



Grado en Ingeniería en Tecnologías de la Telecomunicación

Curso Académico 2018/2019

Trabajo Fin de Grado

TALLERES DIDÁCTICOS ORIENTADOS A LA
DIFUSIÓN DE EXPERIENCIAS VR ENTRE
ALUMNOS DE ENSEÑANZAS MEDIAS.

Autor : Jonatan Santana Pero

Tutor : Pedro de las Heras Quirós

Trabajo Fin de Grado/Máster

Título del Trabajo con Letras Capitales para Sustantivos y Adjetivos

Autor : Jonatan Santana Pero

Tutor : Pedro de las Heras Quirós

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 2019

*Dedicado a
mi mismo*

Agradecimientos

AGRADECIMIENTOS

Resumen

La finalidad de este proyecto es la introducción de diferentes tecnologías a alumnos de enseñanzas medias interesados en la realidad virtual.

La tres tecnologías que suponen el núcleo de este proyecto son:

- Beetleblocks.
- A-Frame.
- Microbit.

Para hacerlo, este proyecto esta basado en la ejecución de 3 talleres enfocados en cada una de ellas. Los talleres a su vez forman un todo, con el que conseguiremos una experiencia VR completa.

Para realizar el proyecto, primeramente se procedió a un estudio de cada una de las tecnologías y con la ayuda del tutor se acordó en que la mejor forma de introducirlas es con una serie de talleres.

Para la realización de los talleres se ha escogido un formato en el que los alumnos tengan un camino guiado, pero en los que se deja lugar a la creatividad en determinados puntos.

Índice general

1. Introducción	1
1.1. Experiencias VR	1
1.1.1. Utilidad VR en la educación	2
1.2. Diseño de figuras 3D	2
1.2.1. Entornos VR con figuras en 3D	3
1.3. Microbit	3
1.3.1. Microbit como interfaz de usuario	4
1.4. Estructura de la memoria	5
2. Objetivos	7
2.1. Objetivos	7
2.1.1. Implementar entornos VR en la educación	7
2.2. Motivación	7
2.3. Metodología y Plan de Trabajo	8
2.3.1. Metodología	8
2.3.2. Plan de trabajo	9
3. Creación de experiencias VR	11
3.1. Tecnologías	11
3.2. Diseño 3D	11
3.2.1. Programación mediante bloques	11
3.2.2. Beetleblocks	12
3.2.3. Blender	16
3.3. A-frame	17

3.3.1.	Programación A-frame mediante HTML	17
3.3.2.	Programación A-frame mediante JS	17
3.4.	Microbit	18
3.4.1.	Pyautogui	19
3.4.2.	Bitio	19
3.4.3.	Micropython	20
3.4.4.	Programación Microbit mediante bloques(Makecode)	21
4.	Diseño e implementación de los talleres	23
4.1.	Introducción y propósito de los talleres	23
4.2.	Taller 1 (Creación y exportación de un objeto 3D)	24
4.2.1.	Bettleblocks	24
4.2.2.	Blender	27
4.3.	Taller 2 (Creación de un entorno VR)	29
4.3.1.	Creación de entorno VR con A-Frame	29
4.3.2.	Animación básica con A-frame.	31
4.3.3.	Anexo: Lleva tu experiencia VR a tu smartphone	32
4.4.	Taller 3(Microbit como IU en entorno VR)	34
4.4.1.	Instalación de Bitio	35
4.4.2.	Instalación de Pyautogui	36
4.4.3.	Programación del Microbit	37
5.	Conclusiones	43
5.1.	Consecución de objetivos	43
5.2.	Aplicación de lo aprendido	43
5.3.	Lecciones aprendidas	43
5.4.	Trabajos futuros	44
A.	Manual de usuario	45
	Bibliografía	47

ÍNDICE GENERAL

B. Introducción	49
B.1. Sección	49
B.1.1. Estilo	49
B.2. Estructura de la memoria	50
C. Estado del arte	51
C.1. Sección 1	51

Índice de figuras

1.1. Microbit	4
3.1. Blender	16
3.2. Editor Mu	20
3.3. Makecode	21
4.1. Beetleblocks	25
4.2. Orientación en Beetleblocks	25
4.3. Blender	27
4.4. Funciones del Microbit	34
4.5. Resoluciones	39
4.6. Valores del acelerómetro	40
B.1. Página con enlaces a hilos	50
C.1. Estructura del parser básico	52

ÍNDICE DE FIGURAS

Capítulo 1

Introducción

1.1. Experiencias VR

La realidad virtual, es la sensación de estar inmerso en un entorno con objetos o escenas de apariencia real. El usuario puede sumergirse en imágenes 3D realistas, generadas por ordenador, a través de tecnología como los visores de realidad virtual una experiencia que luego puede enriquecerse con otros dispositivos.

En los últimos años hemos visto un crecimiento en su importancia y uso, se trata de una tecnología relativamente temprana tanto en su concepción como en su aplicación, hasta los años 90 no se consiguió simplificar los grandes simuladores de realidad virtual en dispositivos portátiles, desde cascos de realidad virtual, a gafas en los que los smartphones se encargan de hacer realidad la experiencia.

Entre los años 2015 y 2016 se vivió el gran avance en estos dispositivos al animarse grandes marcas como HTC o Sony a sacar estos dispositivos a la venta para el público.

A la hora de hablar de la realidad virtual nos centraremos en las denominadas experiencias, se trata de entornos en realidad virtual que nos permiten situarnos en medio de la escena, de la que podemos ser espectadores, o interactuar con ella, ayudándonos de sensores o controladores.

Si bien es cierto que actualmente el mayor uso de la realidad virtual va dirigido a videojuegos, también es empleada en campos como el de la medicina o la educación.

1.1.1. Utilidad VR en la educación

El uso de la realidad virtual en la educación, es algo que está empezando a introducirse en la sociedad. Durante años ha estado limitada a la formación de pilotos aéreos en carísimos simuladores pero gracias a la mejora en la tecnología y la reducción en costes, podemos ir viendo como poco a poco más universidades e institutos introducen la realidad virtual como apoyo para la enseñanza.

Una gran baza de esta tecnología es su accesibilidad, los visores de realidad virtual ofrecen la mejor experiencia pero con unas gafas de cartón como las Cardboard de Google y un móvil podemos ser capaces de experimentar miles de experiencias en realidad virtual. Tampoco podemos obviar su inmersión, lo que aumenta la motivación y aporta un mayor impacto en los procesos de aprendizaje haciendo al alumno partícipe activo de la experiencia y por tanto aumentando su capacidad de retención de lo enseñado.

Declaraciones como las de Baptiste Grève, creador de la plataforma de experiencias virtuales *Unimersiv*¹, no hace mas que fundamentar lo positivo de esta tecnología, dado que el cerebro humano retiene el 10 % de lo que lee, el 20 % de lo que oye y el 90 % de lo que experimenta.

1.2. Diseño de figuras 3D

El diseño de las figuras en 3D se realiza mediante software de diseño, existen multiples herramientas de creación y diseño 3D tanto de pago como gratuitas.

En este trabajo de fin grado nos vamos a centrar en dos de ellas, Blender ² un software Open Source de creación 3D más clásico y Beetleblocks³ una plataforma Open Source que nos propone un método de diseño mas sencillo y basado en la programación por bloques lo cual resulta perfecto para introducirse en el diseño 3D.

Más adelante en este trabajo se aprenderá su uso de tal forma que consigamos crear objetos con Beetleblocks, estos irán desde figuras geométricas básicas, a formas tan complejas como uno quiera debido a la versatilidad de la plataforma.

¹<https://unimersiv.com/>

²<https://www.blender.org/>

³<http://beetleblocks.com/>

Como este trabajo de fin grado esta focalizado en la creación de talleres didácticos para gente sin experiencia en el diseño 3D, Beetleblocks nos permite mostrar el diseño 3D de una forma amena y menos compleja que si usáramos una herramienta más potente.

1.2.1. Entornos VR con figuras en 3D

Los entornos de realidad virtual se componen de figuras en 3 dimensiones. Es el propio entorno en si el que hace que la experiencia se denomine de realidad virtual. Al igual que para para el diseño 3D existen multiples programas de software que nos permiten la creación de estos entornos.

Tal y como está planteado este trabajo, no hemos buscado una experiencia realista sino una centrada en el aprendizaje de una experiencia básica que nos permita familiarizarnos con su creación. Debido a esto en este trabajo de fin de grado hemos optado por A-Frame una utilidad de creación de experiencias y entornos VR Open Source y gratuito.

Gracias a ella aprenderemos a crear un entorno de realidad virtual básico en el que exportaremos nuestras figuras. La creación del entorno de realidad virtual se llevará a cabo en el capítulo 3.3

1.3. Microbit

”Micro:bit es una pequeña computadora programable, diseñada para hacer que el aprendizaje y la enseñanza sean fáciles y divertidos!”

