo uso de la interfaz de evento de DOM[8] para grabar el evento realizado en este parámetro y poder implementar el método stopPropagation() para detener la propagación del evento click a la entidad padre, es decir, al programador.

[8] `https://developer.mozilla.org/es/docs/Web/API/Event`

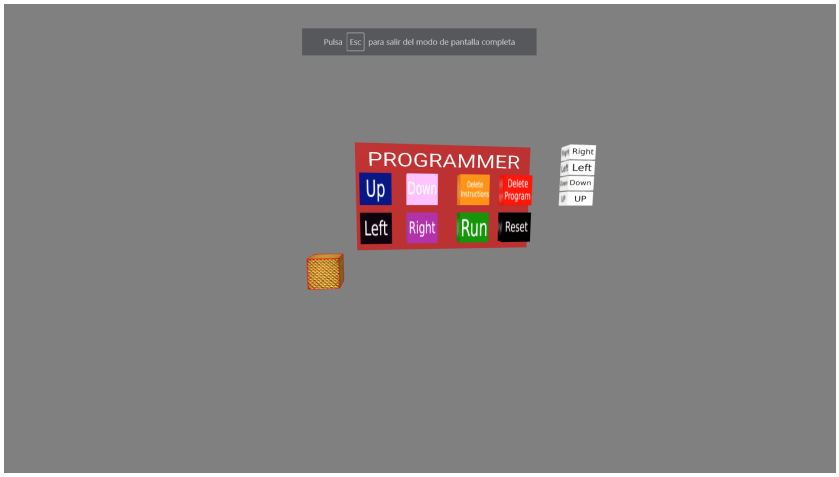

Figura 4.7: Ejercicio 5 con instrucciones generadas

4.3.3. Segunda Fase

En esta fase[9], se trabajó sobre dos archivos, que son los siguientes:

- components.js
- exercise-5.html

Como podemos ver en la figura 4.7, en la segunda fase de la iteración hemos añadido dos botones más para generar instrucciones, izquierda y derecha. También, hemos añadido otros cuatro botones: eliminar instrucciones, eliminar programa, ejecutar y resetear. En esta fase hemos hecho las texturas de los botones y de las instrucciones haciendo uso de GIMP. Hemos aplicado una capa de texto y otro de color de fondo. Después, hemos añadido estas texturas en los 'assets' del HTML para poder hacer uso de ellas.

```
<a-entity cursor="rayOrigin:mouse"></a-entity>
<a-entity laser-controls="hand: right"></a-entity>
```

El HTML ha sido simplificado y modificado. Como podemos ver en estas líneas de código, el cursor ha pasado a ser ahora el ratón y se ha definido el mando derecho como puntero láser para poder interactuar dentro de la escena con las gafas de relidad virtual y los mandos. Además, el programador ya no está definido en el HTML. Se encargará de inicializarlo e insertarlo en el DOM el componente de ámbito de programación.

[9] https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Exercises/exercise5

En el script se han modificado componentes. El ámbito de programación crea las entidades, las geometrías, aplica texturas y posiciones, enlaza los componentes con su respectiva entidad e inserta las entidades en el DOM formando así la estructura del menu del programa entero. El componente programmer_component solamente tiene esquema, el cual tiene la propiedad count para contar el número de instrucciones existentes. El componente mobile_component tiene una propiedad nueva en el esquema denominada 'position'. Es utilizada para guardar la posición original del móvil. El componente instruction_component decide en qué dirección se va a mover el móvil en función del valor de la propiedad type de la instrucción. El componente button tiene dos manejadores de evento para generar instrucciones izquierda y derecha. Además, tiene otros cuatro manejadores de evento adicionales con funcionalidad completamente diferente. El manejador 'eventButtonHandlerRun', usado por el botón run para ejecutar las instrucciones generadas sobre el móvil. El manejador 'eventButtonHandlerReset', usado por el botón reset para devolver a la posición de origen al móvil. Al inicializar el componente se establece el valor de la propiedad como la posición actual. es decir, la posición inicial. Esta acción es realizada gracias a la propiedad position del componente que utiliza el móvil. El manejador 'eventButtonHandlerDeleteInstructions', usado por el botón Delete Instructions para eliminar las instrucciones creadas por el programador y reininiciar el contador de instrucciones a cero, es decir, la propiedad count de programmer_component. Por último, el manejador 'eventButtonHandlerDeleteProgram', usado por el botón Delete Program para eliminar el programa entero incluidas las instrucciones. Como podemos ver en la figura 4.8, al usar dicho botón es eliminado todo el programa de manera que queda el móvil en la escena. Este componente busca el nodo padre de dicho programa y lo elimina del DOM. En consecuencia, son eliminados sus hijos, es decir, las entidades que componen todo el menú y sus instrucciones.

4.4. Iteración 3

En esta iteración pasaremos a trabajar con varios programas y móviles a la vez. Ampliaremos y modificaremos el menú de programa, crearemos el menú de móvil y crearemos los menús de ámbito de programación y de móviles. Trataremos las dos fases seguidas en esta iteración.

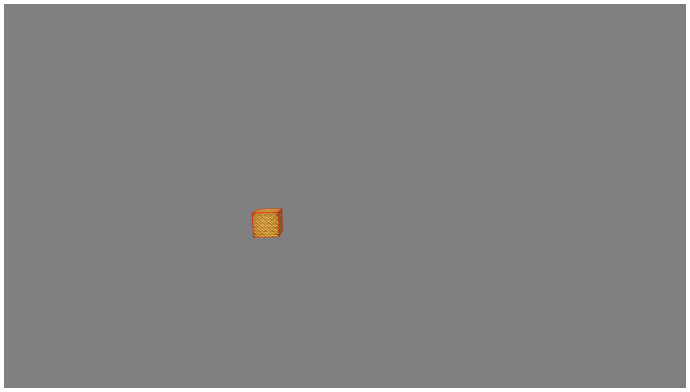

Figura 4.8: Resultado obtenido después de usar el botón Delete Program

4.4.1. Objetivos

El objetivo de esta iteración era pasar a trabajar con varios móviles y programas a la vez. Para ello había que crear un menú que contuviera todos los programas y fuera capaz de crear nuevos programas, y otro menú que contuviera todos los móviles y fuera capaz de crear nuevos móviles. Hubo que pensar la forma en la que se iban a distribuir en la escena todos los programas, móviles e instrucciones generados. Además, hubo que realizar una distribución de la funcionalidad ya creada pues la función de ejecutar y resetear pertenecía a los móviles. También, otro objetivo era ampliar el número de intrucciones que se pudieran generar y formar un menú de móvil con toda la funcionalidad básica necesaria, es decir, eliminar móvil, ejecutar, resetear y cambiar de programa.

