



INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Curso Académico 2019/2020

Trabajo Fin de Grado

INTERFACES FOR BUILDING SCENES IN VIRTUAL REALITY

Autor : Javier Jesús Bravo Donaire

Tutor : Dr. Jesús María González Barahona

Trabajo Fin de Grado/Máster

Interfaces for Building Scenes in Virtual Reality

Autor : Javier Jesús Bravo Donaire

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
mi familia / mis amigos / mis compañeros*

Agradecimientos

Página de agradecimientos en proceso.

Resumen

Resumen en proceso. Mejor escribirlo al final.

Summary

Página de resumen escrito en inglés. Al igual que el resumen, se escribe lo último.

Indice General

1	Introducción	1
1.1	Contexto	1
1.2	Objetivos principales	1
1.3	Objetivos secundarios	1
1.4	Planificación temporal	1
1.5	Estructura de la memoria	2
2	Tecnologías	5
2.1	Aframe	5
2.2	HTML5	5
2.3	JavaScript	5
2.4	WebGL	6
2.5	WorldGL	6
2.6	Three.js	6
2.7	GitHub	6
2.8	LaTeX	8
3	Diseño e implementación	11
3.1	Sprint 1	13
3.1.1	Objetivo	13
3.1.2	Desarrollo	13
3.1.3	Funcionalidades añadidas	18
3.1.4	Resultado	19
3.2	Sprint 2	19

3.3	Sprint 3	19
3.4	Sprint 4	19
4	Resultado Final	21
4.1	Manual de Usuario	21
4.2	Arquitectura resultante	21
5	Conclusiones	23
5.1	Consecución de objetivos	23
5.2	Aplicación de lo aprendido	23
5.3	Lecciones aprendidas	23
5.4	Trabajos futuros	24
	Bibliography	25

Lista de Figuras

2.1	Estructura de ramas Git	8
3.1	Estructura de la metodología Scrum	12
3.2	Punto de partida Minijuego	14
3.3	Plataformas para subir de nivel	14
3.4	Piedras que bloquean el camino y bloque gigante	15
3.5	Obstáculo de baldosas con movimiento	16
3.6	Escena victoria	20

Capítulo 1

Introducción

Introducción en la que se nombre un poco las tecnologías empleadas y los objetivos de este.

1.1 Contexto

Aquí se encuentra el contexto basado en cosas que se han creado con la misma tecnología.

1.2 Objetivos principales

Sección en la que se expone el objetivo principal del proyecto.

1.3 Objetivos secundarios

Objetivos secundarios, como futuras implementaciones siguiendo el actual trabajo.

1.4 Planificación temporal

Este proyecto se inició con la apertura del segundo semestre de curso, y se ha ido desarrollando mientras terminaba este y realizaba las prácticas en empresa. Para ello, seguimos el modelo Scrum, en el cual, el proyecto se divide en distintas etapas denominadas "Sprints". Para distinguir estas etapas, manteníamos reuniones (al principio presenciales y más tarde por videoconferencia) en las que se resolvían dudas y se aclaraban los distintos objetivos a conseguir en cada

”Sprint”. La duración de este han sido un total de 9 meses, en los cuales ha habido temporadas más activas y otras más calmadas debido a exámenes o trabajo, pero en todas ellas el periodo de trabajo era por las tardes después de clase, y sobre todo los fines de semana. En general, todo el proceso de creación de este proyecto se puede simplificar en cuatro fases:

- Decisión del objetivo. Esta fase consta de las reuniones que llevé a cabo mi tutor Jesús. En ellas, se hablaba de las distintas corrientes de estudio que se podían seguir, decantándome al final por la presente.
- Adaptación a la tecnología. Para adentrarme en las tecnologías mencionadas más adelante en el capítulo 2, decidimos realizar un minijuego que me permitiese aprender el funcionamiento de Aframe 2.1 o aprender lenguajes de programación como JavaScript 2.3.
- Ejecución de los objetivos. Las distintas reuniones marcaban el inicio y final de las distintas etapas con las que avanzaba el proyecto. Es la fase más amplia de todas ya que es en la que mas documentación es necesaria para avanzar y más tiempo se invierte en la escritura de código, pruebas, etc.
- Redacción de la memoria.

1.5 Estructura de la memoria

Para una correcta lectura del presente proyecto, se aclara la estructura que se sigue a continuación:

- En este primer capítulo se presenta una introducción al proyecto, exponiendo sus objetivos y contexto actual, así como la planificación temporal de todo el proyecto.
- En el capítulo 2 se muestran las distintas tecnologías que se enlazan a lo largo del proyecto con algunos ejemplos de uso.
- A continuación, se presenta el proceso de desarrollo en el capítulo 3. Además, en él se explica detalladamente el modelo Scrum, los diferentes Sprints y tanto los problemas como objetivos que se van solucionando.