Esta es la frase que encontramos al entrar en <https://microbit.org/es>, y eso es Micro:bit, un sistema de Hardware embebido, diseñado por la BBC para la enseñanza informática en UK. Su primera aparición fue el 12 de Marzo de 2015, y se empezó a distribuir en Febrero de 2016, consta de un procesador ARM CortexM0, acelerómetro y magnetómetro, conectividad Bluetooth y USB, 25 leds, 2 botones programables y puede ser alimentada por medio de pilas o USB. Las entradas y salidas del dispositivo están formadas por 5 anillos conectores que forman parte de un conector mayor de 23 pines.

Gracias a la gran cantidad de sensores que incorpora, sólo con la tarjeta se pueden llevar a cabo centenares de proyectos. BBC Micro: bit también es una plataforma IoT (Internet of Things), lo que la hace muy interesante para usuarios avanzados.



Figura 1.1: Microbit

Y es Open Source, por supuesto. Tanto el hardware como el software de micro:bit es de código abierto.

1.3.1. Microbit como interfaz de usuario

Como ya hemos visto Microbit posee múltiples aplicaciones. Su función en este trabajo de fin grado va dirigida a la interfaz de usuario, es decir, será el controlador de la experiencia de realidad virtual que se conseguirá crear al finalizar la serie de talleres didácticos.

El Microbit nos permitirá un movimiento dentro de la experiencia gracias a su acelerómetro y un movimiento que será tanto físico como de la cámara en nuestro entorno.

Para hacer esto se ha realizado un programa en micropython una variación de python adaptada al Microbit que nos permite ejecutar programas de python en nuestro dispositivo, de tal forma que mapeamos las teclas del teclado a las direcciones en las que movemos nuestro acelerómetro, esto es solo un ejemplo de la versatilidad del Microbit.

1.4. Estructura de la memoria

Tras esta introducción en la que hemos visto una visión de la realidad virtual, en la que nos hemos centrado en las experiencias VR y su continua introducción en la educación Seguidamente podemos ver el diseño de objetos en 3D mediante bloques ya que es el método que se usará en este trabajo de fin de grado. Introducimos Beetleblocks y Blender. Por último conocemos el Microbit y sus múltiples posibilidades, en este caso nos centramos en darle un uso como interfaz de usuario para complementar nuestra experiencia de realidad virtual.

Seguidamente se describen los objetivos que se presentan en este trabajo de fin de grado, en el Capítulo 2, en el que también hablamos de la motivación que se ha tenido para la elección de este proyecto, como de la metodología y el plan de trabajo que se ha seguido.

A continuación en el Capítulo 3 aprenderemos a usar las tecnologías que se nos presentan. Para el diseño 3D

- Beetleblocks y Blender, para la el diseño de objetos en 3D.
- A-Frame, se utilizará para la creación de nuestra experiencia de VR.
- Microbit, aprenderemos a programarlo y usarlo como interfaz de usuario.

En el capítulo 4 nos encontramos la serie de 3 talleres o prácticas de las que consta el núcleo práctico de este trabajo. EL contenido y estructura de cada uno es el siguiente

- El primero centrado en el diseño de figuras en 3D.
- El segundo para aprender a crear nuestro entorno de realidad virtual.
- Por último un tercero para usar nuestro Microbit como controlador en el entorno y poder movernos con él.

Para finalizar tenemos el capítulo 5 con las Conclusiones, donde se analizan los resultados obtenidos de las prácticas y las posibles mejoras futuras que se podían aplicar.

Capítulo 2

Objetivos

2.1. Objetivos

Ya hemos visto una breve introducción de las experiencias de realidad virtual, y del diseño de objetos en 3 dimensiones, ahora toca centrarse en los objetivos de este trabajo de fin de grado.

2.1.1. Implementar entornos VR en la educación

El primer objetivo a la hora de realizar este trabajo de fin de grado, es exponer la utilidad de la realidad virtual en la educación, y como gracias a ella podemos llegar a aprender a programar figuras sencillas en 3D y experimentar con un entorno de realidad virtual de una forma sencilla e interactiva.

2.2. Motivación

La principal motivación que ha llevado a realizar este trabajo de fin de grado, es la propia tecnología en sí, siempre me ha llamado la atención , tanto el diseño de objetos en 3D como las experiencias en realidad virtual.

El gran potencial que tienen, y los múltiples usos que se le pueden dar me han parecido muy interesantes desde un primer momento, gracias a la realización de este trabajo podía a aprender acerca de todo esto y además darle una utilidad en forma de talleres para que alguien aprenda de una forma sencilla y amena.

2.3. Metodología y Plan de Trabajo

En esta sección se expone la metodología y el plan de trabajo que se han llevado a cabo para la realización de este trabajo de fin de grado.

2.3.1. Metodología

La metodología que se ha seguido para la realización de este trabajo de fin grado, esta orientada en la construcción de una serie de talleres didácticos.

Un taller es un programa educacional corto e intensivo, para una cantidad relativamente pequeña de personas, en un área de conocimientos determinada que hace énfasis en la participación para la resolución de problemas.

Los pasos que se han seguido para la creación de los talleres son los siguientes:

- Definir los objetivos del taller.
- Adecuarlos a los participantes, en este caso se trata de alumnos de enseñanzas medias por lo que deben ser acordes a sus conocimientos.
- Definir el formato del taller, será un taller en el que los alumnos harán experiencias guiadas pero en las que participaran activamente en ciertos momentos para involucrarlos con el taller.

Cada taller a su vez seguirá una metodolgia común que es la siguiente.

- Presentación del tema general del taller.
- Planteamiento de los objetivos del taller.
- Presentación del software que se usará.
- Fomentar la participación activa
- Una vez finalizado resumir la sesión y pedir feedback al grupo.

2.3.2. Plan de trabajo

En base a la realización de los talleres se debía tener un conocimiento profundo de todo el software que se usa en ellos, debido a esto se ha realizado un plan de trabajo en el que primeramente se estudiaron las 2 tecnologías Beetleblocks y A-Frame. Gracias a la ayuda del tutor mediante tutorías a través de Hanghouts y mediante la investigación y realización de tutoriales se obtuvo una base sólida acerca de ellas. Posteriormente se contempló el uso de Tensorflow para añadir un tipo de interfaz de usuario a la experiencia de realidad virtual, este primer planteamiento fue desechado en detrimento de la introducción del Micro: bit, la popularidad y variedad de aplicaciones de este último fue determinante, y por último se desarrolló una interfaz de usuario en la que el controlador era el Micro: bit.

Una vez cimentado un conocimiento acerca de Beetleblocks, A-Frame y Micro:bit se pasó a plantear los talleres y se dividieron en una serie de 3 talleres autoconclusivos pero a su vez recursivos entre ellos. Con esto conseguimos que realizando uno de ellos se obtuvieran conocimientos acerca de la tecnología en cuestión y a su vez alguien que realizara los tres obtuviera una experiencia completa de realidad virtual.

Capítulo 3

Creación de experiencias VR

3.1. Tecnologías

En este capítulo nos centraremos en conocer el software y el hardware que vamos a usar en este trabajo, además aprenderemos a usarlo, y veremos unos pequeños ejemplos.

3.2. Diseño 3D

Para diseñar nuestros objetos en 3D, nos ayudaremos de Beetleblocks una potente herramienta online que usa la programación mediante bloques para realizar nuestros objetos de una manera más gráfica y amena.

Una vez tengamos nuestra figura deseada en Beetleblocks, usaremos la segunda de las herramientas para el diseño en 3D, se trata de Blender un software open source. Blender posee numerosas utilidades, pero en este caso nos centraremos solamente en valernos de él, para lograr la extensión deseada de nuestro objeto 3D.

3.2.1. Programación mediante bloques

La programación mediante bloques, facilita mucho las cosas si no tenemos experiencia de ningún tipo con la programación, en este trabajo de fin de grado la veremos tanto en Beetleblocks como cuando usemos el Micro: bit.

3.2.2. Beetleblocks

Beetleblocks es un proyecto creado por Eric Rosenbaum, Duks Koschitz, y Bernat Romagosa, además de la ayuda en la programación de Jens Mönig.

Beetle Blocks esta basado en Scratch e implementado usando Snap! y ThreeJS.











A la hora de comenzar a usar Beetle Blocks, estos son los pasos a seguir:

- Creamos una cuenta.
- Categorías de los bloques.
 1. Motion: Donde se engloban los bloques para movernos por la malla
 2. Shapes: Figuras y formas predeterminadas
 3. Sensing: funciones para usar con nuestros bloques
 4. Variables: creación y uso de variables de entorno
 5. Control: operadores para nuestros conjuntos de bloques, eventos , funciones etc..
 6. Colors: Bloques para dar color a nuestros diseños
 7. Operators: operadores matemáticos para usar en nuestros bloques
 8. Myblocks: bloques propios creados en el proyecto.
- En el capítulo ?? (ojo, otra referencia automática) se muestran los objetivos del proyecto.



Guía de uso Beetleblocks

Para empezar a usar beetleblocks vamos a empezar a crear una serie de figuras básicas. Para aprender a usar beetleblocks expongo a continuación una serie de ejemplos prácticos, en los que enseño como crear figuras básicas, crear círculos, moverse en los 3 ejes espaciales y por último un objeto que usa todo lo aprendido para la creación de unos auriculares.