4.4.2. Primera Fase

En esta fase[10], se trabajó sobre dos archivos, que son los siguientes:

- components.js
- exercise-6.html

En el HTML se han hecho pocas modificaciones respecto a la iteración anterior. Se ha añadido en los assets la textura creada para el botón New Program y se ha modificado el valor de la propiedad program de mobile_component a programmer1 (id del primer programa). Esta

[10] `https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Exercises/exercise6`

última modificación se debe al hecho de que en esta fase el objetivo era trabajar con un móvil y varios programas. De manera que al cargar la escena por primera vez el móvil tuviera asignado el primer programa. Esto quiere decir que en esta fase solamente podrá ejecutar las instrucciones del primer programa pues no tenemos modo aún de cambiar de programa.

En el script se han hecho una serie de modificaciones. En el componente de ámbito de programación se ha añadido al esquema la propiedad 'count' la cual se encargará de llevar la cuenta de cuantos programas han sido creados para formar el id de los programas. En el componente programmer_component se ha añadido en el esquema la propiedad 'position' para saber de qué número de programa se trata. Por último, en el componente button tiene un nuevo manejador de evento creado para el botón New Program el cual se encarga de crear nuevos programas y hacer que se generen uno debajo de otro sin colisión. En este sentido el comportamiento es parecido a la hora de calcular las posiciones de las instrucciones. Para calcular en qué posición del espacio se colocarán las instrucciones se utiliza la propiedad 'position' del componente programmer_component para definir la primera instrucción de ese programa. Las siguientes se calculan en base a la posición de la última instrucción generada. De manera que se superponen una encima de otra. Los programas son colocados debajo del último programa existente. A continuación, vamos a realizar y explicar una secuencia de ejecución:

- Como se puede observar en la figura 4.9, los programas y las instrucciones se generan como he mencionado anteriormente. Se ha generado un programa más con el botón New Program, se ha generado la instrucción UP con el programa uno y la instrucción Down con el programa dos. Además el menú principal se ha ampliado para hacer hueco al nuevo programa.

- Si le damos al botón Run del programa dos el resultado obtenido es el mismo al de la figura 4.9.

- Si le damos al botón Run del programa uno el móvil se desplaza hacia abajo como podemos ver en la figura 4.10.

- Si hicieramos click sobre el botón Reset de alguno de los programas, el móvil volvería a su posición original, es decir, como en la figura 4.9.

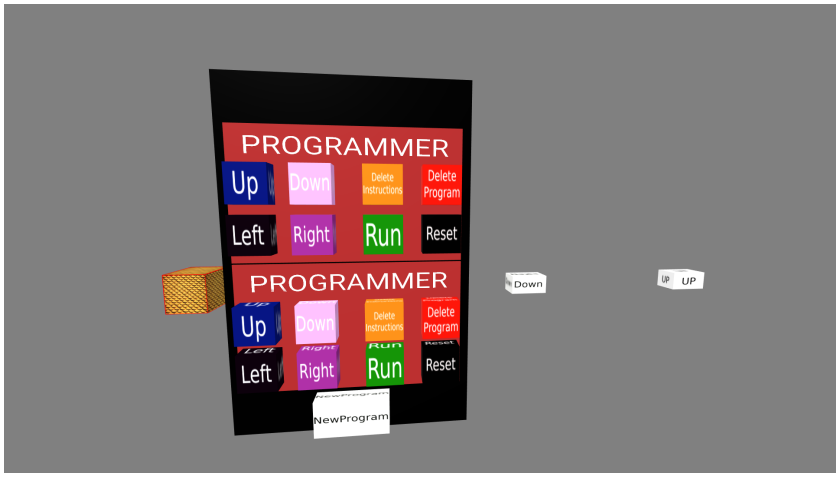

Figura 4.9: Resultado obtenido después de generar dos programas e instrucciones con cada uno de ellos

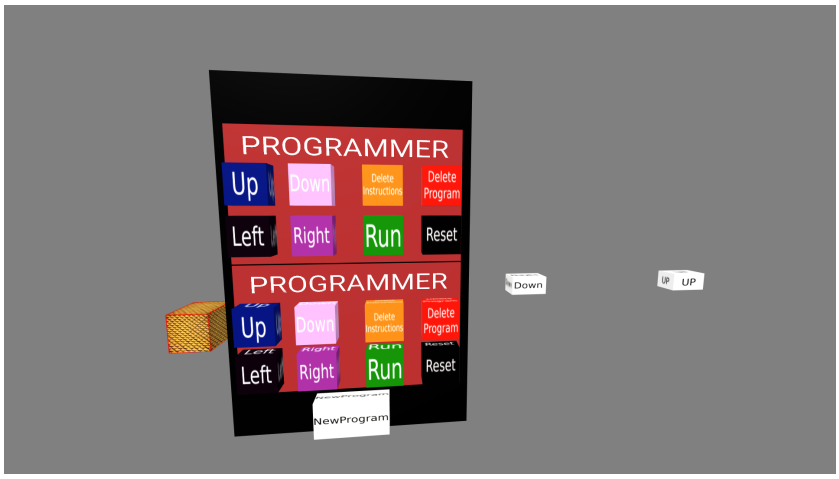

Figura 4.10: Resultado obtenido después de hacer click sobre el botón Run del primer programa

- A continuación, si hacemos click sobre el botón Delete Instructions del primer programa, se eliminan sus instrucciones obteniendo el resultado de la figura 4.11.

- Por último, si hacemos click sobre el botón Delete Program del segundo programa veremos, al igual que en la figura 4.12, se reordena el menú de programas, para que no haya ningún hueco en blanco, y se elimina dicho programa y sus instrucciones.

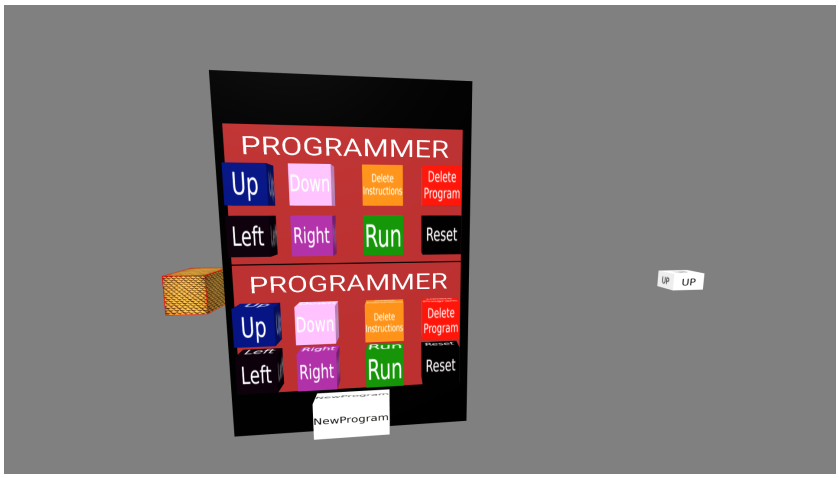

Figura 4.11: Resultado obtenido después de hacer click sobre el botón Delete Instructions del primer programa