- El capítulo 4 muestra el resultado final desde dos distintas perspectivas, una en la que se expone una guía para el usuario y otra más técnica donde se explica la arquitectura resultante, focalizándose en los componentes que implementa la escena.
- Las conclusiones quedan recogidas en el capítulo 5, donde se analizará todo lo aprendido durante este periodo, los objetivos logrados y los problemas resueltos.

Capítulo 2

Tecnologías

2.1 Aframe

Hemos hablado de cómo incluir figuras. Pero no hemos dicho nada de tablas. A mí me gustan las tablas. Mucho.

2.2 HTML5

Aquí describo una nueva tecnología. Por ejemplo, HTML5.

2.3 JavaScript

JavaScript[3] es un lenguaje de programación interpretado, lo que significa que no necesita de compilación para su ejecución. Su motivo consiste en infundir dinamicidad a nuestras aplicaciones web mediante distintas animaciones, manejo de eventos y modificaciones del DOM en general. Como cualquier otro lenguaje de programación puede realizar operaciones, tratamiento de strings, almacenamiento de variables y métodos algorítmicos.

Acceder al DOM gracias a JavaScript se convierte en una tarea sencilla además de potente. El código HTML es estático, pero con JavaScript podemos alterarlo en vivo con distintos eventos. Gracias a sentencias como la siguiente podemos acceder a cualquier elemento, por ejemplo, al título:

```
document.getElementsByTagName("h1");
```

Dicho esto, igual que accedemos al título, podemos acceder a un componente de nuestra escena en Aframe: cubos, cilindros, el cielo o incluso a la cámara de nuestra escena. Nos aporta infinidad de opciones: crear botones que activen eventos, movimiento a distintas entidades, etc. Junto con todo lo que nos ofrece Aframe, JavaScript será la base este proyecto.

Habitualmente, el código JavaScript se encuentra en el lado del cliente, pero debido a su gran popularidad se creo una versión con la capacidad de ser ejecutada en el lado del servidor, Node.js.[4] Al estar basado en JavaScript, mantiene las ventajas de este, sigue siendo software libre y además, es asíncrono. Su función principal es el manejo de solicitudes de entrada y la salida de respuesta, llegando incluso a poder realizar una solicitud HTTP.

2.4 WebGL

Aqui describo una nueva tecnología. Por ejemplo, webgl.

2.5 WorldGL

Aqui describo una nueva tecnología. Por ejemplo, worldgl.

2.6 Three.js

Aqui describo una nueva tecnología. Por ejemplo, three.

2.7 GitHub

GitHub[6] es una de las mayores plataformas de almacenamiento de software y proyectos basado en el sistema de control de versiones Git[1], creado por Linus Torvalds¹ en el año 2005.

¹https://es.wikipedia.org/wiki/Linus_Torvalds

El propósito de Git es la eficiencia de producción y el mantenimiento de un registro de los cambios que se producen, con el fin de coordinar el trabajo compartido. Para lograr esto, se aclaran unas normas a seguir que recibe el nombre de Flujo de Trabajo.

Git sigue una estructura de árbol, en la que cada rama representa un flujo de trabajo. La rama principal recibe el nombre de "master", esta rama contiene la versión funcional del proyecto que se esté llevando a cabo, es la rama de producción. Para crear esta rama necesitamos establecer un repositorio en el que poder añadir nuestros archivos y añadirlos. Git tiene una serie de comandos para realizar todas estas acciones, por ejemplo, para crear el repositorio usamos *git init*, y para añadir nuestros archivos (consiguiendo así protegerlos) *git commit* seguido de *git push*.

Para prevenir que se trabaje todo sobre la versión de producción, Git aporta un comando para crear ramas adicionales en las que distintos grupos de trabajo pueden trabajar para evitar modificar los mismos archivos y eliminar sus propios avances, *git branch*. Una vez terminada la funcionalidad en la que se trabaja en una rama aparte, se debe enlazar con la rama maestra, para ello empleamos el comando *git merge*. Se forma así una estructura de árbol como la que podemos observar en la Figura 2.1, en la que cada nodo corresponde con una representación de nuestros archivos comprometidos, lo que se llama "log". Gracias a esto, podemos volver a antiguos nodos del árbol para recuperar versiones anteriores, eliminar otros, etc.