■ Figuras

1. Dibujar línea: Usamos la orden  que se encuentra en la pestaña de Shapes
2. Creación de variable: En la pestaña Variables tenemos una primera opción make a variable que nos permite crear una variable global para utilizar.
3. Creamos una variable x a la que podemos asignar un valor gracias al bloque  en el que introducimos el valor que queremos que tenga
4. Ahora usamos un bucle para repetir el proceso de la creación de las líneas de una figura geométrica, los bucles vienen indicados en beetleblocks por el bloque  en el hueco del bloque insertaremos nuestra variable, esto provoca que todos los bloques que insertemos a posteriori se repiten el número de veces que valor tenga nuestra variable.
5. Dentro del bucle insertaremos 2 bloques
 - El bloque  el cual dibujará una línea que tendrá la longitud igual al valor que insertemos en su hueco. En este hueco insertaremos nuestra variable arrastrándola, de tal forma que quedara así 
 - El bloque  nos permite girar la dirección de nuestro puntero en este caso lo queremos girar tantos grados como lados tenga nuestra figura geométrica esto lo haremos de una forma muy sencilla con el bloque  Con el que dividimos por el valor de nuestra variable y los combinamos 
 - Por lo tanto, nuestro bucle completo quedaría así.



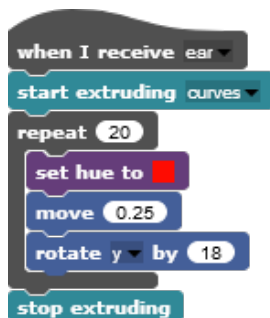
6. Dos bloques más que nos serán muy útiles para crear cualquier figura son  que nos inicia la serie de bloques que anidemos cuando hagamos click. Y  que nos deja el plano en blanco.

7. Finalmente, nuestro bloque para crear figuras geométricas quedaría así.




■ Círculos

Para crear círculos de una manera muy sencilla simplemente empezando a dibujar curvas o líneas nos moveremos hasta completar los 360 grados del círculo girando x grados uno de los ejes. En este ejemplo dibujaríamos un círculo rojo moviéndonos 0.25 hacia delante cada vez que giramos 18 grados, esto lo haríamos 20 veces para completar los 360 grados.

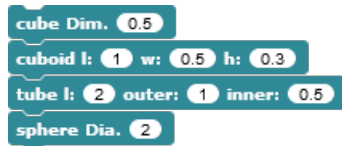


■ Moverse por los ejes

Dependiendo de como dibujemos nuestras figuras deberemos movernos por los ejes y orientar la mariquita que sirve como puntero de orientación para dibujar. El bloque que se usa para movernos y situarnos en donde queremos es este. 

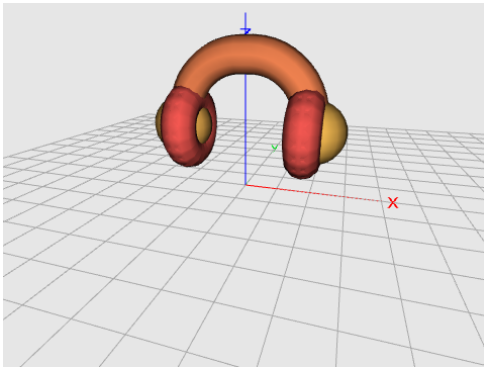
- Rellenar
- Figuras predeterminadas

Las figuras predeterminadas que nos ofrece bettleblocks son el cubo (en dos formas), el cilindro y la esfera cuyos bloques se corresponden con:



- Auriculares

Para probar el manejo de Bettleblocks he creado una figura sencilla , se trata de unos cascos en los que uso una serie de bloques parra dibujarlos.



3.2.3. Blender

Blender es una potente herramienta open source de creación 3D. En este trabajo de fin de grado le vamos a dar un uso muy concreto, se trata de pasar nuestra figura creada con Beetle-blocks a un formato compatible con A-Frame (.obj)

Abrimos Blender lo primero que vemos será una pantalla como esta

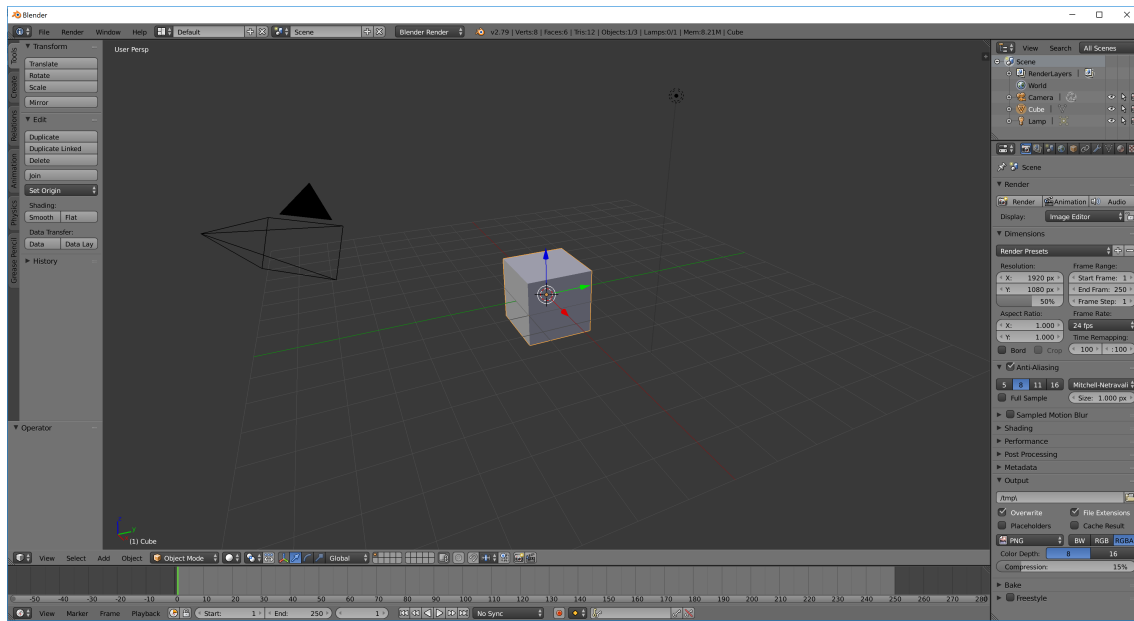


Figura 3.1: Blender

- Los pasos a seguir para exportar nuestra figura en **.obj** y **.mtl** son los siguientes:
 1. Nos posicionamos encima del cubo y presionamos **x** para borrarlo.
 2. pulsamos espacio y escribimos **import stl**.
 3. Pulsamos espacio de nuevo y escribimos **export obj** seleccionando las checkboxes **Write Normals**, **Write Materials**

3.3. A-frame

Como ya sabemos A-Frame es el framework que nos permite crear nuestra experiencia de realidad virtual, su aplicación es posible mediante HTML o JavaScript aunque en la serie de talleres, el uso que le daremos será mediante documentos HTML, en este capítulo aprenderemos a usarlo en las dos vertientes.

A continuación expongo una breve guía de uso de ambas

a-Frame is a web framework for building virtual reality (VR) experiences. A-Frame is based on top of HTML, making it simple to get started. But A-Frame is not just a 3D scene graph or a markup language; the core is a powerful entity-component framework that provides a declarative, extensible, and composable structure to three.js.

Originally conceived within Mozilla and now maintained by the co-creators of A-Frame within Supermedium, A-Frame was developed to be an easy yet powerful way to develop VR content. As an independent open source project, A-Frame has grown to be one of the largest VR communities.

A-Frame supports most VR headsets such as Vive, Rift, Windows Mixed Reality, Daydream, GearVR, Cardboard, Oculus Go, and can even be used for augmented reality. Although A-Frame supports the whole spectrum, A-Frame aims to define fully immersive interactive VR experiences that go beyond basic 360° content, making full use of positional tracking and controllers.

3.3.1. Programación A-frame mediante HTML

REKLLLENAR

3.3.2. Programación A-frame mediante JS

Gracias a la importación de archivos en JavaScript podemos añadir múltiples funcionalidades a A-Frame.

Un paquete muy utilizado es el `aframe-extras`¹ se trata de una serie de Add-ons y helpers para A-Frame, e incluye componentes de control, modelos predefinidos, primitivos ...

¹<https://github.com/donmccurdy/aframe-extras>.

Otro recurso muy utilizado para A-Frame y basado en JavaScript son los components², estos nos habilitan una serie de funciones y funcionas como módulos, existe una amplia biblioteca de components, entre ellos hay algunos que se organizan en paquetes como superframe.³

El uso de cada component es particular y puede encontrarse en su documentación, pero generalmente el uso conlleva a vincularlo con una entidad `<a-entity>` y configurarlo.

Tomamos como ejemplo particle system, su uso sería de la siguiente forma:

- Importamos el component de la misma manera que ya lo hicimos con A-Frame en nuestro index.html

```
<head>
  <script src="https://aframe.io/[...].min.js">
    </script>
  <script src="https://unpkg.com/aframe-[...]-component.min.js">
    </script>
</head>
```

- Y a continuación vinculamos particle system con una entidad

```
<body>
  <a-scene>
    <a-entity particle-system="preset: snow" position="0 0 -10">
      </a-entity>
    </a-scene>
  </body>
```

3.4. Microbit

La última tecnología en introducir en este trabajo de fin de grado es el Microbit, ya vimos en el capítulo 1.3 su definición y características, su función no es otra que servir de controlador de nuestra experiencia.

