



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.
- Micro Bit.

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 en la que primeramente se creará un objeto 3D con programación mediante bloques, a continuación se mostrará como crear un entorno de realidad virtual con A-Frame y por último se explicará como usar el Micro Bit como interfaz de usuario de la experiencia anteriormente creada.

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 didácticos con cada uno de ellos incidiendo en una de ellas.

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 esto se ha conseguido mediante fichas.



# Índice general

<b>1. Introducción</b>	<b>1</b>
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. Micro Bit . . . . .	3
1.3.1. Micro Bit como interfaz de usuario . . . . .	5
1.4. Estructura de la memoria . . . . .	5
<b>2. Objetivos</b>	<b>9</b>
2.1. Objetivos . . . . .	9
2.1.1. Exponer la utilidad de la VR en la educación . . . . .	9
2.1.2. Creación de un objeto 3D y una experiencia VR . . . . .	9
2.1.3. Implementar el Micro:bit dentro de A-Frame . . . . .	10
2.2. Motivación . . . . .	10
2.3. Metodología y Plan de Trabajo . . . . .	11
2.3.1. Metodología . . . . .	11
2.3.2. Plan de trabajo . . . . .	12
<b>3. Creación de experiencias VR</b>	<b>13</b>
3.1. Tecnologías . . . . .	13
3.2. Diseño 3D . . . . .	13
3.2.1. Programación mediante bloques . . . . .	14
3.2.2. BeetleBlocks . . . . .	15

3.2.3.	Categorías de los bloques . . . . .	16
3.2.4.	Prácticas de aprendizaje en BeetleBlocks . . . . .	17
3.2.5.	Blender . . . . .	19
3.3.	A-frame . . . . .	20
3.3.1.	A-frame dentro de HTML . . . . .	20
3.3.2.	Uso de JavaScript dentro de A-Frame . . . . .	22
3.4.	BBC Micro Bit . . . . .	24
3.4.1.	Pyautogui . . . . .	24
3.4.2.	Bitio . . . . .	24
3.4.3.	Micropython . . . . .	25
3.4.4.	Programación del Micro:bit mediante bloques(Makecode) . . . . .	26
<b>4.</b>	<b>Diseño y desarrollo de los talleres</b>	<b>27</b>
4.1.	Introducción y propósito de los talleres . . . . .	27
4.2.	Taller 1 (Creación de un objeto 3D con Beetleblocks. ) . . . . .	28
4.2.1.	Introducción a Bettleblocks . . . . .	28
4.2.2.	El Primer Objeto . . . . .	29
4.2.3.	Formato . . . . .	32
4.2.4.	Blender . . . . .	32
4.3.	Taller 2 (Creación de un entorno VR ) . . . . .	33
4.3.1.	Creación de entorno VR con A-Frame . . . . .	33
4.3.2.	Explorando A-Frame . . . . .	33
4.3.3.	Animación básica con A-frame. . . . .	35
4.3.4.	Llevar la experiencia VR a un smartphone . . . . .	37
4.4.	Taller 3(Micro Bit como IU en entorno VR) . . . . .	39
4.4.1.	Instalación de Bitio . . . . .	40
4.4.2.	Instalación de Pyautogui . . . . .	42
4.4.3.	Programación del Micro:bit . . . . .	43
<b>5.</b>	<b>Conclusiones</b>	<b>49</b>
5.1.	Consecución de objetivos . . . . .	49
5.2.	Aplicación de lo aprendido . . . . .	50

## ÍNDICE GENERAL

5.3. Lecciones aprendidas . . . . .	50
5.4. Trabajos futuros . . . . .	51
<b>Bibliografía</b>	<b>53</b>
<b>A. Guía de uso Beetleblocks</b>	<b>55</b>
A.1. Movimiento . . . . .	55
A.2. Control . . . . .	56
A.3. Colores . . . . .	56
A.4. Figuras . . . . .	57
A.5. Funciones . . . . .	57
A.6. Crear una variable . . . . .	57
A.7. Crear un bloque . . . . .	58
A.8. Operadores . . . . .	59
<b>B. Auriculares en Beetle Blocks</b>	<b>61</b>