Tomando como base Git, GitHub nos permite visualizar en su plataforma web el flujo de trabajo conseguido. La principal diferencia entre ambos consiste en que GitHub nos aporta una interfaz gráfica de nuestro árbol, mientras que Git solo funciona a través de comandos en una terminal. Asimismo, GitHub añade más herramientas que nos aportan claridad y orden al proyecto:

- Creación de perfiles para cada usuario donde se muestran los repositorios en los que trabaja (tanto propios como conjuntos), junto con estadísticas de trabajo: lenguajes de programación más usados, historial de producción, etc.
- Página de "wiki" para cada repositorio en la que se puede comentar cada apartado de este para que los lectores puedan apoyarse.
- Una sección llamada "issues" donde los usuarios pueden dejar un mensaje comentando cualquier fallo que haya encontrado en nuestro trabajo.

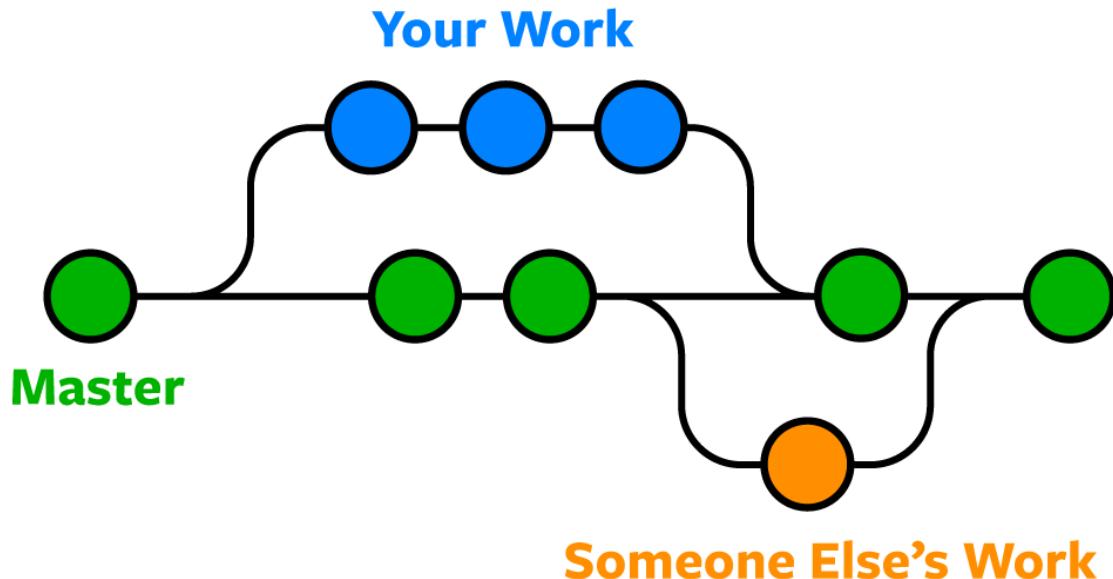


Figure 2.1: Estructura de ramas Git

- La casilla "projects" permite crear tareas que se pueden asignar a distintos grupos de trabajo para tener un control de las funcionalidades que se están creando, así como creación de columnas para ordenar estas tareas según si se encuentran finalizadas, en proceso o por iniciar.
- Capacidad de copiar un repositorio ajeno para trabajar sobre él. Esto es lo que se conoce como "fork", nos permite editar los archivos y en el caso de mejorar alguna funcionalidad, podemos solicitar al autor una unión (merge) con lo que se denomina "pull request".

Todas estas cualidades mostradas, han sido las responsables para decidirme a alojar y proteger mi proyecto en GitHub².

2.8 LaTeX

LaTeX[5] trata de un procesador de textos de software libre típicamente utilizado para textos técnico-científicos. Su objetivo principal es separar las reglas de estilo del contenido. La diferencia más importante con respecto a otros editores de textos es su estructura basada en instrucciones. Entre sus ventajas se encuentran la capacidad de establecer órdenes al inicio de

²<https://github.com/JavierBravoDonaire/A-frame>

nuestro texto que afectarán a este, o la despreocupación a la hora de situar figuras y tablas, ya que como he comentado antes, él se encarga del estilo, nosotros del contenido.

Al tratarse de un software libre su uso es completamente gratuito (además, de que se mantiene siempre en desarrollo) y se encuentra disponible para múltiples sistemas operativos.³

³<https://www.latex-project.org>

Capítulo 3

Diseño e implementación

Para poder explicar todo el proceso de diseño del proyecto, tengo que comenzar hablando de Scrum[2]. Scrum es una metodología de trabajo desarrollada en el año 1993 por Jeff Sutherland¹ y Ken Schwaber², y finalmente formalizada en el año 1995. El objetivo de esta es conseguir un desarrollo ágil y productivo con la mejor organización posible para llegar a un producto completo. En ella, se define que el trabajo realizado debe organizarse en distintos espacios de tiempo denominados iteraciones o "Sprints" en los que se definen una serie de objetivos, y no se da paso al siguiente "Sprint" hasta que se solucionan los problemas y se consigue todo lo planeado en el anterior.