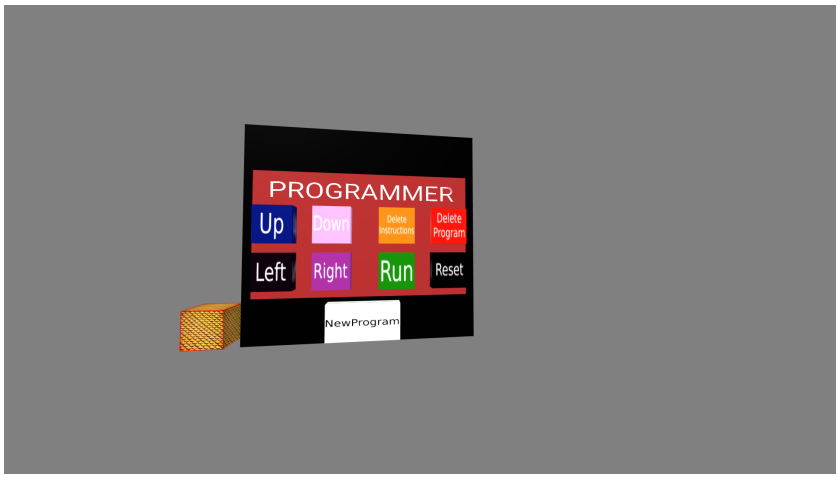

Figura 4.12: Resultado obtenido después de hacer click sobre el botón Delete Program del segundo programa

4.4.3. Segunda Fase

En esta fase[11], se trabajó sobre dos archivos, que son los siguientes:

- components.js
- exercise-7.html

En el HTML, a diferencia de la fase anterior, se han realizado una serie de modificaciones. Se han añadido en assets la nuevas texturas definidas para el menú del móvil y se han añadido los veinte iconos usados por los programas. Estos iconos son usados para ayudar a diferenciar unos programas de otros y para identificar el programa del cual está haciendo uso el móvil. También, se ha borrado el móvil que había definido y en su lugar se ha definido la entidad padre que contendrá todos los móviles con el componente 'mobiles'.

En el script se han modificado componentes y se ha añadido un componente nuevo. Empecemos con el nuevo componente denominado mobiles. En el esquema del componente tiene definida la propiedad 'count' para llevar la cuenta de los móviles creados y formar el id de cada uno. En init inicializa el menú de móviles con el botón New Program e inicializa el primer móvil junto a su menú. El menú de un móvil contiene los botones Run, Reset, Delete Mobile y Program. También, contiene una placa la cual refleja el icono del programa que esté usando. En caso de no usar ninguno, esta se encuentra en blanco. Véase la figura 4.13. El menú de programa ha cambiado. El componente de ámbito de programación a diferencia de la fase uno inicializa del primer programa todos los botones que generan instrucciones, los botones de eliminar instrucciones y programa y dos nuevos botones (Forward y Back). Véase la figura 4.14. En el componente programmer_component se ha añadido la propiedad 'icon' para indicar que icono representa a ese programa. De manera que cuando se generen programas será asignado el siguiente icono que no esté en uso. Con respecto al componente mobile_component, se pasa como argumentos el evento click y la entidad móvil que se quiere mover, cuando el manejador de evento emite el evento click sobre cada instrucción. En el componente instruction_component se han añadido dos manejadores de evento más para las instrucciones Forward y Back. Estas instrucciones permiten al usuario mover el móvil hacia adelante y atrás, respectivamente. Con

[11] `https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Exercises/exercise7`

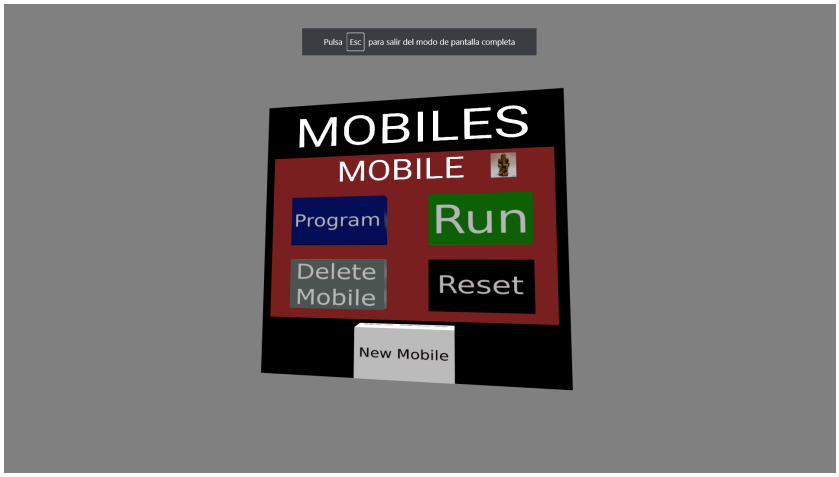

Figura 4.13: Menú de móviles y menú de inicio del primer móvil

repecto al componente button, se han añadido varias entidades botón. Por tanto, se han creado nuevos manejadores de evento para cada uno de ellos. Los manejadores de evento eventButtonHandlerForward y eventButtonHandlerBack generan instrucciones hacia adelante y hacia atrás, respectivamente. El manejador eventButtonHandlerDeleteMobile elimina tanto al móvil como su panel de configuración. El botón New Mobile crea un nuevo móvil y su panel de configuración. Los paneles de configuración se crean uno debajo del otro al igual que los programas y los móviles se crean posicionándolos a la izquierda del último creado o del que debería de ir pues el anterior puede haber sido eliminado antes. El botón Program trabaja con el manejador de evento eventButtonHandlerProgram, el cual crea un menú a la izquierda del panel de configuración con el mismo número de programas que hayan creados. Cada uno de los botones de ese nuevo panel trabajan con el manejador de evento eventButtonHandlerChangeProgram. Dicho manejador cambia el programa en uso del correspondiente móvil y cambia el icono de la placa del panel de configuración al icono del programa seleccionado. En definitiva, el botón Program permite al usuario cambiar de programa a un móvil.

4.5. Iteración 4

Esta es la última iteración del proceso de desarrollo. En ella nos encargaremos de dar el toque definitivo a la estructura final del DOM, mejoraremos el código y ajustaremos el escenario a las gafas VR.

Figura 4.14: Menú de programas y menú de inicio del primer programa

4.5.1. Objetivos

Los objetivos de esta iteración son dar mayor visibilidad a la escena, simplificar y mejorar el código, mejorar la estructura del DOM para que los componentes sean lo más independientes posibles unos de otros y adaptar la escena a las gafas VR.

4.5.2. Primera Fase

En esta fase[12], se trabajó sobre dos archivos, que son los siguientes:

- components.js
- exercise-8.html

En el HTML hemos realizado una serie de cambios con el fin de dar mayor visibilidad a la escena.