²<https://aframe.io/docs/0.9.0/introduction/entity-component-system.html>.

³<https://github.com/supermedium/superframe>.

Para lograr convertirlo en una interfaz de usuario, hay dos herramientas fundamentales pyautogui y bitio.

Gracias a la ayuda del tutor, se partió de una base como es esta pequeña práctica <https://github.com/sarehp/miframe>, en ella podemos ver la aplicación de estas dos herramientas y fue idónea para comenzar la investigación.

El siguiente paso fue llevar a cabo una serie de test y pruebas con estas dos herramientas. A continuación se entra más en detalle en estas dos herramientas.

3.4.1. Pyautogui

Para aprender a utilizar pyautogui se han utilizado estas documentaciones

<https://buildmedia.readthedocs.org/media/pdf/pyautogui/latest/pyautogui.pdf> <https://pyautogui.readthedocs.org/>

Como se puede leer en ellas, pyautogui es un módulo de python que nos permite mapear el teclado y el ratón, es decir controlar mediante la programación todas las teclas de ambos dispositivos.

Gracias a pyautogui, se ha podido trasladar las pulsaciones y gestos del microbit a teclas de nuestro teclado o movimientos de nuestro ratón.

El código fuente se encuentra en <https://github.com/asweigart/pyautogui>

3.4.2. Bitio

Bitio es la segunda herramienta que se ha usado para conseguir convertir nuestro Microbit en un controlador, se trata de una librería de Microbit para Python, nos permite correr código de python en nuestro PC y poder interactuar directamente con nuestro Microbit.

Para aprender su uso se siguió la documentación, la cual se puede ver en el siguiente enlace: <https://github.com/whaleygeek/bitio>

En ella encontramos desde como instalarlo en nuestro programa de python en la sección *Getting Started*, como hacer la conexión con nuestro Microbit y una serie de ejemplos de las funciones propias de bitio como puede ser leer los valores del acelerómetro o detectar cuando pulsamos uno de los botones.

3.4.3. Micropython

Micropython es el lenguaje de programación propio del Microbit. En este trabajo de fin de grado es el utilizado a la hora de realizar los programas que corran las acciones de nuestra placa.

Basado en python se usa de una forma prácticamente idéntica. Cualquier editor de texto es válido para programar los scripts, en este caso se ha optado por la recomendación oficial que es el editor Mu.

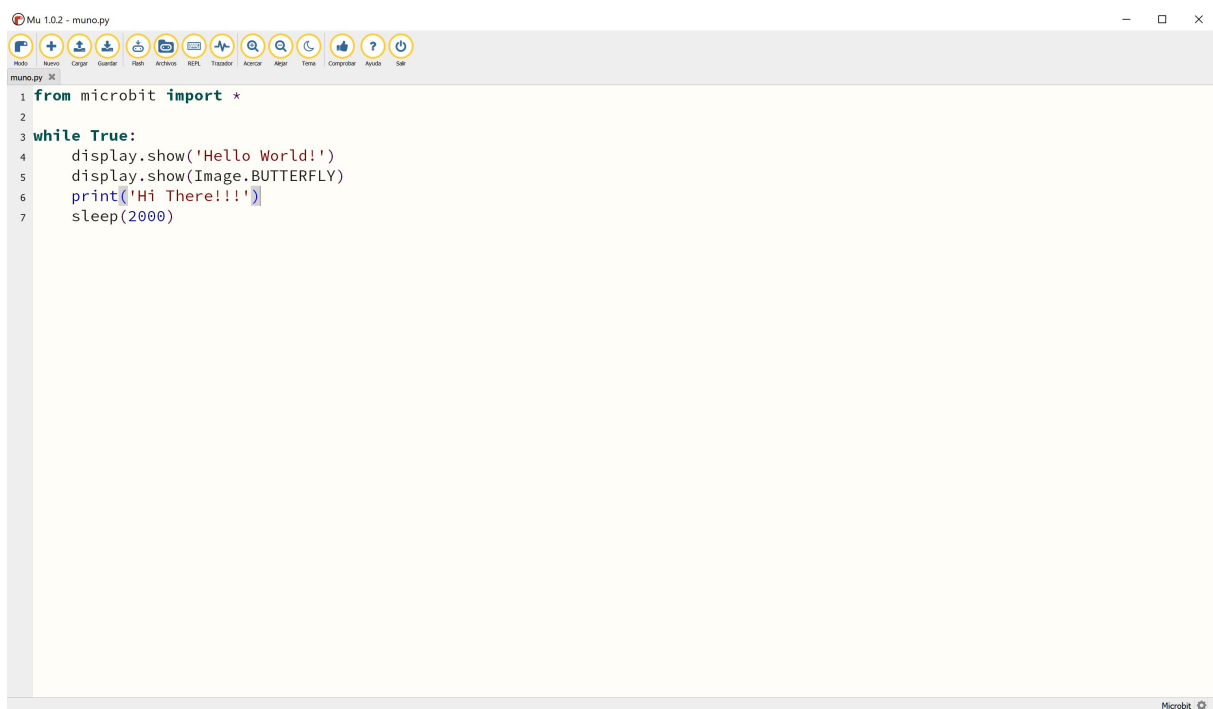


Figura 3.2: Editor Mu

Nos permite de una forma intuitiva cargar el código en el dispositivo con un botón dedicado para ello y ofrece entre sus distintos modos la posibilidad de visualizar la salida estándar.

3.4.4. Programación Microbit mediante bloques(Makecode)

Makecode es un editor para programar el Microbit que usa el método de bloques.

Con una interfaz muy parecida a la de Beetleblocks divide sus bloques en grupos, que aportan distintas funciones como Bucles , acciones para los LEDs, Lógica, Matemática . . . etc

En la web de Makecode⁴ disponemos de numerosos tutoriales para familiarizarnos con su uso. Aunque en este trabajo Makecode no ha sido utilizado para la realización de talleres se profundizó en cada uno de los tutoriales ya que ofrecen una mejor visión global de las posibilidades del Microbit, pudiendo programarse sencillos juegos de una forma rápida y amena.

En la siguiente figura 3.3 podemos ver su aspecto inicial con una distribución común de la programación por bloques con los diferentes grupos de funciones y variables a la izquierda (2) y una área de trabajo grande para arrastras los bloques (3). Como complemento se puede hacer una previsualización del resultado de la ejecución de nuestros bloques (1).

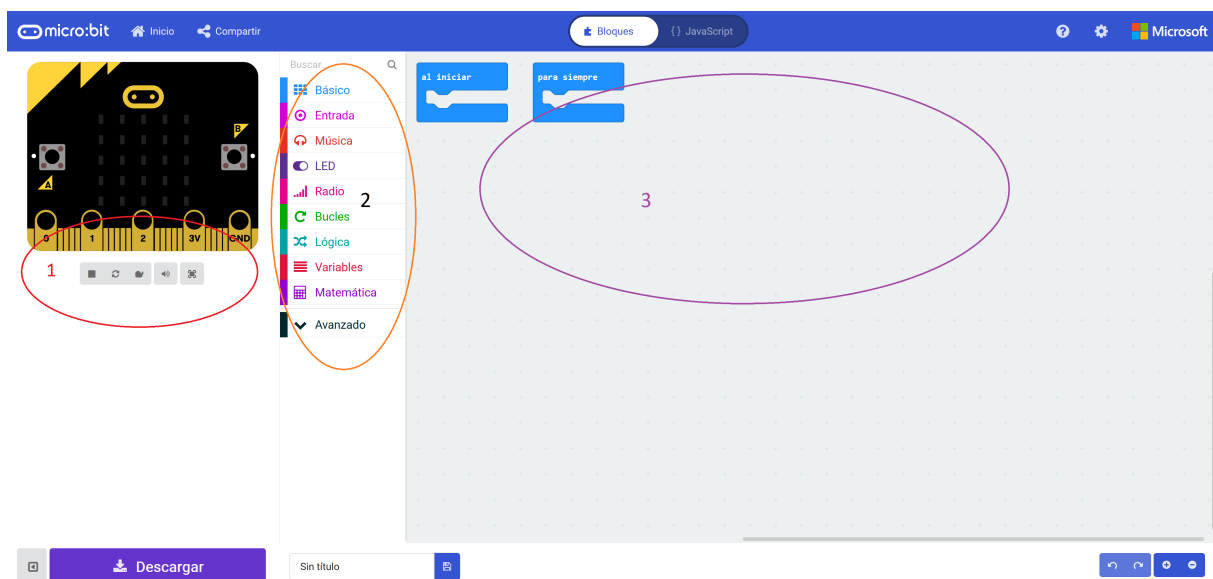


Figura 3.3: Makecode

⁴<https://makecode.microbit.org/>

Capítulo 4

Diseño e implementación de los talleres

4.1. Introducción y propósito de los talleres

El principal motivo de este trabajo de fin de grado es introducir todas estas herramientas que hemos explicado previamente a alumnos de enseñanzas medias, y para ello voy a desarrollar una serie de tres talleres, estos son autoconclusivos pero a su vez cada uno de ellos utiliza lo que aprendimos en el anterior haciendo que una vez finalizamos los tres tengamos una experiencia de realidad virtual completa creada con A-Frame en la que veamos un objeto creado con Beetleblocks y que puede ser controlada con nuestro Micro:bit para movernos por ella.