El equipo que emplee esta metodología debería dividirse en tres partes: el Scrum Master, el Equipo de Trabajo y el Cliente. En este caso, el papel de Scrum Master lo lleva a cabo el profesor, Jesús, (aunque también podría adquirir el papel de Cliente) y el papel del Equipo de Trabajo lo representa el alumno, en este caso, yo. El objetivo del Scrum Master consiste en asegurar el entendimiento de las reglas y prácticas llevadas acabo. Además, mantiene una estrecha relación con el Cliente haciendo de intermediario entre este y el Equipo de Trabajo, consiguiendo y aclarando los objetivos propuestos y aportando técnicas para una efectiva producción. Por otro lado, el Equipo de Trabajo esta formado por los profesionales que se encargar del desarrollo y resolución de problemas que propone el Cliente. Son los encargados de la consecución de los objetivos para poner fin a los distintos "Sprints".

Para este caso en particular, cada reunión entre Jesús (Scrum Master/Cliente) y yo (Equipo

¹<https://www.scrumguides.org/jeff.html>

²<https://www.scrumguides.org/ken.html>

Scrum Process

Enter your subhead line here

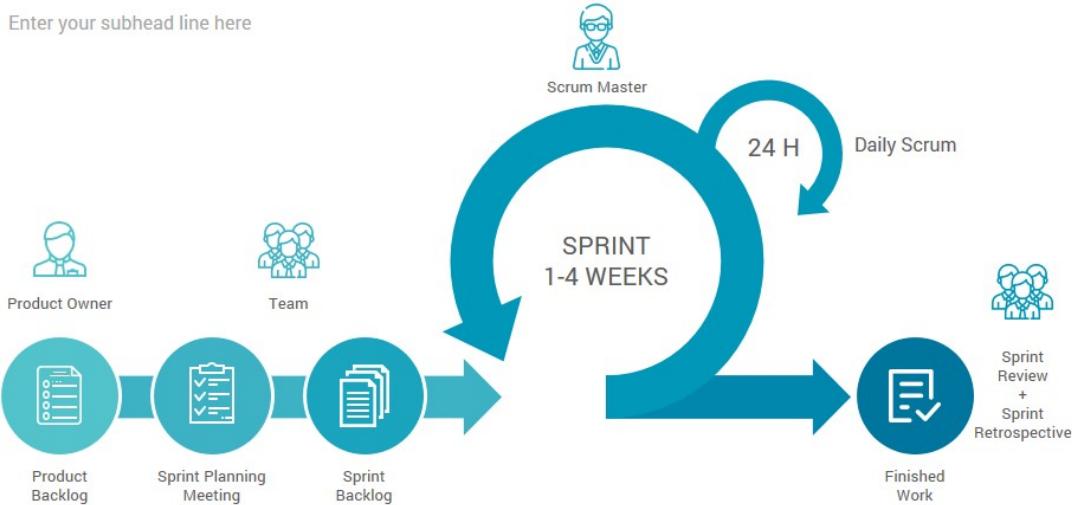


Figure 3.1: Estructura de la metodología Scrum

de Desarrollo), marcarán el inicio y final de cada "Sprint". Inicialmente, una reunión marcará los objetivos a conseguir en ese "Sprint", durante unas semanas se trabaja en ello y finalmente se prepara una nueva reunión. En esta nueva reunión, se hablan de los problemas que han surgido y lo conseguido. Si se han logrado con éxito cada una de las metas propuestas, esta reunión será la que dará fin a ese "Sprint" e iniciará el siguiente. Normalmente, en cada "Sprint" se crea una versión funcional que implementa lo conseguido y da paso como versión inicial al siguiente. En caso contrario, se tratan las dudas y problemas surgidos intentando llegar a una solución. Se trabaja en esto durante un periodo y se vuelve a tener la reunión hasta conseguir la versión funcional y dar paso al siguiente "Sprint" hasta llegar a la versión final del producto, como podemos observar en la Figura 3.1.

Por último, algunas de las ventajas del uso de Scrum son las siguientes:

- Los "Sprints" se realizan de manera mensual (o de varias semanas), lo que permite un avance rápido y marcado del trabajo, además de conseguir gradualmente resultados antes del producto final.
- Existe una flexibilidad entre el cliente y el equipo de desarrollo que permite el intercambio de ideas o soluciones de problemas, debido a las reuniones mensuales.

- Rígida definición de plazos y tareas.
- Reducción de la complejidad de un proyecto complejo gracias a su desglose en etapas.
- Mayor satisfacción moral del equipo y por consiguiente, mejora del ambiente de trabajo.