En cuanto a la estética de la escena, anteriormente definimos en el HTML la entidad cielo (a-sky) con un tono gris como en la siguiente línea de ejemplo.

```
<a-sky color="grey"></a-sky>
```

Ahora se ha utilizado un escenario ya creado el cual hemos definido en el HTML como una entidad mediante la siguiente línea de código:

[12] https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Exercises/exercise8

```
<a-entity environment="preset: tron"></a-entity>
```

En cuanto a los móviles, hasta ahora eran cajas. En su defecto han sido cambiados por drones. Para ello los hemos añadido en los assets del HTML. Algunos de ellos vienen predefinidos con movimientos propios. Este es un ejemplo de inserción de uno de los drones en assets:

```
<a-asset-item id="object1" class="object" src="../Objects/object1/scene.gltf"></a-asset-item>
```

Por último, hemos hecho uso de Github Corners. Esta herramienta despliega una etiqueta en una esquina de la pantalla con el símbolo de Github. De manera que al hacer click sobre ella nos redirige a la url definida en el atributo 'href'. En nuestro caso redirige al código fuente de esta escena, es decir, al HTML.

En el script no hemos añadido nueva funcionalidad. Hemos hecho cambios de posición y de rotación a la hora de insertar entidades en la escena. Esto es debido a que en esta fase trabajamos con un escena predifinida con unas características, es decir, tiene suelo, luminosidad, obstáculos. Por tanto hemos tenido que adaptarnos a la forma de la escena. Los paneles de configuración crecen de forma vertical a izquierda o derecha en función de si son móviles o programas. Ya que si lo hicieran de forma horizontal se perdería la visión de ellos pues estarían por debajo del suelo. Las instrucciones se generan encima del menú de programas. También comenzamos a simplificar código. En esta fase se simplificó la forma de generar el primer programa. Esta tarea se fue realizando de manera progresiva durante toda la iteración.

4.5.3. Segunda Fase

En esta fase[13], se trabajó sobre dos archivos, que son los siguientes:

- components.js
- exercise-9.html

En el HTML se han eliminado todos los nodos hijo de assets para que sean los propios componentes quienes inserten aquellas texturas y objetos de los cuales vayan a hacer uso. De

[13] https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Exercises/Final/exercise9

este modo el HTML está menos sobrecargado. Además, se ha cambiado la url de Github Corners para que nos redirija al archivo exercise-9.html.

En esta fase se ha seguido simplificando código y se ha hecho una refactorización. Se ha definido una estructura más organizada y más clara en el DOM. Esto ha sido posible gracias a los cambios realizados en el script. Como decía antes la estructura es más amplia y mejor organizada. Esto implica la creación de nuevos componentes. Cada uno de ellos inicializa e inserta en el DOM todo lo referente a su entidad y a sus nodos hijos. También se ha hecho un mayor uso de funciones para desarrollar un código más legible y mejor optimizado. Además, se ha separado el menú de móviles del menú de cada móvil.

4.5.4. Tercera Fase: Adaptación de los paramétros a las gafas VR

En esta fase[14] se trabajó sobre el archivo:

- exercise-9.html

La idea de esta fase era depurar y corregir todos aquellos detalles o problemas que se pudieran dar en el entorno de realidad virtual, es decir, realizar los ajustes necesarios para obtener la mejor experiencia posble de usuario con las gafas VR y los triggers.

En esta fase únicamente se hicieron cambios en el HTML, ya que los comportamientos de las entidades eran los esperados. De entrada nos dimos cuenta de que habían que realizar varios ajustes. Para empezar queríamos suavizar los bordes para que fueran más diagonales o curvados. Para ello habilitamos antialiasing como podemos ver en esta línea de código:

```
<a-scene physics="debug: true" id="scene" renderer="antialias: true">
```

También, hubo que realizar ajustes de la posición de la cámara y de los widgets debido a que los mandos en el entorno de realidad virtual no se mostraban a la altura de las manos. Se puso como puntero láser el mando correspondiente a la mano derecha. De manera que al apuntar con el láser en cualquier botón de la escena y hacer click en el botón del mando podemos interactuar con la escena.

[14] https://github.com/AlvaroLopezGarcia/a-frame-pruebas/blob/master/Exercises/Final/exercise9-oculus

Capítulo 5

Resultados

El resultado final de este proyecto es una biblioteca configurable de realidad virtual. Nos permite generar interfaces a nuestro gusto en el navegador. Podemos cambiar el escenario de manera sencilla, la forma de los objetos móviles, los iconos, las formas de los paneles y de los botones, y la distribución de los elementos en la escena. Además, dicha interfaz de programación permite realizar programas sencillos en realidad virtual y hacer uso de ellos sin tener conocimientos técnicos de programación. Como se definió en los objetivos.

En cuanto a la estructura, debemos hablar del HTML y del DOM una vez ha sido inicializada la escena.

```
<a-scene physics="debug: true" id="scene" renderer="antialias: true">
    <a-assets id="elements"></a-assets>

    <a-entity environment="preset: tron"></a-entity>

    <!--USER POSITION-->
    <!-- Camera rig, with a camera and a cylynder -->
    <a-entity movement-controls="fly: true" position="0 0 .2" rotation="0 0 0">
    <a-entity camera position="0 1.6 0" look-controls wasd-controls laser-controls="hand: right">
    </a-entity>
    <a-entity oculus-go-controls x-button-listener cursor="rayOrigin:mouse"></a-entity>
    </a-entity>

    <a-entity programming-environment  id="programming-environment" position="2.1 3 -5.9">
    </a-entity>

    <a-entity mobiles-environment id="mobiles-environment"></a-entity>

</a-scene>
```

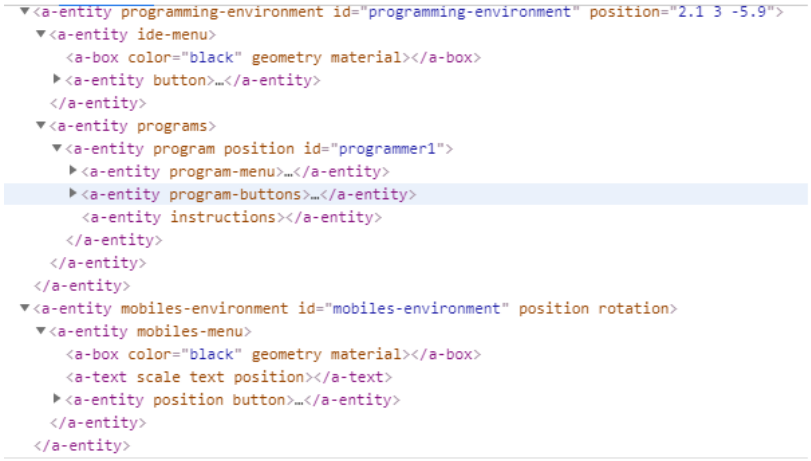

Figura 5.1: Estructura DOM: IDE y Móviles

Como podemos observar este es parte del cuerpo del HTML de la última versión realizada. En él podemos ver que de entrada hay dos componentes:

- programming-environment: este componente se encargará de inicializar todas las entidades referentes al IDE y al primer programa. Además, se encargará de asociar determinados componentes a entidades o nodos hijos para que generen toda la estructura inicial en el DOM. Por tanto, todo lo referente a los programas y a las instrucciones quedarán como nodos hijos de dicha entidad.