# Índice de figuras

1.1. BBC Micro:bit . . . . .	4
3.1. Puntuacion media de la evaluación de estudiantes a preguntas de programación	14
3.2. Tasa de rendimiento medio para estudiantes en diferentes conceptos . . . . .	14
3.3. Inicio Beetle Blocks . . . . .	15
3.4. Escarabajo del área de visualización 3D . . . . .	16
3.5. Bloque para moverse en los ejes . . . . .	17
3.6. Bloque para crear círculos . . . . .	17
3.7. Bloques de las figuras predeterminadas de BeetleBlocks . . . . .	18
3.8. Bloques e imagen de auriculares . . . . .	18
3.9. Blender . . . . .	19
3.10. Exportar objeto . . . . .	19
3.11. Editor Mu . . . . .	25
3.12. Makecode . . . . .	26
4.1. Crear perfil e iniciar BeetleBlocks . . . . .	28
4.2. Inicio Beetle Blocks . . . . .	29
4.3. Orientación y movimiento en Beetle Blocks . . . . .	29
4.4. Bloque combinado . . . . .	31
4.5. Bloques útiles . . . . .	31
4.6. Sprite para crear figuras . . . . .	31
4.7. inicio de Blender y exportación . . . . .	32
4.8. Cubo y esfera primitives en A-Frame . . . . .	34
4.9. Cursor en una escena de A-Frame . . . . .	37

## ÍNDICE DE FIGURAS

4.10. Ngrok . . . . .	38
4.11. Funciones del Micro:bit . . . . .	39
4.12. Resoluciones . . . . .	45
4.13. Valores del acelerómetro . . . . .	46
A.1. Bloques Beetle Blocks A . . . . .	60
A.2. Bloques Beetle Blocks B . . . . .	60



# 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, es una experiencia que luego puede enriquecerse con otros dispositivos.

En los últimos años se ha 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. Estos van desde cascos de realidad virtual, a gafas de plástico o cartón 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 grandes marcas como HTC o Sony se animaron a sacar estos dispositivos a la venta para el público.

A la hora de hablar de la realidad virtual en este trabajo de fin de grado se ha profundizado en las denominadas experiencias, se trata de entornos de realidad virtual que nos permiten situarnos en medio de la escena, de la que podemos ser espectadores o interactuar dentro de ellas, 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, punto en el que se centra este trabajo y que se describe en la siguiente sección.

### 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*<sup>1</sup>, 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. Este trabajo de fin grado se va a dar uso a dos de ellas.

Principalmente BeetleBlocks<sup>2</sup> 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 como se puede leer en el artículo [1], será uno de los núcleos de este trabajo y la que es usada para llevar a cabo el diseño de las figuras en este trabajo. La segunda Blender<sup>3</sup> un software Open Source de creación 3D más clásico y cuyo uso se limitará a pequeños retoques a las figuras creadas previamente con BeetleBlocks.

---

<sup>1</sup><https://unimersiv.com/>

<sup>2</sup><http://Beetleblocks.com/>

<sup>3</sup><https://www.blender.org/>

Más adelante en este trabajo se describe la manera de crear objetos con BeetleBlocks en el capítulo 3.2, estos irán desde figuras geométricas básicas, a formas tan complejas como uno quiera debido a la versatilidad de la plataforma.

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, esta afirmación queda fundamentada en este par de artículos [4] [11].

### 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 se puede aprender a crear un entorno de realidad virtual básico en el que se pueden exportar todo tipo de figuras. La creación del entorno de realidad virtual con A-Frame se llevará a cabo en el capítulo 3.3

## 1.3. Micro Bit

“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 se puede encontrar cuando se entra en su página oficial <sup>4</sup>, y eso es Micro:bit, un sistema de Hardware embebido, diseñado por la BBC para la enseñanza informática en el Reino Unido.

---

<sup>4</sup><https://microbit.org/es>

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, la mayoría de esto puede verse en la Figura 1.1.