3.1 Sprint 1

Esta iteración comienza con la primera reunión que mantuvimos Jesús y yo. En ella se acordó que para empezar debía iniciarme en Aframe junto con JavaScript (hasta el momento desconocido para mí) y HTML entre otros, por lo que la idea sería crear un minijuego básico que uniese todas las tecnologías.

3.1.1 Objetivo

El objetivo de esta fase es aprender a usar Aframe, en esto se incluye: aprender que son las entidades, crear entidades, crear eventos, crear componentes, dinámicas y físicas, etc. Como JavaScript es desconocido para mí, otro gran objetivo es ir aprendiendo a la vez que lo enlazo con Aframe. El fin del minijuego es hacer un pequeño recorrido en el que tengas que realizar distintas acciones para llegar a la meta.

3.1.2 Desarrollo

La idea es crear un recorrido con distintos obstáculos que implementen distintas acciones y eventos para adecuarme a ellos. El recorrido es el que sigue a continuación:

1. Apareces en un estrecho pasillo con una puerta delante, la cual debes abrir llamando al timbre y evitando los dos obstáculos que tratarán de tirarte como se puede observar en la Figura 3.2.
2. Una vez abiertas las puertas, el pasillo continúa dejando paso a dos plataformas que te permiten llegar al segundo piso (Figura 3.3). Para ello, debes subirte en una de las dos plataformas en el momento correcto.