- mobiles-environment: este componente se encargará de inicializar todo aquello referente al menú de móviles, al menú del primer móvil y al primer dron. Además, se encargará de asociar determinados componentes a entidades o nodos hijos para que generen toda la estructura inicial en el DOM.

Una vez que se ha inicializado la escena podemos ver en la figura 5.1 y en la figura 5.2 la estructura completa. Por tanto, tenemos componentes que trabajan sobre la estructura de móviles, otros que trabajan en la estructura de programas y otros que en ambos. Utilizamos una serie de componentes que son los siguientes:

1. Componentes usados para el menú del IDE:

    - programming-environment

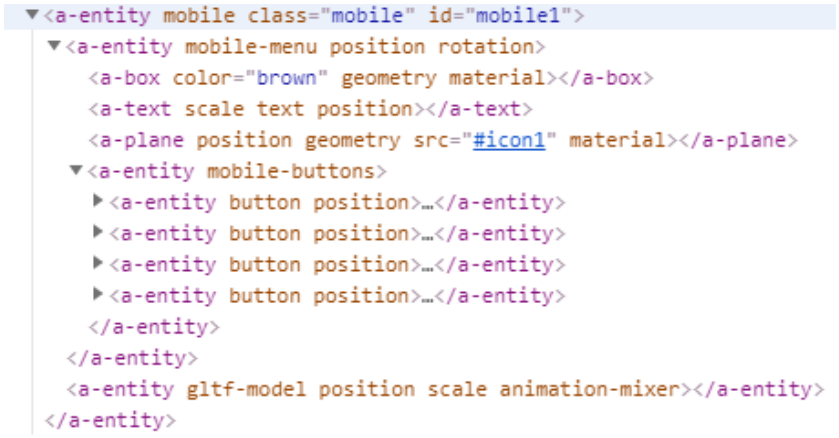

Figura 5.2: Estructura DOM: Móvil

- ide-menu: este componente se encargará de crear todos los elementos visibles del IDE (el botón y el panel) y la entidad que contendrá todos los programas que se generen a lo largo de la interacción.
- programs: es el componente encargado de insertar en el DOM las texturas e iconos a usar por los programas y el primer programa.
- button

2. Componentes usados para cada programa:

   - program: este componente crea toda la estructura de un programa y la inserta en el DOM.
   - program-menu: este componente crea e inserta en el DOM el panel del programa y el texto.
   - program-buttons: este componente crea todos los botones que tiene un menú de programa e inserta en el DOM las texturas correspondientes a sus botones.
   - instructions: este componente únicamente indica la entidad que engloba todas las instrucciones del programa.
   - instruction: este componente mueve el dron en una dirección en función del tipo de instrucción que sea.
   - button

3. Componentes usados para el menú de móviles:

- mobiles-environment
- mobiles-menu: este componente crea toda la estructura del menú de móviles.
- button

4. Componentes usados para cada móvil (menú y dron):

   - mobile: este componente se encargará de inicializar todo aquello referente al menú del móvil e insertar el dron en la escena. Además de ejecutar todas las instrucciones del programa que tenga asignado el dron.
   - mobile-menu: este componente se encargará de crear el menú del móvil.
   - mobile-buttons: este componente se encargará de crear los botones del menú del móvil y de insertar las texturas a usar en los assets del DOM.
   - button

En cuanto a los componentes usados, el componente 'instruction' no se encuentra en ninguna entidad cuando se acaba de inicializar la escena completa. Estará en el DOM junto a su entidad cuando generemos alguna instrucción. El componente button se usa varias veces debido a que todos los botones usan dicho componente sea la función que tenga el botón. Este componente se encarga de dar una funcionalidad específica a cada botón.

Antes de enseñar un ejemplo de ejecución debemos comprender que es cada elemento y para qué sirve. Debemos diferenciar entre móviles y programas. Los móviles son los objetos que vamos a modificar (moverlos en el espacio tridimensional). Estos objetos serán modificados a través de los programas, que serán ejecutados por el usuario. En la interfaz definida trabajamos con direcciones, es decir, los programas modificarán la posición de los móviles. Estas direcciones son instrucciones que crean los programas. Dichas instrucciones serán ejecutadas en el orden que fueron creadas. Es importante saber que la posición de un móvil no puede ser modificada por varios programas de forma simultánea. El móvil está hecho para que solamente pueda hacer uso de un programa. En caso de querer hacer uso de otro, tendrá que cambiar de programa.

La interfaz tiene dos modos de uso:

- El primero es mediante un ordenador ejecutándolo en el navegador. Con el ratón se mueve

la cámara e interactuamos con los botones haciendo click, y con las flechas se mueve el usuario.

- El segundo es mediante las gafas VR y los mandos ejecuntándolo en el navegador. Con las gafas movemos la cámara y con los mandos usamos un puntero láser. Apuntamos a un objetivo y hacemos click en el mando derecho para interactuar con los botones de la escena. El desplazamiento en este caso se puede realizar o desplazándote tú físicamente o haciendo uso del joystick del mando.

A continuación, veremos algunas muestras del funcionamiento de la interfaz:

- Como podemos observar en la figura 5.3 disponemos de varios elementos en la escena, es decir, tres paneles y un objeto móvil (dron). El panel de la derecha es el IDE (Integrated Development Environment), el cual contiene un programa. El programa será diferenciado del resto de programas por el icono que le representa. Dicho icono es la imagen de fondo del panel de configuración. En el centro de la escena tenemos el panel de configuración de móviles, el cual se encargará de generar nuevos objetos móvil y su correpondiente panel de configuración de móvil. A la izquierda vemos el panel de configuración del dron. El primer dron sabemos que utiliza el primer programa debido a que su panel de configuración en la esquina superior derecha muestra dicho icono.

- Con respecto al IDE, su función es generar nuevos programas. Si hacemos click sobre el botón New Program crearemos un nuevo programa. Cuando generemos nuevos programas el menú de IDE se ampliará y cuando eliminemos un programa se reducirá sin dejar espacios en blanco. Véase la figura 5.4.

- Con repecto al panel de móviles, su función es generar nuevos móviles. Si hacemos click sobre el botón New Mobile crearemos un nuevo móvil. Véase la figura 5.5.

- Si eliminamos el primer programa veremos que solamente tendremos un programa y que el primer móvil ya no tiene ningún programa asociado. Esto lo sabemos pues la placa del panel de configuración del móvil está en blanco. Para ello hemos hecho click en el botón Delete Program de dicho programa. Véase la figura 5.6.