Por tanto el desarrollo y contenido de los talleres es el siguiente:

- **Taller 1:** Crea tu objeto 3D con Beetleblocks.

En este taller nos centraremos en la creación de un objeto 3D con Beetleblocks, aprenderemos a exportarlo y a darle el formato adecuado para el siguiente taller.

- **Taller 2:** Crea tu experiencia de realidad virtual con A-frame.

En este taller aprenderemos a usar A-Frame creando un entorno de realidad virtual en el que esta incluido nuestro objeto creado anteriormente con Beetleblocks.

- **Taller 3:** Micro:bit como interfaz de usuario para tu experiencia de realidad virtual.

En este taller convertiremos nuestro Micro:bit en un controlador para nuestra experiencia de realidad virtual

4.2. Taller 1 (Creación y exportación de un objeto 3D)

Este primer taller tiene el objetivo de que aprendas a crear un objeto en 3D con Beetleblocks y su accesible método de programación mediante bloques. Además en la segunda parte del taller usaremos Blender para dejar nuestro objeto preparado para usarlo en un entorno VR creado con A-Frame.

El objetivo de este taller es introducir el diseño en 3D a alumnos de enseñanzas medias de tal forma que con una sesión, sean capaces de crear algún objeto básico en 3D y darle las herramientas necesarias para que puedan desarrollar algo más complejo en un futuro

4.2.1. Beetleblocks

Beetleblocks nos pone las cosas fáciles a la hora de crear objetos 3D, con este taller se pretende que un alumno con conocimientos muy básicos acerca de la programación consiga crear una figura en 3D, con unos pocos pasos.

Antes de comenzar a programar nuestra primera figura debemos crear una cuenta en Beetleblocks para tener nuestros proyectos guardados.

- Vamos a la web de Beetleblocks¹ y en la esquina superior derecha vemos la opción para crear/iniciar sesión.
- Una vez hecho esto podemos comenzar a usar Beetleblocks, pulsando en *Run Beetleblocks*.

¹<http://beetleblocks.com>

El Primer Objeto

Abrimos Beetleblocks y tendremos una pantalla como la de la Figura 4.1

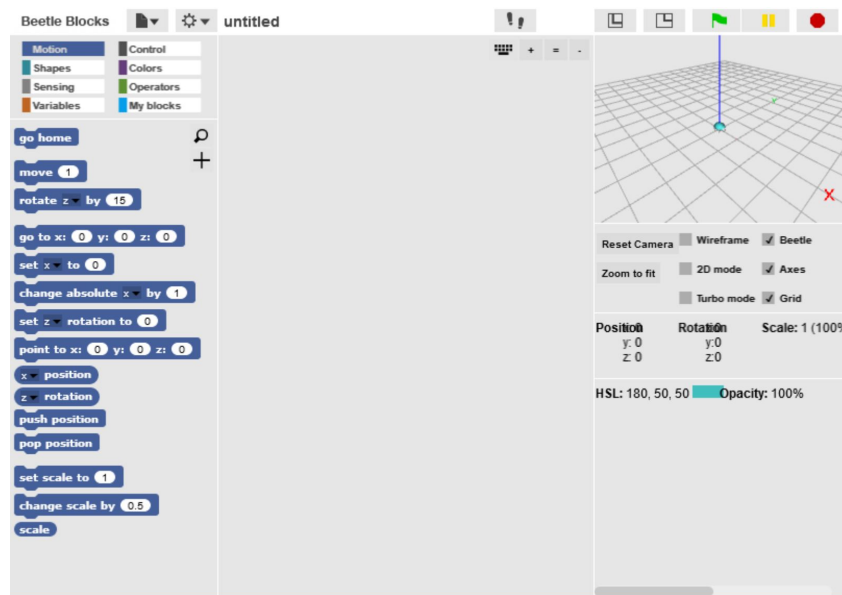


Figura 4.1: Beetleblocks

A la izquierda tenemos el menú en el que encontramos los diferentes bloques ordenados por categorías. En el centro tenemos el área de trabajo, y a la derecha una pre visualización en 3D de nuestra área de trabajo.

En esta última encontramos una mariquita la cual es muy importante porque nos indica nuestra orientación respecto a los 3 ejes, siendo su cabeza la dirección hacia la que estamos apuntando actualmente, en la Figura 4.2 estaríamos mirando hacia la parte positiva del eje x .

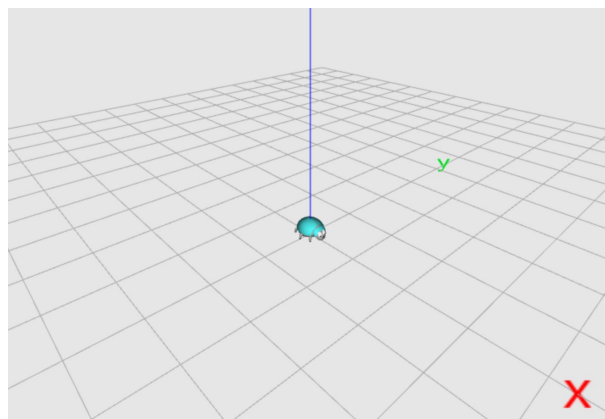












Figura 4.2: Orientación en Beetleblocks

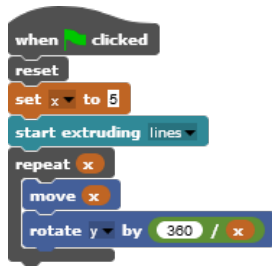
Sabiendo esto ya podemos empezar a crear programar nuestro primer objeto.

1. Dibujar línea: Usamos la orden  que se encuentra en la pestaña de Shapes
2. Creación de variable: En la pestaña Variables tenemos una primera opción make a variable que nos permite crear una variable global para utilizar.
3. Creamos una variable x a la que podemos asignar un valor gracias al bloque  en el que introducimos el valor que queremos que tenga
4. Ahora usamos un bucle para repetir el proceso de la creación de las líneas de una figura geométrica, los bucles vienen indicados en bettleblocks por el bloque  en el hueco del bloque insertaremos nuestra variable, esto provoca que todos los bloques que insertemos a posteriori se repiten el número de veces que valor tenga nuestra variable.
5. Dentro del bucle insertaremos 2 bloques
 - El bloque  el cual dibujará una línea que tendrá la longitud igual al valor que insertemos en su hueco. En este hueco insertaremos nuestra variable arrastrándola, de tal forma que quedara así 
 - El bloque  nos permite girar la dirección de nuestro puntero en este caso lo queremos girar tantos grados como lados tenga nuestra figura geométrica esto lo haremos de una forma muy sencilla con el bloque  Con el que dividimos por el valor de nuestra variable y los combinamos 
 - Por lo tanto, nuestro bucle completo quedaría así.



6. Dos bloques más que nos serán muy útiles para crear cualquier figura son  que nos inicia la serie de bloques que anidemos cuando hagamos click.
Y  que nos deja el plano en blanco.

7. Finalmente, nuestro bloque para crear figuras geométricas quedaría así.



Formato

Por último necesitamos exportar nuestra figura, en formato *.stl*.

Beetleblocks no nos permite la exportación en un formato compatible con A-Frame, en la siguiente sección veremos como conseguirlo gracias a Blender.

4.2.2. Blender

Ya hemos aprendido como crear nuestro objeto, ahora solo nos falta que tenga una extensión *.obj* para ello vamos a usar un software OpenSource de diseño 3D como es Blender.

Los pasos a realizar son muy sencillos, solo usaremos Blender como si de una herramienta de conversión se tratase.