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. Y es Open Source, por supuesto. Tanto el hardware como el software de micro:bit es de código abierto un requisito indispensable ya que como todas las tecnologías utilizadas en este trabajo, esto ha ayudado a que todo el que realice los talleres pueda continuar experimentando en su propio ordenador una vez han sido finalizados.



Figura 1.1: BBC Micro:bit

### 1.3.1. Micro Bit como interfaz de usuario

Como ya hemos visto Micro Bit 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 previamente es creada mediante A-Frame.

A la hora de buscar una funcionalidad para el Micro Bit el alumno poseía un ejemplo de movimiento físico en un entorno de A-frame, tras el estudio de las posibilidades que ofrece la programación del Micro Bit se ha conseguido implementar una serie de funciones extras. Al finalizar el último de los talleres El Micro Bit permitirá un movimiento dentro de la experiencia tanto físico como de la cámara mediante el acelerómetro integrado, también se añade un método de interacción con el objeto creado con BeetleBlocks y animado con A-Frame.

Para hacer esto se ha realizado un programa en micropython una variación de Python adaptada al Micro Bit que nos permite ejecutar programas de python en nuestro dispositivo mediante a la importación de una serie de módulos de Python todo esto se describe en el la sección 3.4.4, pero esto es solo un ejemplo de la versatilidad del Micro Bit.

Como forma de apoyo en el aprendizaje de micropython y diversas funciones y metodología de Python se ha hecho uso de las siguientes documentaciones [9] [5].

## 1.4. Estructura de la memoria

Tras esta introducción en la que se ha visto una visión de la realidad virtual origen y evolución, así como la descripción y significado de las experiencias VR y su continua introducción en la educación como método didáctico y de gran valor práctico. Seguidamente se describen las herramientas de diseño 3D BeetleBlocks y Blender y se describe el diseño de objetos en 3D mediante bloques ya que es el método que se usará en este trabajo de fin de grado y el cual resulta perfecto para introducirse en la creación y diseño de estas figuras. La última tecnología a describir es el Micro Bit y dentro de sus múltiples posibilidades, se explica el porqué introducirlo en una experiencia de A-Frame y es para darle un uso como interfaz de usuario con esto se consigue complementar la experiencia de realidad virtual.

En el siguiente capítulo 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 se ofrece una explicación del proceso de estudio de las diferentes tecnologías que se han usado en este trabajo así como un breve explicación de uso o práctica de cada una de ellas para ayudar a comprender su funcionamiento. Se puede dividir por tanto en tres partes diferenciadas.

- Sección 3.2.

Su contenido corresponde al diseño de objetos en 3D con la descripción de las dos tecnologías usadas BeetleBlocks y Blender.

- Sección 3.3.

Centrada en A-Frame en esta sección se describe el método y creación de experiencias de realidad virtual usando este *framework*.

- Sección 3.4.

Se describen los distintos métodos de programación del Micro:bit y se definen los módulos que se usan para implementar la función de controlador.

En el capítulo 4 se encuentra la serie de 3 talleres o prácticas de las que consta el núcleo práctico de este trabajo.

El contenido y tecnología implementada de cada uno puede verse en la Tabla ?? y es el siguiente

- Primer taller, sección 4.2, el objetivo de este taller es la creación de una figura 3D mediante BeetleBlocks y su método de programación mediante bloques.
- Segundo taller, sección 4.3, la pretensión de este taller es conseguir impartir el método de creación de una experiencia de realidad virtual haciendo uso de las herramientas que ofrece A-frame.
- Por último un tercer taller, sección 4.4, centrado en el proceso de programación del Micro Bit para implementarlo en A-Frame como controlador y así poder moverse con él.

	TALLER 1	TALLER 2	TALLER 3
Tecnología	BeetleBlocks	A-Frame	Microbit
Contenido	Diseño de un objeto 3D	Creación de experiencia VR	Programar Micro:bit como IU

Cuadro 1.1: Talleres didácticos

El capítulo 5 corresponde a las conclusiones, es donde se analizan los resultados obtenidos y si por tanto se ha alcanzado a conseguir los objetivos propuestos previamente.