- Si queremos escoger o cambiar de programa tenemos que hacer uso del botón Program del panel de configuración del móvil. Veremos en la figura 5.7 que se ha abierto otro menú que te permite escoger cualquier programa, donde el botón tiene la misma textura que el icono del programa. En este caso solamente tenemos uno. Por tanto solamente te da una opción a elegir. De manera que cuando hagamos click el primer móvil habrá escogido ese programa. Es importante mencionar que no se pueden escoger programas en varios móviles de manera simultánea y que un programa puede ser utilizado por varios móviles.

- Si observamos la figura 5.8, podemos ver que hemos generado un nuevo programa y que se lo hemos asignado al segundo móvil. Además veremos que se han generado instrucciones para cada uno de los programas. Las intrucciones de cada programa son aquellas que se situan encima de cada panel de programa. Hemos generado 6 instrucciones hacia la izquierda con el primer programa y 6 instrucciones hacia la derecha con el segundo programa. Para ello hemos hecho click en los botones Left y Right de cada programa.

- Si observamos la figura 5.9, se han desplazado los dos móviles. Uno hacia la izquierda y el otro hacia la derecha. Esto es debido a que hemos hecho click sobre los botones Run. Por tanto, se han ejecutado las instrucciones de ambos programas.

- Si hacemos uso de los botones Reset, los móviles volverán a su posición original, es decir, como en la figura 5.8. Se intercambiarán las posiciones.

- En la siguiente figura 5.10 podemos ver como se eliminaron las isntrucciones del primer programa mediante el botón Delete Instructions.

- Por último, si observamos la figura 5.11 se ha eliminado el primer móvil mediante el botón Delete Mobile.