Abrimos Blender lo primero que vemos será una pantalla como esta

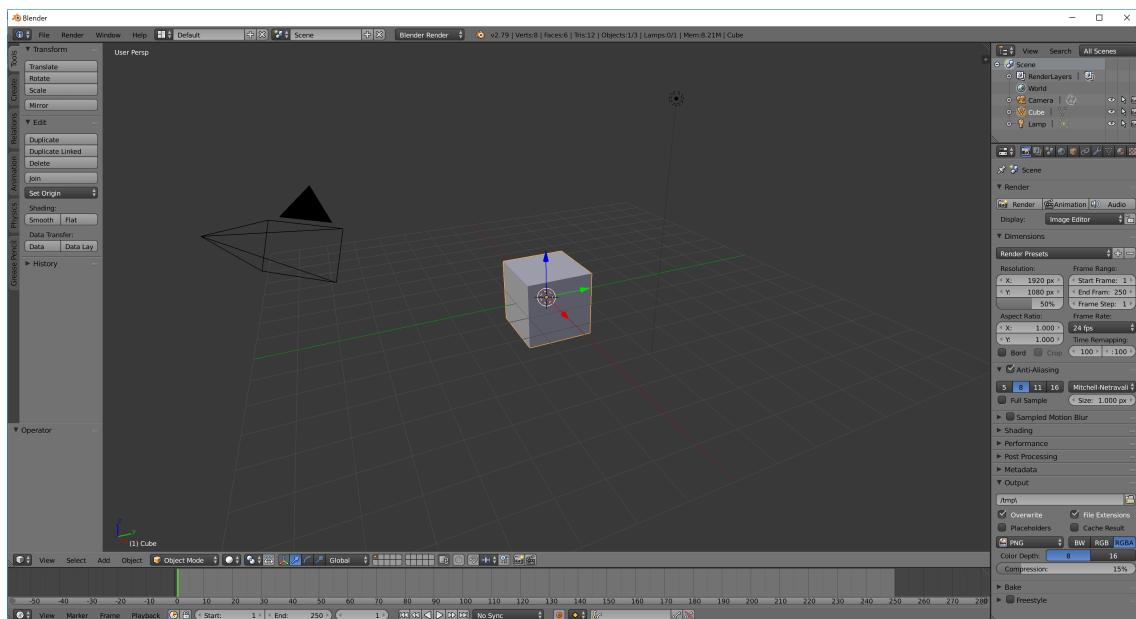


Figura 4.3: Blender

- Los pasos a seguir para exportar nuestra figura en **.obj** y **.mtl** son los siguientes:
 1. Nos posicionamos encima del cubo y presionamos **x** para borrarlo.
 2. Pulsamos *espacio*, escribimos **import stl** y seleccionamos nuestro objeto importado de Bettleblocks.
 3. Con nuestro objeto cargado en Blender pulsamos *espacio* y escribimos **export obj** seleccionando las checkboxes **Write Normals**, **Write Materials** y guardamos lo exportamos a la ruta deseada.



Veremos que han aparecido dos nuevos archivos **xxx.obj** y **xxx.mtl**.

Con esto conseguimos el formato apropiado de nuestro objeto para el siguiente taller, en el que aprenderemos a crear una experiencia de realidad virtual con A-frame.

4.3. Taller 2 (Creación de un entorno VR)

En este segundo taller aprenderemos a crear una experiencia de realidad virtual con A-Frame, además importaremos nuestro objeto creado en el primer taller con Beetleblocks, y podremos visualizarla en nuestro smartphone.

4.3.1. Creación de entorno VR con A-Frame

Para crear nuestra experiencia de realidad virtual el primer paso es crear nuestro `index.html` en el directorio de nuestro proyecto, para esto crearemos un repositorio de la siguiente manera: En un terminal unix escribimos

```
mkdir A-Frame
cd A-Frame
makefile index.html
```

A-Frame puede ser utilizado desde un documento plano de HTML sin la necesidad de instalar nada.

Para empezar a usar A-Frame en nuestro `index.html` debemos importarlo en la cabecera de esta manera:

```
<head>
  <script src="https://aframe.io/releases/0.9.0/aframe.min.js"></script>
```

Con esto ya hemos cargado la versión de A-Frame y podemos dar paso a nuestra primera prueba.

■ Primera figura

Un ejemplo sencillo para aprender a usar A-Frame consiste en cargar un primitive, plantillas de elementos que están dentro del código de A-Frame, gracias a ellos podemos crear un cubo , un cilindro, una esfera . . . de una forma muy sencilla. A la hora de usarlos en nuestro *index.html* basta con nombrarlos como si se tratara de una entidad, a los primitives se le pueden dar diferentes propiedades. Siempre que cargamos una figura en A-Frame tiene que ser dentro de la escena, en nuestro documento html `<a-scene>`.

Cargamos un cubo y una esfera en nuestra escena de la siguiente manera:

```

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

```

Además podemos crear nuestros propios primitivos, algo que veremos en la siguiente sección de este capítulo, exportando un archivo JavaScript.

Ahora que ya sabemos la metodología para cargar objetos en A-Frame, el siguiente paso a realizar es cargar nuestro objeto creado con Beetleblocks, ya con el formato *.obj* que le hemos conseguido dar con Blender.

- El proceso para cargarlo en nuestro index.html es el siguiente:

1. Inicializamos nuestras variables *.obj*, *.mtl*, y *textura*.

```

<a-entity
  obj-model="obj: #crate-obj;" mtl="#crate-mtl"
  material="src: #crate-texture">

</a-entity>.

```

2. Cargamos el *.obj*, *.mtl*, y *textura*² que queramos³.

```

<a-assets>
  <a-asset-item id="crate-obj" src="models/cascos3.obj"></a-asset-item>
  <a-asset-item id="crate-mtl" src="models/cascos3.mtl"></a-asset-item>
  
</a-assets>

```

²Para la textura nos vale cualquier *.jpg*.

³En la propiedad *src* indicamos la ruta del archivo.

4.3.2. Animación básica con A-frame.

A-Frame nos permite añadir algo de animación básica a nuestra figura, gracias a la importación de paquetes en JavaScript.

Nuestro objetivo va a ser que nuestra figura rote, para hacerlo vamos a valernos del paquete `aframe-animation-component`.

Lo importamos en nuestra cabecera `<head>`

```
<script src="https://rawgit.com/ngokevin/aframe-animation-component
  /master/dist/aframe-animation-component.min.js">

</script>
```

Para usar la animación usamos `<a-animation>` dentro de `<a-entity>`

```
<a-entity [...]>

  <a-animation attribute="rotation" begin="click"
    repeat="5" to="0 360 0"> </a-animation>

  <a-event name="mouseenter" scale="4 1 6">
</a-event>

</a-entity>
```

Con esto conseguimos que el objeto rote cuando hacemos click con el ratón en él.

Al estar en un entorno de realidad virtual, no disponemos de nuestro ratón y lo que funciona como tal es el centro de nuestra mirada, para conseguir apuntar al objeto creamos un cursor en el centro de la pantalla que no es otra cosa que el centro de la perspectiva que tenemos en cada momento. El cursor se crea una vez definimos la posición inicial de la cámara, en este caso le daremos un color rojo que resalte con el fondo.

[...]

```
<a-camera position="1.5 1 6.5">

<a-cursor color="#FF0000">

</a-camera>
</a-entity>
[...]
```

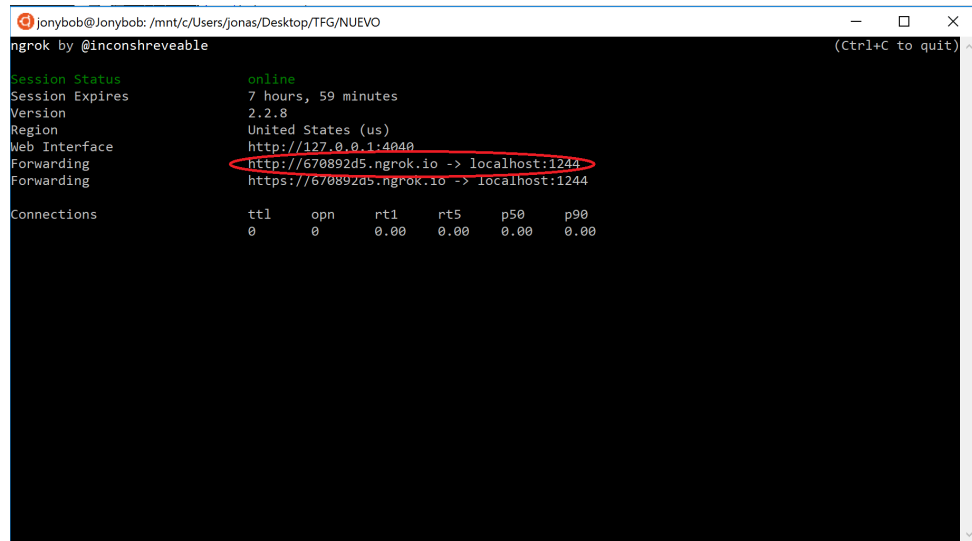
4.3.3. Anexo: Lleva tu experiencia VR a tu smartphone

Ya tenemos nuestra figura cargada en A-Frame de forma local. Pero como ya sabemos para visualizar el entorno de realidad virtual, necesitamos algo más y es un visor, la forma más rápida y sencilla de ver nuestra experiencia, es a través de nuestro smartphone, para exportar nuestro index.html a internet nos vamos a ayudar de una utilidad llamada ngrok.

- Para usar ngrok hacemos lo siguiente
 1. Abrimos una terminal y no vamos a nuestra carpeta raíz del proyecto, lanzamos nuestro servidor de forma local:

```
python -m SimpleHTTPServer 1234
```
 2. instalamos ngrok, simplemente descargando el .zip y descomprimiéndolo en nuestra carpeta raíz.
 3. después de esto lo ejecutamos pasándole como argumento el puerto de nuestro servidor.

```
./ngrok http 1234
```
 4. Ngrok nos devuelve una pantalla con la Url de nuestra aplicación.