Además se expone la aplicación de los conocimientos obtenidos por el alumno durante la realización de este grado que han sido fundamentales para la consecución de este trabajo de fin grado. Los dos últimos puntos corresponden a los conocimientos útiles aprendidos propios de este trabajo y a las posibles mejoras y trabajos futuros relacionados con los expuestos aquí que se podrían realizar o aplicar en un futuro.

Las últimas secciones de este trabajo corresponden a la Bibliografía y los Apéndice A y Apéndice B en el que se describe de una forma más detallada una guía de uso y descripción de todos los bloques de BeetleBlocks y el proceso de creación de un objeto en 3D con BeetleBlocks.





# Capítulo 2

## Objetivos

### 2.1. Objetivos

Una vez han sido descritas las experiencias de realidad virtual, el diseño 3D así como las tecnologías que van a ser usadas en este trabajo, el siguiente paso es describir la serie de objetivos que se proponen conseguir con este trabajo de fin de grado.

#### 2.1.1. Exponer la utilidad de la 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 y experimentar con un entorno de realidad virtual de una forma práctica.

Gracias a la novedad y el gran valor interactivo que proporciona la realidad virtual la retención de las tecnologías aplicadas por los alumnos son retenidas de una manera efectiva.

#### 2.1.2. Creación de un objeto 3D y una experiencia VR

El segundo objetivo a conseguir es lograr crear una dirección didáctica en la que los alumnos de enseñanzas medias sean introducidos a la creación de objetos 3D y de una experiencia de realidad virtual.

Para conseguirlo se ha optado por la programación mediante bloques para el diseño 3D en base a este artículo [1] que corrobora su efectividad como método introductorio a la hora de realizar un primer acercamiento a este campo.

### 2.1.3. Implementar el Micro:bit dentro de A-Frame

El tercer y último objetivo está relacionado con el Micro:bit. La introducción de este dispositivo se introduce en este trabajo gracias a la recomendación del tutor.

A la hora de implementar alguna función del Micro:bit que estuviera relacionada con los dos talleres previos, se concluyó que la mejor opción es darle funciones de controlador o interfaz de usuario en la experiencia de A-Frame.

El objetivo por tanto es integrarlo en A-Frame programándolo con su lenguaje propio micropython. Esto llevará a cabo una investigación previa del alumno de este lenguaje y sus diferentes módulos.

## 2.2. Motivación

La principal motivación para llevar a cabo la realización de este trabajo de fin de grado, es la propia tecnología en sí, el alumno siempre se ha visto interesado en las temáticas tratadas, tanto el diseño de objetos en 3D, como las experiencias en realidad virtual y la programación embebida.

El gran potencial que tienen, y los múltiples usos que se le pueden dar han sido muy interesantes desde un primer momento, para el alumno ha sido una gran motivación la realización de este trabajo ya que podía aprender acerca de estos ámbitos, que resultaban una novedad en sus conocimientos y además darle una utilidad en forma de talleres para que más gente pueda aprovechar la investigación llevada a cabo.

Otra motivación principal está relacionada con los entornos de desarrollo de las 3 tecnologías principales, BeetleBlocks, A-Frame y Micro:bit incitan a la creatividad debido a su potencial y múltiples funciones con una interfaz sencilla.

## 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 tanto para la memoria como para la serie de talleres.

### 2.3.1. Metodología

La metodología que se ha seguido para la realización de este trabajo de fin de 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.

Antes de la creación de los talleres ha sido necesario un estudio de las tecnologías utilizadas, realización de pequeños tutoriales y selección de las prácticas más adecuadas para el nivel de conocimientos de los participantes.

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 sigue una metodología común que viene dada por los siguientes puntos:

- \* 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 pero este primer planteamiento fue desechado en detrimento de la introducción del Micro: bit, la popularidad y variedad de aplicaciones de este último fue determinante, se llegó a la conclusión de introducirlo como interfaz de usuario por tanto se desarrolló un programa en python que le otorgara funciones de controlador dentro de las experiencias de A-Frame.