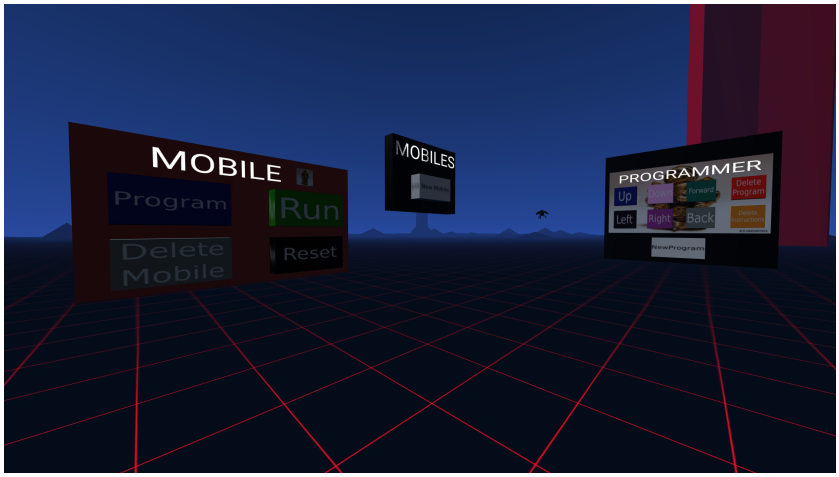

Figura 5.3: Escena inicial

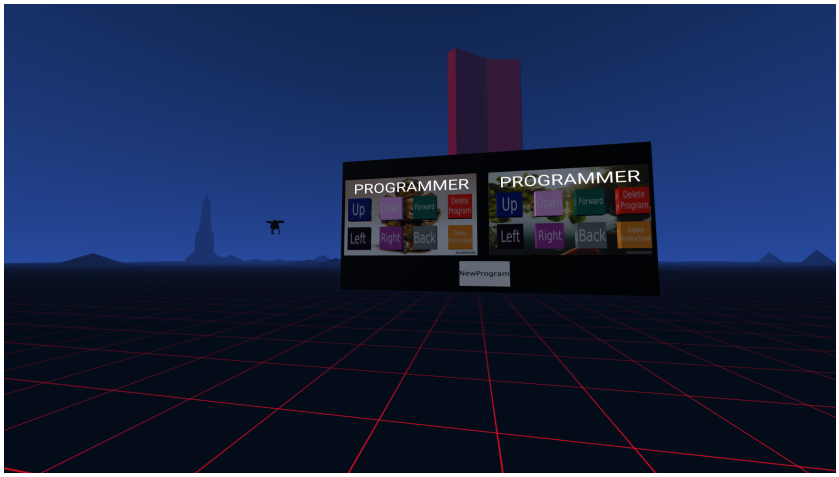

Figura 5.4: Nuevo programa

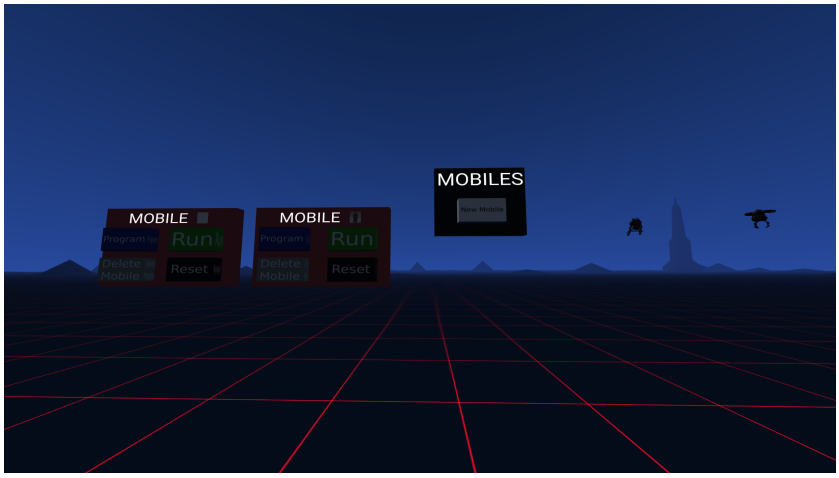

Figura 5.5: Nuevo móvil

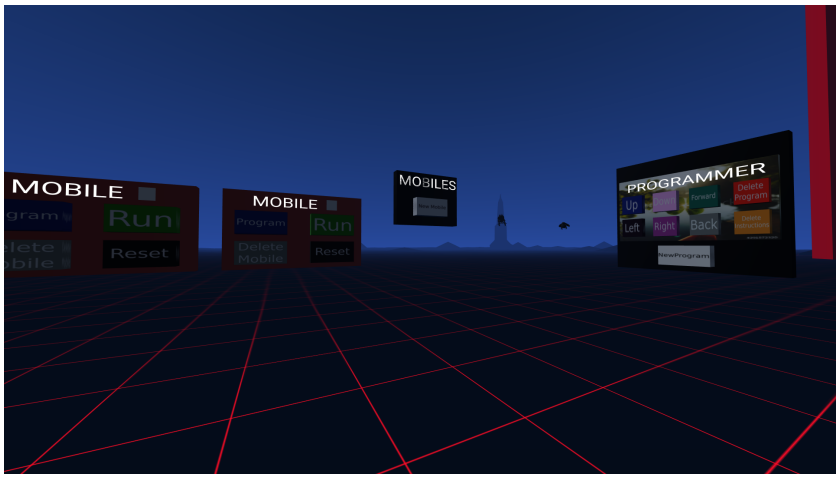

Figura 5.6: Eliminamos un programa

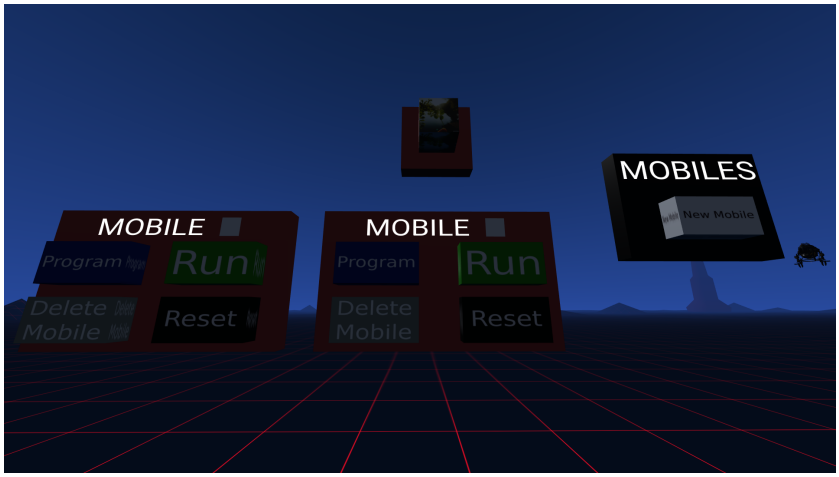

Figura 5.7: Elegimos un programa

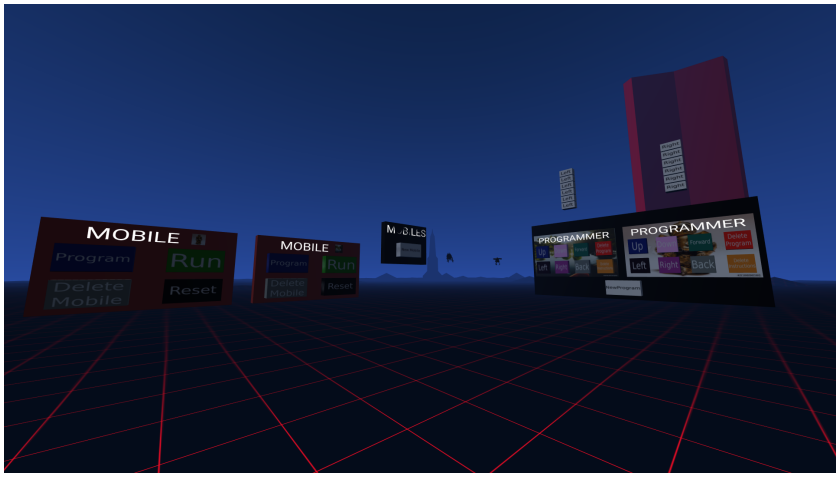

Figura 5.8: Creamos otro programa, generamos instrucciones y escogemos el segundo programa para el segundo móvil

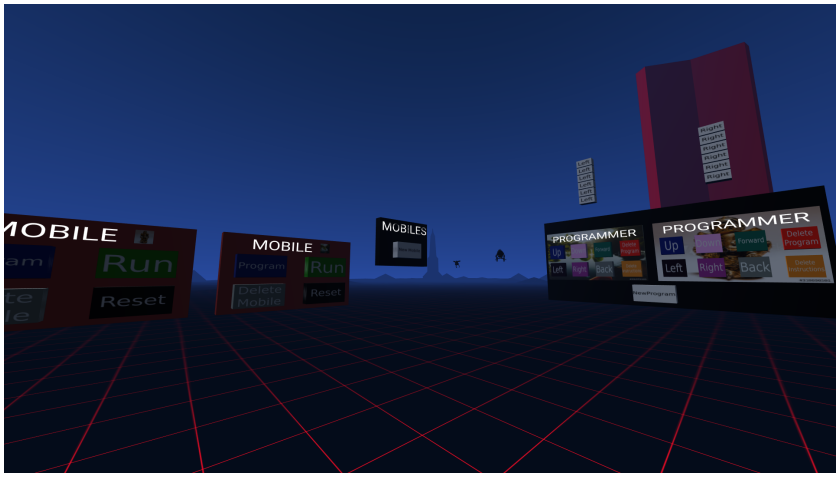

Figura 5.9: Ejecutamos los programas mediante los botones Run

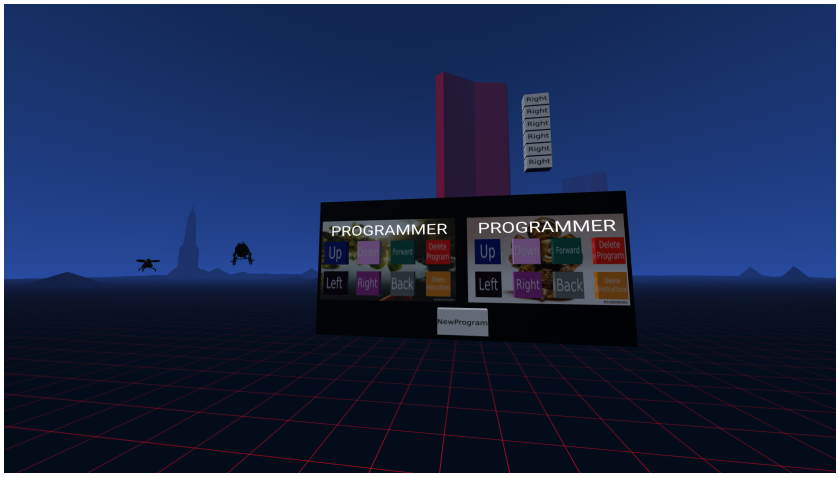

Figura 5.10: Eliminamos instrucciones del primer programa mediante el botón Delete Instructions

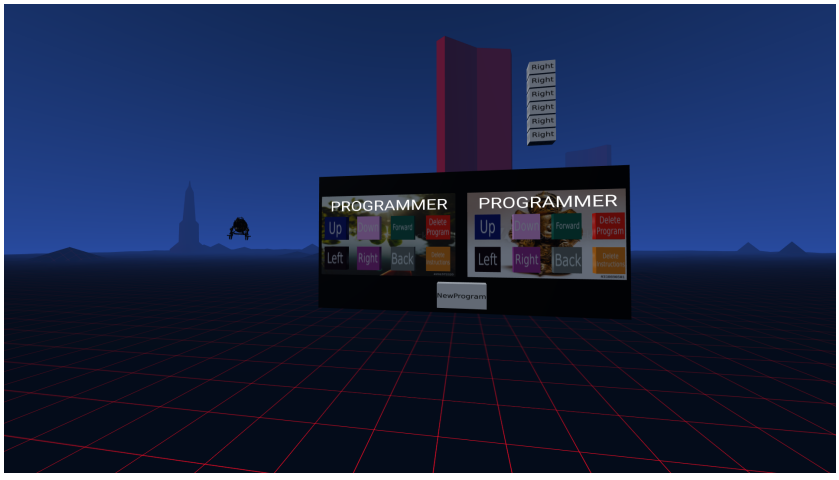

Figura 5.11: Eliminamos el primer móvil mediante el botón Delete Mobile

Capítulo 6

Conclusiones

6.1. Consecución de objetivos

En este apartado hablaremos acerca de los objetivos conseguidos y los problemas que han surgido.