A terminal window titled 'jonybob@Jonybob: /mnt/c/Users/jonas/Desktop/TFG/NUEVO' displays the output of the 'ngrok by @inconshreveable' command. The output shows session status as 'online', session expires in 7 hours, version 2.2.8, region 'United States (us)', and web interface 'http://127.0.0.1:4040'. The forwarding section shows 'http://670892d5.ngrok.io -> localhost:1244' and 'https://670892d5.ngrok.io -> localhost:1244'. The http URL is circled in red. A table of connections is also shown at the bottom.

```
jonybob@Jonybob: /mnt/c/Users/jonas/Desktop/TFG/NUEVO
ngrok by @inconshreveable (Ctrl+C to quit)

Session Status      online
Session Expires     7 hours, 59 minutes
Version             2.2.8
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://670892d5.ngrok.io -> localhost:1244
                   https://670892d5.ngrok.io -> localhost:1244

Connections
  ttl    opn    rt1    rt5    p50    p90
    0      0    0.00   0.00   0.00   0.00
```

5. Introduciendo esta url en un navegador compatible como Google Chrome podemos visualizar nuestra experiencia en nuestro smartphone u otro dispositivo compatible.

4.4. Taller 3(Microbit como IU en entorno VR)

En este tercer taller se mostrará como convertir nuestro Microbit en un controlador para una experiencia de realidad virtual creada con A-Frame.

El propósito de este taller es partir del anterior a la hora de usar el controlador en la experiencia, pero no necesariamente es obligatorio ya que el Microbit debe funcionar en cualquier entorno de A-Frame que permita movimiento independientemente de como sea la experiencia en si.

Las funciones que llevará cabo el Microbit una vez finalizado el taller, serán las siguientes:

- Movimiento físico en la experiencia al pulsar B.
- Movimiento de la cámara al pulsar A.
- Interacción al pulsar los dos botones a la vez del Microbit.

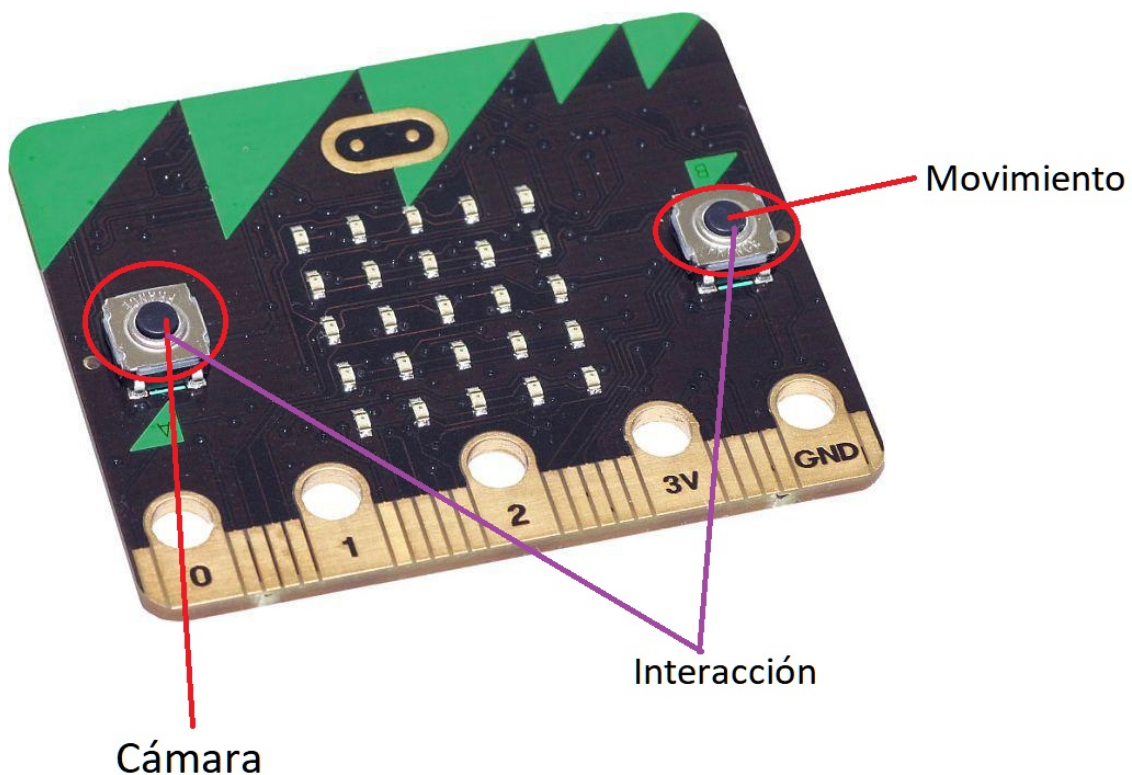


Figura 4.4: Funciones del Microbit

Para programar el Microbit primero se debe preparar el entorno, como ya se indica en la introducción el lenguaje utilizado será python, o en su defecto micropython. El sistema operativo será Linux, Microbit también dispone de compatibilidad con Windows mediante la instalación de un driver, esto será abordado en el anexo del trabajo.

Los pasos a realizar por tanto serán:

- Disponer de un editor de texto.
- Instalar bitio.
- Instalar pyautogui.

Con esto ya se dispone de un entorno apto para empezar a programar nuestra placa.

4.4.1. Instalación de Bitio

El primer paso es instalar Bitio, para ello lo primero que se debe hacer es ir a su repositorio de github.

<https://github.com/whaleygeek/bitio>

Como la propia documentación indica la mejor forma de usar bitio en un entorno educativo es copiar la carpeta 'src/microbit' en el directorio Home del proyecto actual.

Con esto se consigue que automáticamente cualquier programa que esté en el mismo directorio acceda a la carpeta de bitio cuando se importe el paquete microbit, al principio del programa **.py**

import microbit

Dentro de la carpeta 'src' se encuentran una serie de programas para probar las distintas funcionalidades de bitio.

La forma de probar si se ha instalado bitio correctamente es tan simple como hacer una prueba con uno de ellos. Tomando como prueba *button.py* se obtienen estos valores en la terminal al ejecutar el programa.

`python button.py`

```
No micro:bit has previously been detected
Scanning for serial ports
```

```
remove device, then press ENTER
scanning...
port[0]:COM1
found 1 device(s)
plug in device, then press ENTER
scanning...
port[0]:COM1
port[1]:COM6
found 2 device(s)
found 1 new device
selected:COM6
Do you want this device to be remembered? (Y/N)
connecting...
Your micro:bit has been detected
Now running your program
micro:bit connected - press button A to test
Button A pressed
Button A pressed
Button A pressed
```

El funcionamiento de cualquier programa de bitio es el siguiente primero escanea los dispositivos conectados a nuestro pc, sin el Microbit entre ellos, a continuación pide al usuario que lo conecte y establece la conexión con él que corresponde al puerto nuevo encontrado. Una vez que el programa está corriendo la última línea que se encuentra es.

Now running your program

En este caso el programa nos devuelve una línea de texto cada vez que se pulsa el botón A.

4.4.2. Instalación de Pyautogui

A continuación se debe instalar pyautogui para poder correr código de python en el Microbit.

El método de instalación difiere un poco en función del sistema operativo que se utiliza, en este trabajo de fin de grado los test han sido tanto en Linux como en Windows 10.

Al igual que con bitio el método de instalación que se describe aquí esta basado en la propia documentación:

<https://github.com/asweigart/pyautogui>

Para instalar pyautogui en Linux:

```
pip3 install python3-xlib
sudo apt-get install scrot
sudo apt-get install python3-tk
sudo apt-get install python3-dev
pip3 install pyautogui
```

Con esto ya se cuenta con pyautogui totalmente funcional al importar el paquete al principio del programa mediante la línea.

import pyautogui

Una prueba sencilla para saber si pyautogui es funcional es probar el programa testpy.py que se adjunta en este trabajo. Su contenido es el siguiente.

```
import microbit
import pyautogui
while True:
    if microbit.button_a.was_pressed():
        microbit.display.show("PULSA A")
        screenWidth, screenHeight = pyautogui.size()
        pyautogui.moveTo(screenWidth / 2, screenHeight / 2)
        microbit.sleep(500)
        microbit.display.clear()
```

La funcionalidad del programa es hacer centrar la posición de nuestro ratón cuando se pulsa A en el microbit, para saber si pyautogui esta correctamente instalado basta con lanzar este programa y ver si el puntero del ratón se desplaza al centro de la pantalla al pulsar A.

4.4.3. Programación del Microbit

Una vez se tiene el entorno preparado y todas las herramientas instaladas se puede proceder a programar el microbit y convertirlo en el controlador de la experiencia.

Como ya se hace referencia anteriormente en el trabajo, el punto de partida se corresponde con el siguiente repositorio de github.

<https://github.com/sarehp/microbit-aframe>.

El objetivo del taller será optimizar este código base para crear un controlador con más funcionalidades.

A continuación se describe el estudio y realización de cada una de ellas correspondientes a la figura 4.4

A) Movimiento

B) Control de la cámara.

C) Interacción.

A) Movimiento

Para programar el movimiento dentro de A-Frame se tomó el código del repositorio de github y se le añadió la funcionalidad de centrar la cámara antes de comenzar el movimiento para facilitar la experiencia. Esto es programado con pyautogui.

```
screenWidth, screenHeight = pyautogui.size()
pyautogui.mouseDown()
pyautogui.moveTo(screenWidth / 2, screenHeight / 2)
pyautogui.mouseUp()
```

- Con la primera línea se guarda la resolución de la pantalla en dos variables una para el ancho y otra para el alto.
- Se deja el ratón pulsado con la orden `mouseDown`, se hace esto para entrar en el control de la cámara en A-frame.
- Se mueve el ratón al centro de la pantalla
- Por último se suelta el botón izquierdo del ratón para salir del control de la cámara.