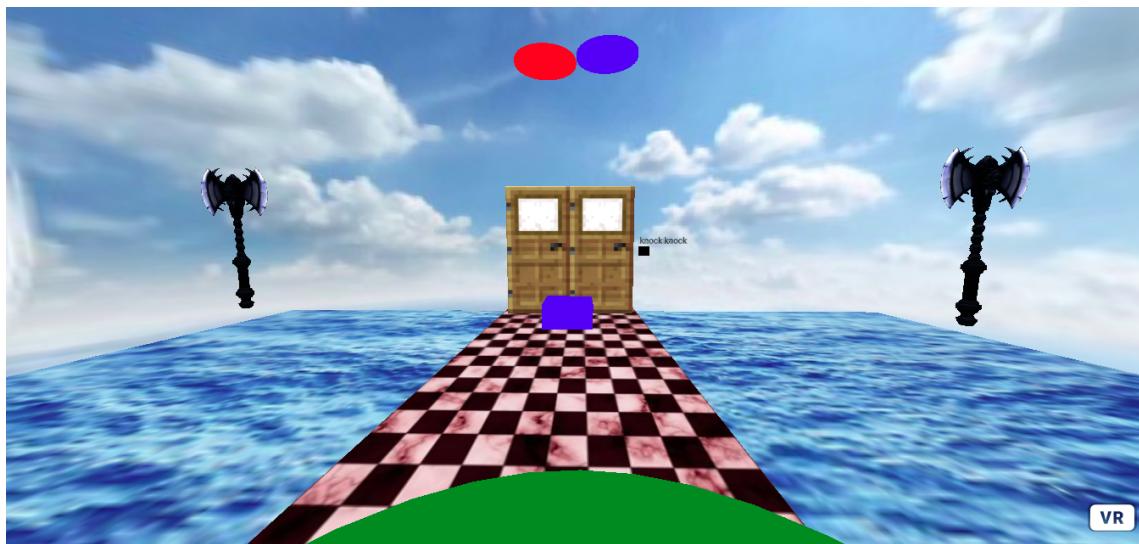


Figure 3.2: Punto de partida Minijuego

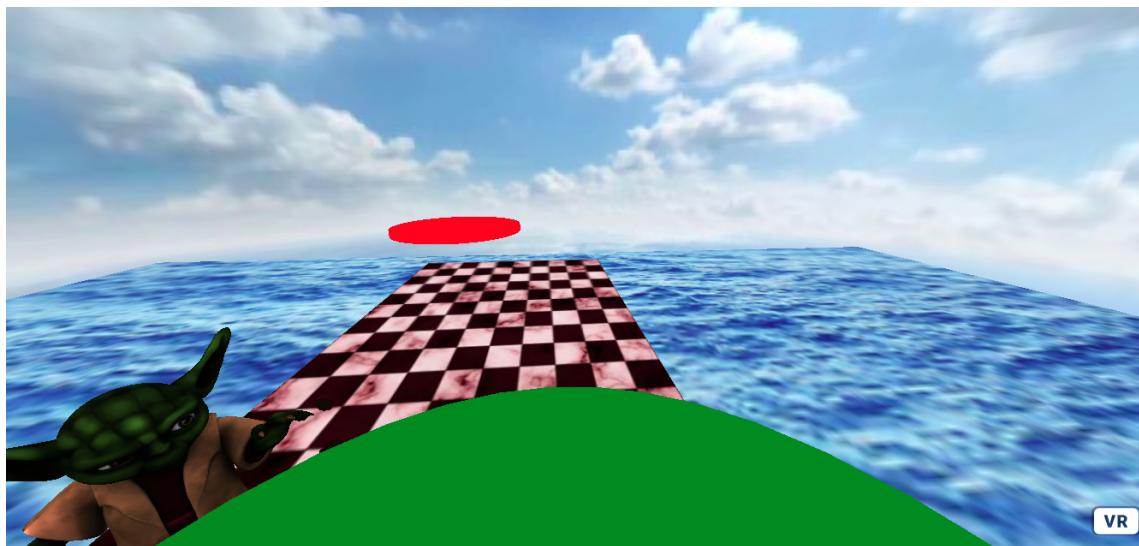


Figure 3.3: Plataformas para subir de nivel

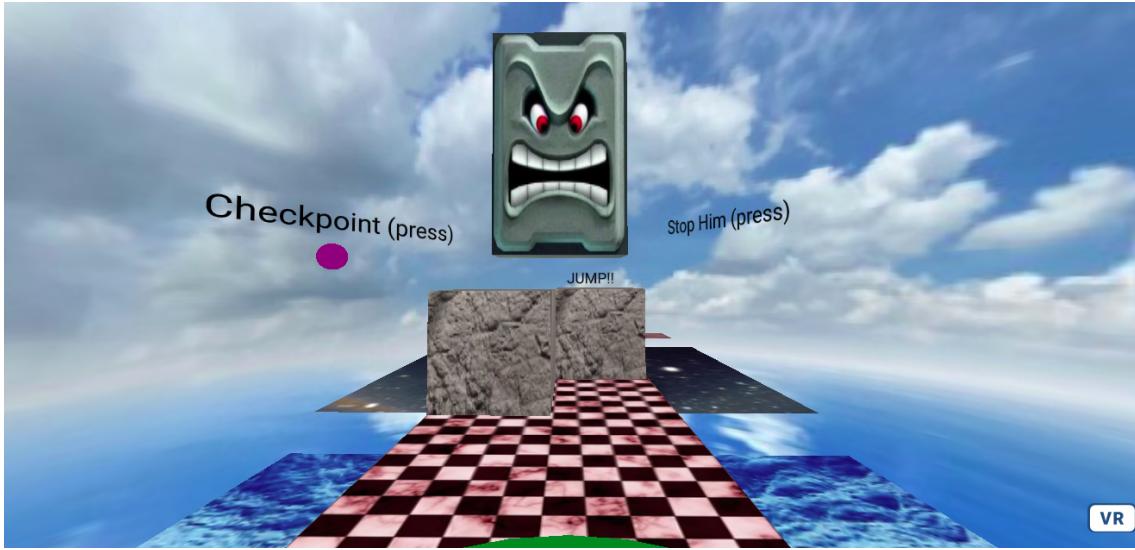


Figure 3.4: Piedras que bloquean el camino y bloque gigante

3. Ya subidos en la plataforma giraremos 180 grados para encontrarnos un nuevo obstáculo en movimiento, la piedra gigante (Figura 3.4). Pero para llegar a ella primero debemos empujar las dos piedras que nos cortan el camino, haciendo así que caigan al vacío.
4. Para conseguir cruzar debemos parar el bloque gigante a nuestros pies, lo que nos dejará ver el siguiente y último reto, las baldosas móviles (Figura 3.5). Cruzando con cuidado llegaremos a la última plataforma gigante.
5. En el centro de esta plataforma nos espera un botón que invoca un evento de felicitación junto con las palabras de fin del juego.

Son muchos obstáculos pero esto permite que utilice el mayor número de entidades y cree los eventos necesarios. Por último, si caes en cualquier momento, volverás al inicio para además añadirle un nivel de dificultad, al fin y al cabo, es un juego.

Para crear este pequeño juego se comienza con un archivo HTML en el que se importan los scripts necesarios de Aframe para poder comenzar a crear entidades, estos son algunos de ellos:

```
<script src="../../assets/js/aframe.min.js"></script>
<script src="../../assets/js/aframe-physics-system.min.js"></script>
```

Gracias a esto podemos empezar a crear nuestras entidades. Las entidades son los distintos objetos que podemos crear como pueden ser un plano, un cilindro o un cubo, entre otros. Para