En términos generales hemos logrado los objetivos buscados. Hemos logrado desarrollar una biblioteca configurable que permite generar interfaces de programación en realidad virtual en el navegador. Uno de los objetivos más perseguidos era hacer ver al usuario lo que es un programa y lo que conlleva programar. Todo ello dentro de un marco amigable y divulgativo, y con una serie de herramientas simples de usar. En nuestra interfaz podemos ver un programa como algo tridimensional. En nuestro caso como un conjunto de bloques (instrucciones) ordenados por orden de creación. Además, podemos ver que estos programas se pueden aplicar a otros objetos. En nuestro caso a drones (móviles).

Un objetivo era realizar dos módulos claramente diferenciados e independientes. Para ello hubo que hacer un refactorización, donde se amplió la estructura en el DOM para reorganizar todas las entidades. Esto se hizo a través del desarrollo de nuevos componentes.

Por otra parte hemos conseguido dar al usuario libertad de movimiento para que pueda moverse por toda la escena.

Otro objetivo era realizar un código libre y accesible. Para ello hemos alojado el proyecto en la plataforma Github.

Por último, se buscaba un uso sencillo de la interfaz. Como hemos visto en el apartado de resultados hay dos modos de uso. En ambos casos es sencillo de usar pues interactuamos con

los elementos de la interfaz mediante botones.

6.2. Aplicación de lo aprendido

Durante el grado cursado se aprenden conocimientos de diferentes ramas. Puesto que este proyecto se trata de un trabajo de ingeniería de software hablaremos de aquello relacionado con este ámbito.

Todas aquellas asignaturas cursadas relacionadas con la programación han sido determinantes para el desarrollo de este proyecto. Gracias a ellas he ido adquiriendo ese pensamiento lógico que hay que tener para poder programar. Teniendo esa base he aprendido diferentes lenguajes de programación. Cada uno aplicado a niveles de abstracción diferentes y orientados a diferentes ámbitos. De los aprendidos en asignaturas cursadas han sido determinantes HTML (lenguaje de marcado) y JavaScript. Esto es debido a que los componentes han sido desarrollados en JavaScript y a que la interfaz manipula constantemente la estructura del DOM, lo que hace imprescindible tener unos conocimientos mínimos de HTML.

6.3. Lecciones aprendidas

Durante el Trabajo de Fin de Grado he aprendido diferentes lecciones:

1. He aprendido a usar A-Frame. Este framework es la base de este proyecto. Para hacer uso de él he tenido que aprender qué es la estructura ECS mencionada en el capítulo 3 en el apartado de A-Frame. Una vez sabido esto, he tenido que aprender cómo se genera un entorno de realidad virtual en el navegador haciendo uso de HTML mediante etiquetas propias de A-Frame. También he aprendido que los componentes de A-Frame se desarrollan mediante JavaScript.

2. He aprendido a desarrollar componentes en A-Frame.

3. He aprendido a manipular el DOM haciendo uso de las herramientas del DOM mediante JavaScript.

4. He aprendido a evitar la propagación de eventos de nodos padre a hijos y viceversa.

5. He aprendido a utilizar la consola del navegador. Esto ha sido muy útil pues me ha permitido seguir el hilo de ejecución e ir comprobando los cambios hechos en el DOM.

6.4. Trabajos futuros

En este apartado veremos cambios posibles o recomendadados de cara a seguir con el desarrollo de este proyecto.

1. Ralentizar el movimiento de los drones para que al ejecutar sus programas no se muevan tan rápido de una posición a otra.

2. Cambiar el modo de uso de la interfaz para que los botones podamos presionarlos con las manos en vez de apuntando con el puntero del mando y haciendo click.

3. Reducir el tamaño de los paneles para que podamos cogerlos con las manos.

Bibliografía

[1] GUSTAVO B., *¿Qué es Github?*, https://www.hostinger.es/tutoriales/que-es-github/ (última vez visitada en Abr 2, 2020)

[2] ARKAITZGARRO, *HTML5 CAPÍTULO 1 ¿QUÉ ES HTML5?*, https://www.arkaitzgarro.com/html5/capitulo-1.html (última vez visitada en Abr 2, 2020)

[3] MOZILLA, *¿Qué es CSS?*, https://developer.mozilla.org/en-US/docs/Learn/CSS (última vez visitada en Abr 2, 2020)

[4] ROXANA FALASCO, *Guía definitiva Sublime Text 2*, https://web.archive.org/web/20180625124403/http://falasco.org/guia-definitiva-sublime-text-2 (última vez visitada en Abr 2, 2020)

[5] MOZILLA, *¿Qué es JavaScript?*, https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu_es_JavaScript (última vez visitada en Abr 2, 2020)

[6] A-FRAME, *¿Qué es A-Frame?*, https://aframe.io/docs/1.0.0/introduction/ (última vez visitada en Abr 2, 2020)

[7] A-FRAME, *A-Frame: Developing with three.js*, https://aframe.io/docs/1.0.0/introduction/developing-with-threejs.html (última vez visitada en Abr 2, 2020)

[8] MOZILLA, *WebVR API*, https://developer.mozilla.org/en-US/docs/Web/API/WebVR_API (última vez visitada en Abr 2, 2020)

[9] MOZILLA, *WebGL API*, https://developer.mozilla.org/es/docs/Web/API/WebGL_API (última vez visitada en Abr 2, 2020)

[10] WEBSA100, *¿Qué es Gimp y para qué sirve?*, https://www.websa100.com/blog/que-es-gimp-y-para-que-sirve/ (última vez visitada en Abr 2, 2020)

[11] JOSÉ MARÍA LÓPEZ, *Las ventajas de usar LaTeX en tus documentos*, https://blogthinkbig.com/ventajas-latex-editar-documentos (última vez visitada en Abr 2, 2020)

[12] KEN SCHWABER Y JEFF SUTHERLAND, *La Guía Definitiva de Scrum: Las Reglas del Juego*, https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-SouthAmerican.pdf (última vez visitada en Abr 7, 2020)