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 se consigue que realizando uno de ellos se obtuvieran conocimientos acerca de la tecnología en cuestión y a su vez alguien que realizara los tres tenga una experiencia completa de realidad virtual. El formato de los talleres es del tipo Scratch.

[...]

## Capítulo 3

# Creación de experiencias VR

### 3.1. Tecnologías

Este capítulo se centra en presentar y desarrollar el software y el hardware que va a ser usado en este trabajo, además se ofrece su forma de uso y varios ejemplos para facilitar su comprensión.

### 3.2. Diseño 3D

Dentro del diseño 3D, se ha decidido dividir tres partes diferenciadas correspondientes a los diferentes puntos en los que se divide esta sección.

En primer lugar en el punto 3.2.1 se trata la programación mediante bloques, se explica la razón por la que es un método válido a la hora de ofrecer un taller didáctico centrado en introducir el diseño 3D.

BeetleBlocks es una herramienta que usa la programación mediante bloques para el diseño de objetos 3D, tanto el análisis como el uso de esta herramienta es tratado en 3.2.4 y supone el núcleo del diseño 3D realizado por el alumno en este trabajo.

Por último en la subsección 3.2.5 se expone Blender, la segunda de las herramientas de diseño 3D tratadas en este trabajo, su uso en este caso se limita a proporcionar el formato buscado para el objeto creado con BeetleBlocks, aunque cabe recalcar que se trata de un software de diseño mucho más complejo que este último.

### 3.2.1. Programación mediante bloques

La programación mediante bloques, facilita mucho las cosas si no se tiene experiencia de ningún tipo con la programación, en este trabajo de fin de grado aparece tanto en BeetleBlocks como en uno de los métodos de programación del Micro: bit.

Artículos como los siguientes [1] [2], evidencian que el surgimiento de este tipo de programación esta empezando a introducirse en el comienzo de la docencia de la programación y los resultados obtenidos son mejores en todos los conceptos de la programación, como se puede observar en la siguientes Figuras 3.1 y 3.2.

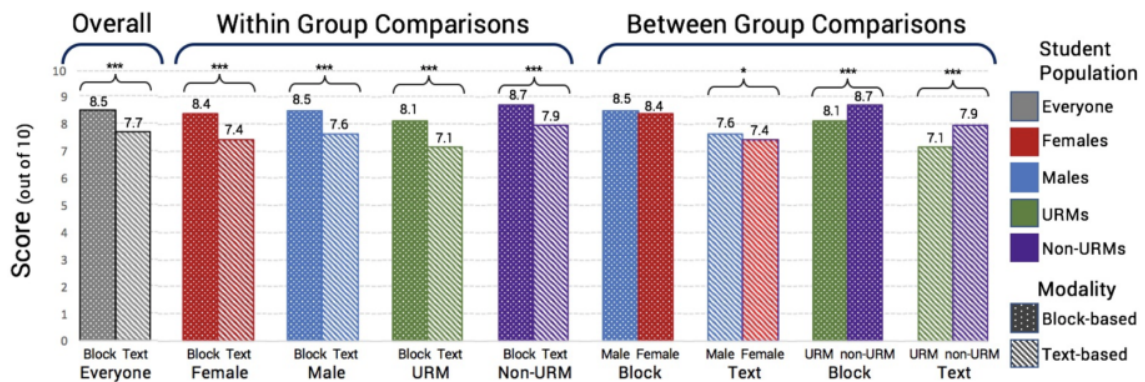


Figura 3.1: Puntuacion media de la evaluación de estudiantes a preguntas de programación