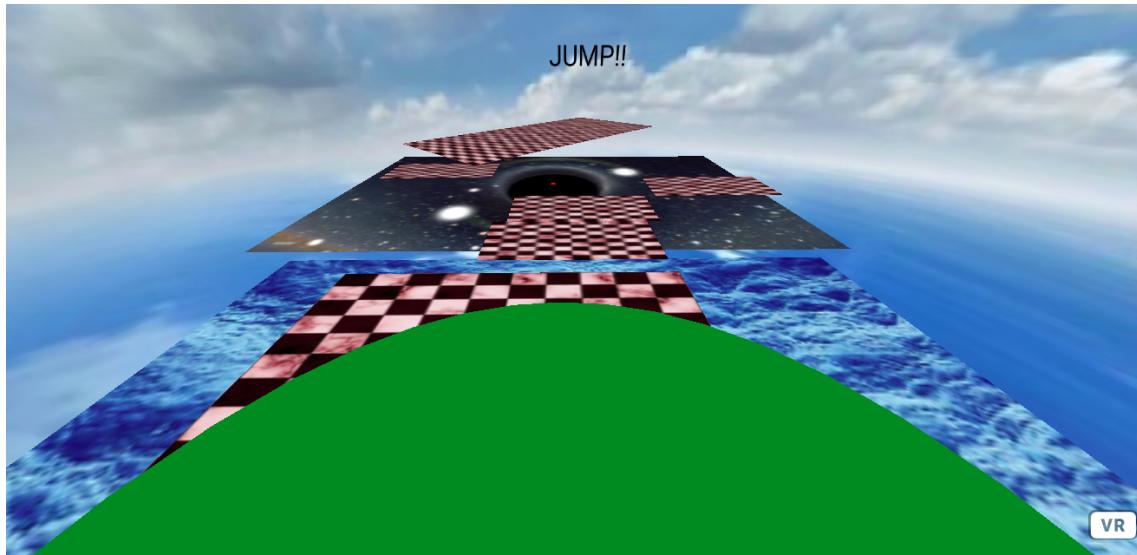


Figure 3.5: Obstáculo de baldosas con movimiento

este caso, el pasillo será un plano, las puertas serán cubos y las plataformas cilindros. Todas las entidades tienen propiedades que pueden ser modificadas dinámicamente, mediante eventos por ejemplo. Modificando las propiedades rotación, ancho y largo del plano, podemos crear nuestro pasillo:

```
<a-plane position="0 0 0" rotation="-90 0 0" width="3.25" height="20"
src="#floor" repeat="1 5" static-body></a-plane>
```

En el apartado "src" podemos asignar una textura a nuestras entidades, mientras que con "repeat" podemos hacer que se repita la textura para que no pierda calidad, o crear incluso un pasillo de baldosas como el visto anteriormente. Siguiendo estas reglas, podemos crear todas las entidades necesarias para nuestra escena. Ahora sólo queda añadir las dinámicas y eventos a estas, al igual que animaciones.

Aframe no nos aporta figuras más complejas que cubos, cilindros, etc. Por eso, podemos crearlos nosotros con distintas aplicaciones de edición 3D en formato gltf³ e importarlas a nuestra escena. Podemos usar distintas páginas web con diseños completamente gratuitos de modelos 3D para nuestras aplicaciones, una de ellas es Sketchfab⁴. Por ejemplo, los diseños del obstáculo de las hachas los encontré en esta página.

³<https://es.wikipedia.org/wiki/GlTF>

⁴<https://sketchfab.com/feed>

Para hablar de los eventos, tomaré como ejemplo el botón para abrir las puertas. El botón es una caja que tiene como propiedad el componente "clickable". Esto permite acceder a el gracias a JavaScript y que cada vez que se posicione el ratón encima de la entidad y se haga click en él, se dispare un nuevo evento. Al capturar ese evento, podemos modificar propiedades de una entidad en tiempo real. El siguiente código muestra un ejemplo de ello:

```
buttonEl.addEventListener('mouseenter', function() {
  buttonEl.setAttribute('color', "gray");
  buttonEl.setAttribute('scale', "0.22 0.22 0.22");
});
```

Así podemos hacer que al pulsar el timbre (que es un cubo), la rotación de las puertas cambien 90 grados (lo que hace que la puerta se abra). Los mismos principios se siguen para hacer que el bloque gigante se pare para que podamos pasar o el botón que activa la escena final del juego.

El movimiento de las hachas, las plataformas que nos suben de piso o las baldosas que nos impiden cruzar es a lo que se llama animación. Una animación nos permite establecer un patrón de movimiento en una entidad (las entidades son estáticas). La animación es otra propiedad de la entidad como puede ser el ancho, se declara como sigue:

```
<a-cylinder class="platform" position="1.5 22 -12"
animation="property: position; dur: 10000; to: 1.5 -1 -12;
dir: alternate; loop: true" static-body></a-cylinder>
```

En este trozo de código declaramos la duración de la animación, la posición final donde acaba, si es infinita y si se alterna o no. Este es el código que usan las plataformas para subir, empiezan en una posición, acaban en otra y gracias al atributo "alternate", cuando llegan a la posición final vuelven a la inicial como si de un ascensor se tratara.