A) Control de la Cámara

El primer paso para programar el control de la cámara fue estudiar las diferentes funciones que ofrece pyautogui a la hora de poder interactuar con la pantalla y el ratón.

Para poder implementar el movimiento de la cámara con el acelerómetro del Microbit, primero es necesario entender como se maneja la cámara en A-Frame con un ratón común. Para mover la cámara es necesario pulsar dentro del escenario de A-Frame y a continuación desplazar el ratón en la dirección deseada sin soltar el botón izquierdo del ratón.

Por tanto lo que se ha hecho en este caso es mapear el movimiento del ratón por la pantalla en base a los valores que ofrece el acelerómetro del Microbit. Los diferentes pasos que se han seguido son los siguientes.

En lo relativo a la pantalla se extrae la resolución. Como se ve en la figura 4.5 existen multitud de resoluciones posibles en función de la pantalla en la que se ejecuta la experiencia de A-Frame, en concreto en la realización de este trabajo de fin de grado se disponía de una pantalla 4k con una resolución de 3840x2160 pixeles, esta no es una resolución muy común en la actualidad, por ello aquí se describirá la programación de la funcionalidad en base a una pantalla Full HD (1920x1080)

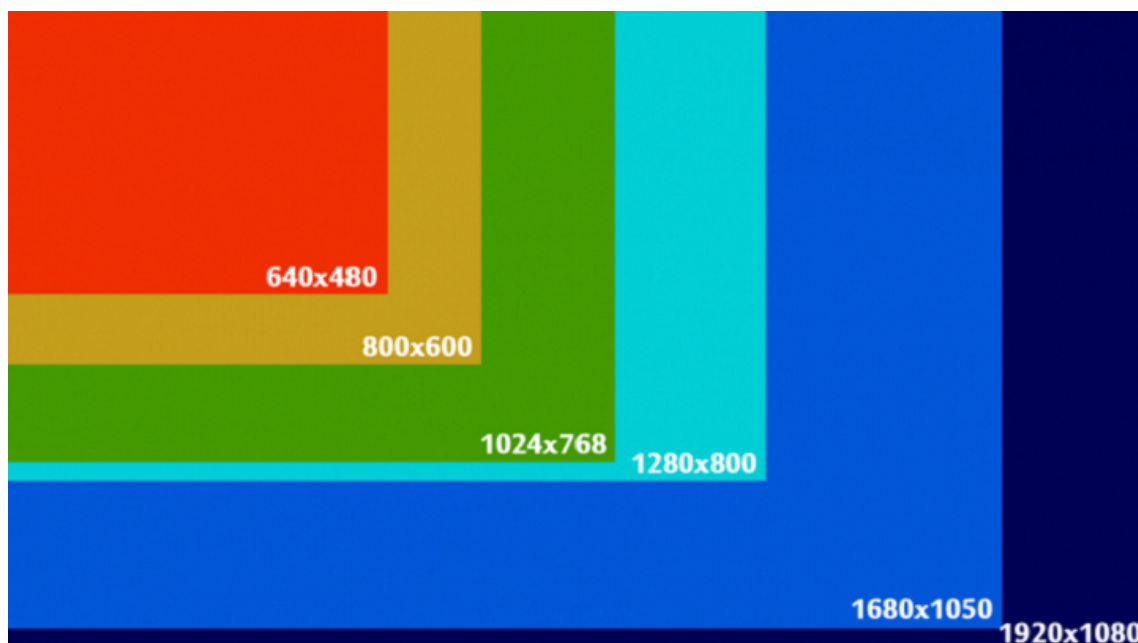


Figura 4.5: Resoluciones

Tomando como base por tanto la resolución Full HD tenemos 1920 píxeles de ancho y 1080

pixeles de alto. Ahora se necesita saber como funciona el acelerómetro del Microbit. Con la función `microbit.accelerometer.get_x()` y `microbit.accelerometer.get_y()` se extraen los valores del acelerómetro los cuales van de 1000 a -1000 en cada uno de los ejes como se ve en la figura 4.6.

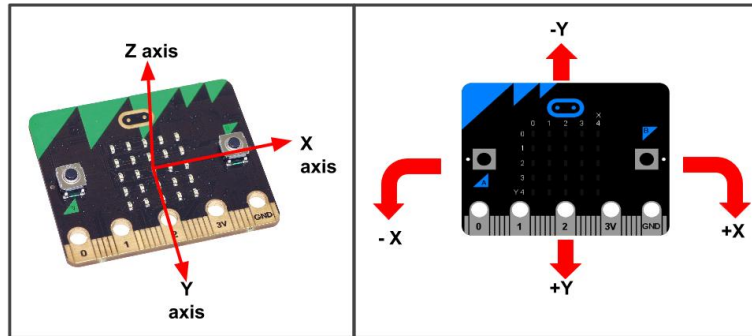


Figura 4.6: Valores del acelerómetro

Como el objetivo es mapear el movimiento del ratón con los valores del acelerómetro solo se necesitan dos ejes, debido a que la pantalla es un plano de dos dimensiones.

A) Interacción

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.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

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. a

2. b

5.4. Trabajos futuros

Ningún 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/TFM.

Apéndice A

Manual de usuario

Bibliografía

- [1] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., 1999.
- [2] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., 1999.

Apéndice B

Introducción

No te olvides de echarle un ojo a la página con los cinco errores de escritura más frecuentes¹.

Aconsejo a todo el mundo que mire y se inspire en memorias pasadas. Las mías están todas almacenadas en mi web del GSyC².

B.1. Sección

Esto es una sección, que es una estructura menor que un capítulo.

Por cierto, a veces me comentáis que no os compila por las tildes. Eso es un problema de codificación. Cambiad de “UTF-8” a “ISO-Latin-1” (o viceversa) y funcionará.

B.1.1. Estilo

Recomiendo leer los consejos prácticos sobre LaTeX de Diomidos Spinellis³.

Sobre el uso de las comas⁴

A continuación, viene una figura, la Figura B.1. Observarás que el texto dentro del a referencia es el identificador de la figura (que se corresponden con el “label” dentro de la misma). También habrás tomado nota de cómo se ponen las “comillas dobles” para que se muestren

¹<http://www.tallerdeescritores.com/errores-de-escritura-frecuentes>

²<https://gsyc.urjc.es/~grex/pfcs/>

³<https://github.com/dspinellis/latex-advice>

⁴<http://narrativabreve.com/2015/02/opiniones-de-un-corrector-de-estilo-11-recetas-par>
html



Figura B.1: Página con enlaces a hilos

correctamente. Volviendo a las referencias, nota que al compilar, la primera vez se crea un diccionario con las referencias, y en la segunda compilación se “rellenan” estas referencias. Por eso hay que compilar dos veces.

B.2. Estructura de la memoria

En el capítulo ?? (ojo, otra referencia automática) se muestran los objetivos del proyecto.

Apéndice C

Estado del arte

Puedes citar libros, como el de Bonabeau et al. sobre procesos estigmérgicos [2].

C.1. Sección 1

Hemos hablado de cómo incluir figuras. Pero no hemos dicho nada de tablas. A mí me gustan las tablas. Mucho. Aquí un ejemplo de tabla, la Tabla C.1.

Si tu proyecto es un software, siempre es bueno poner la arquitectura (que es cómo se estructura tu programa a “vista de pájaro”).

Por ejemplo, puedes verlo en la figura C.1.

Si utilizas una base de datos, no te olvides de incluir también un diagrama de entidad-relación.

1	2	3
4	5	6
7	8	9

Cuadro C.1: Ejemplo de tabla

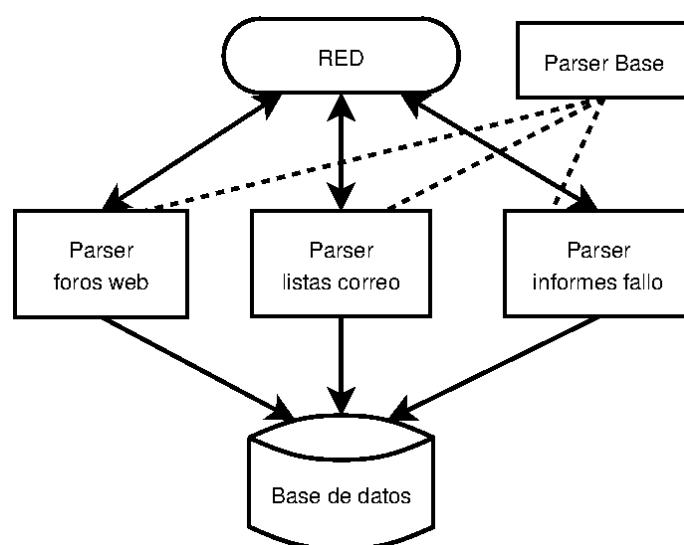


Figura C.1: Estructura del parser básico