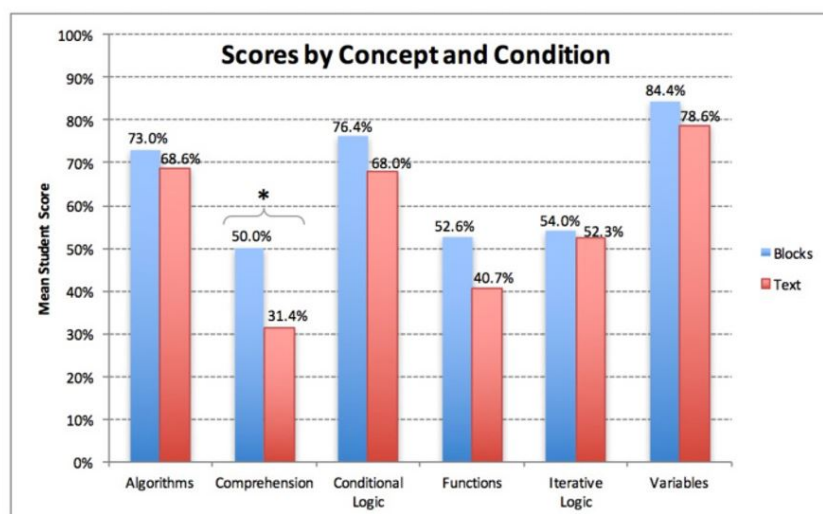


Figura 3.2: Tasa de rendimiento medio para estudiantes en diferentes conceptos

### 3.2.2. BeetleBlocks

Beetle Blocks es un entorno de programación visual por bloques que permite diseñar formas tridimensionales este proyecto es creado por Eric Rosenbaum, Duks Koschitz, y Bernat Romagosa, además de la ayuda en la programación de Jens Mönig. De acuerdo con lo que expone Bernat Romagosa, desarrollador principal de este software, en el artículo [6], Beetle Blocks es más que una herramienta: es también una puerta de entrada muy atractiva al mundo de la programación. El editor de BeetleBlocks está basado en Scratch e implementado usando Snap! y ThreeJS.



Para empezar a usar la herramienta basta con acceder a la web <sup>1</sup> y hacer click sobre “Run BeetleBlocks”. La pantalla mostrada al arrancar Beetleblocks corresponde a la Figura 3.3, se pueden 3 áreas diferenciadas a la izquierda el área de los bloques en el que se muestran los bloques correspondientes a la categoría seleccionada, en el centro el área de trabajo para colocar los sprites o conjuntos de bloques, y a la derecha el área de visualización 3D, para observar el resultado de computación de los bloques creados.

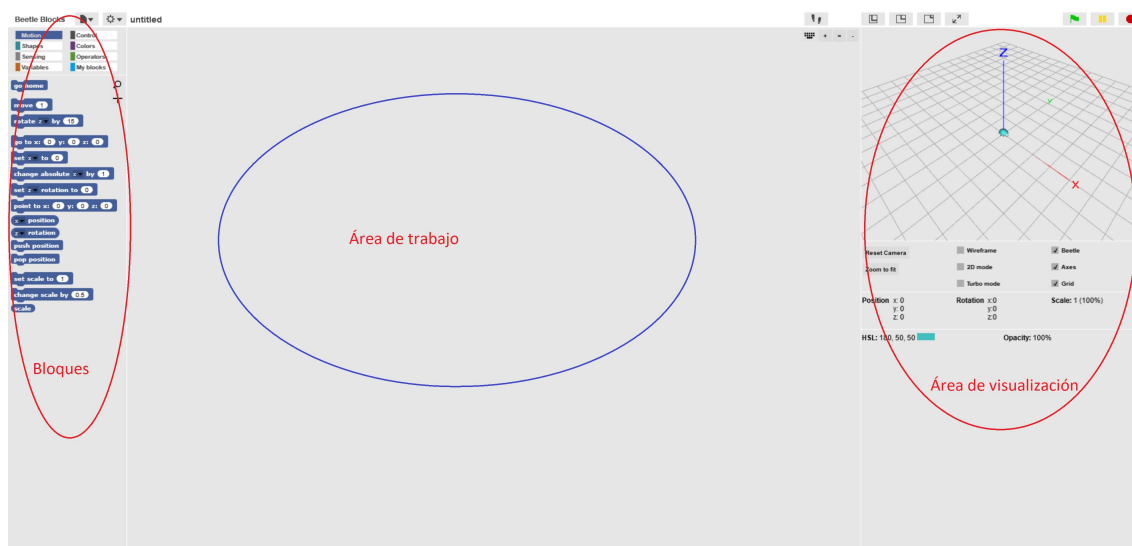


Figura 3.3: Inicio Beetle Blocks

Es recomendable crear una cuenta en BeetleBlocks antes de empezar a usarlo para este trabajo la cuenta que se ha usado ha sido con el usuario **jonybob** y Contraseña **TFG2490**.