Para completar este apartado, resta explicar los usos de los componentes. Un componente es un comportamiento que se asocia a una entidad, por ejemplo el antes mencionado "clickable". Para conseguir la función de reaparición del jugador cada vez que caía del escenario, creé el siguiente componente, en el que se detallan estas condiciones:

```
AFRAME.registerComponent('restart', {
```

```
// ...
schema: {
  checkpoint: {type: 'string', default: '0 0 9'},
},
tick: function () {
  var audio = document.querySelector('#deathsound');
  var data = this.data;

  if(this.el.getAttribute('position').y < -6) {
    this.el.setAttribute('position', data.checkpoint);
    audio.play();
  }
}
} );
}
```

En este componente aparece la función "tick", la cual se ejecuta cada milisegundo, comprobando si la posición "y" del jugador es menor (para así saber si ha caído) y haciendo que reaparezca en el inicio. Los componentes son herramientas muy potentes, ya que nos permiten crear funciones complejas como esta y simplifica el código. Para asociar un componente a una entidad debemos incluirlo como, ya hemos visto anteriormente, un atributo cualquiera.

3.1.3 Funcionalidades añadidas

Siguiendo todo lo comentado en el anterior apartado, se agregaron unas funcionalidades no tan básicas para complementar el juego.

- **Dinámicas.** Como se puede observar en las figuras anteriores, el jugador tiene alrededor un cilindro verde. Este cilindro es el que marca la caja de colisión del jugador para que no pueda atravesar las puertas, pueda empujar las piedras y pueda subirse en las plataformas sin caer. Estas dinámicas vienen dadas en el script `kinema.js`⁵.

⁵<https://www.npmjs.com/package/kinema/v/0.0.1>

- Aprovechando el componente "restart" para cuando el jugador caiga, añadí una funcionalidad de punto de guardado. Cuando llegas al piso de arriba se puede observar un botón de "checkpoint" (Figura 3.4) para no tener que iniciar desde el principio y reaparecer arriba.
- Efectos de sonido. Se incorporaron efectos de sonido en el evento final del juego, al abrir puertas o al detener el bloque.
- Gafas VR. Por último, se adecuó el minijuego para poder jugarlo con las gafas de realidad virtual. Para ello, se necesitaron nuevos componentes que representasen nuestras manos en la escena y pudiesemos usarlas como si fuesen el ratón. Este componente podemos encontrarlo su propia página de GitHub⁶.

3.1.4 Resultado

Como iniciación a las nuevas tecnologías, el resultado fue notablemente bueno. Pude practicar con muchas de las posibilidades que me ofrecía Aframe mientras aprendía JavaScript. Además, tuve mi primer contacto con las gafas de realidad virtual en las que pude depurar el final de mi juego (Figura 3.6). El minijuego se encuentra alojado en la página de GitHub del proyecto y puede ser jugado⁷. El código JavaScript⁸ se encuentra a su vez alojado en la carpeta correspondiente a este "Sprint" junto a su HTML⁹. Para terminar, se mantuvo una reunión en la que se comentó lo aprendido gracias a este "Sprint" y dió comienzo al siguiente apartado.

3.2 Sprint 2

3.3 Sprint 3

3.4 Sprint 4

⁶<https://github.com/wmurphyrd/aframe-super-hands-component>

⁷<https://javierbravodonaire.github.io/A-frame/Fase%200/my-examples/minigame.html>

⁸EN PROCESO

⁹EN PROCESO



Figure 3.6: Escena victoria

Capítulo 4

Resultado Final

En este capítulo se incluyen los resultados de tu trabajo fin de grado.

Si es una herramienta de análisis lo que has realizado, aquí puedes poner ejemplos de haberla utilizado para que se vea su utilidad.

4.1 Manual de Usuario

4.2 Arquitectura resultante

Capítulo 5

Conclusiones

5.1 Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

```
aspell --lang=es_ES -c memoria.tex
```

5.2 Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a
2. b

5.3 Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. Aquí viene uno.
2. Aquí viene oto.

5.4 Trabajos futuros

Ningún proyecto ni software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

Bibliography

[1] Git manual.

<https://git-scm.com/docs>.

[2] Introduction to scrum.

<https://www.scrum.org/resources/scrum-guide>.

[3] Javascript documentation.

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

[4] Node.js documentation.

<https://nodejs.org/es/docs/>.

[5] L. Lamport. *LATEX: a document preparation system: user's guide and reference manual*.

Addison-wesley, 1994.

[6] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan. Curating github for engineered software projects. *Empirical Software Engineering*, 22(6):3219–3253, 2017.