<sup>1</sup><http://beetleblocks.com/>

### 3.2.3. Categorías de los bloques

Como ya se expone anteriormente en el trabajo Beetle Blocks utiliza la programación mediante bloques para crear los objetos en 3D, estos tienen diferentes tipos de categorías según su función, el fundamento de Beetleblocks es programar un escarabajo virtual (Figura 3.4) para que, con su movimiento, vaya generando formas tridimensionales. Las diferentes categorías de los bloques y su función es la siguiente:

1. Motion: Donde se engloban los bloques para movernos por la malla
2. Shapes: Figuras y formas predeterminadas.
3. Sensing: funciones para usar con los bloques.
4. Variables: creación y uso de variables de entorno.
5. Control: operadores para los conjuntos de bloques, eventos, funciones etc..
6. Colors: Bloques para dar color a las figuras.
7. Operators: operadores matemáticos para usar en los bloques.
8. Myblocks: bloques propios creados en el proyecto.

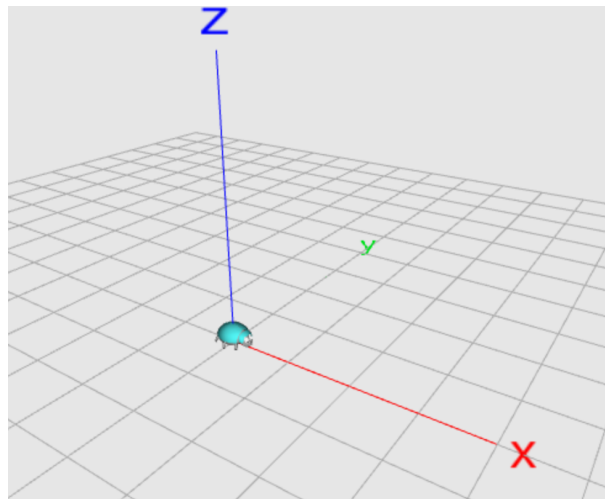


Figura 3.4: Escarabajo del área de visualización 3D

En el Apéndice A se puede ver en más profundidad la descripción y uso de todos los bloques.



### 3.2.4. Prácticas de aprendizaje en BeetleBlocks

A la hora de aprender el uso de Beetle Blocks el alumno comenzó diseñando figuras básicas para familiarizarse con el uso de los distintos bloques. A continuación se exponen una serie de ejemplos prácticos que se utilizaron para el aprendizaje del uso de la herramienta, se muestra como crear figuras básicas, moverse en los 3 ejes espaciales (x, y, z ) y el uso de diferentes funciones. y por último un objeto que usa todo lo aprendido para la creación de unos auriculares.

#### ■ Movimiento por los ejes

Dependiendo de como sea la orientación de las figuras dibujadas es necesario moverse por los ejes y orientar el escarabajo que como ya se sabe sirve como puntero de orientación para dibujar. El bloque que se usa para moverse y situarse en donde se quiere se puede observar en la siguiente Figura 3.5



Figura 3.5: Bloque para moverse en los ejes

#### ■ Círculos

Para crear círculos de una manera muy sencilla simplemente basta con empezar a dibujar curvas o líneas, con los bloques correspondientes y moverse hasta completar los 360 grados del círculo girando  $x$  número de grados uno de los ejes. En el ejemplo de la Figura 3.6 se dibujaría un círculo rojo moviéndose 0.25 hacia delante cada vez que se gira 18 grados, esto se hace 20 veces para completar los 360 grados ( $18^\circ * 20 = 360^\circ$ ).

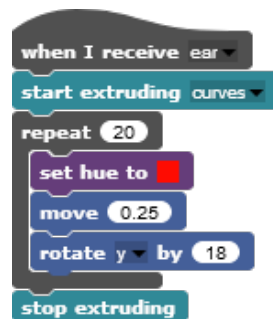


Figura 3.6: Bloque para crear círculos

### ■ Figuras predeterminadas

Las figuras predeterminadas que nos ofrece Beetleblocks son el cubo (en dos formas), el cilindro y la esfera cuyos bloques se corresponden con los de la Figura 3.7 en orden respectivo de arriba hacia abajo. Estas figuras permiten hacer varias figuras más complejas combinándolas entre ellas.



Figura 3.7: Bloques de las figuras predeterminadas de BeetleBlocks

### ■ Auriculares

Como parte final en el aprendizaje de Beetleblocks, se ha creado una figura más compleja para corroborar lo aprendido. La figura en cuestión son unos auriculares, su proceso de creación al detalle se expone en el Apéndice B, y el objeto y bloques que lo componen se puede ver en la Figura 4.3

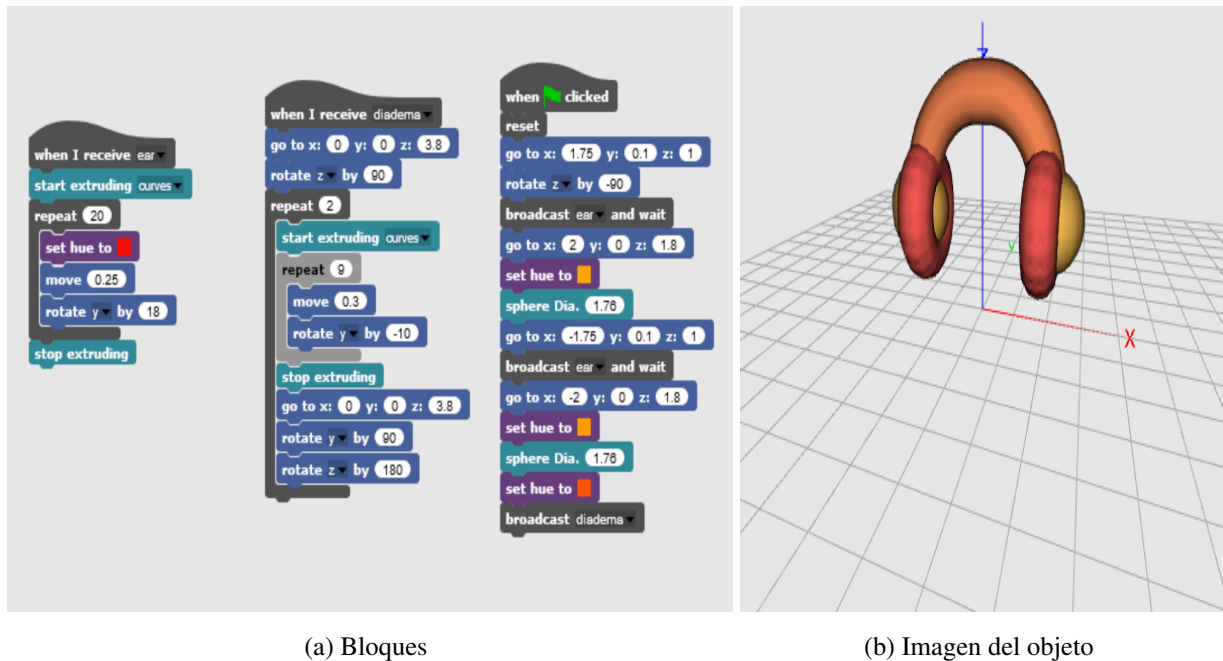


Figura 3.8: Bloques e imagen de auriculares

### 3.2.5. Blender

Blender es una potente herramienta open source de creación 3D. En este trabajo de fin de grado se le ha dado un uso muy concreto, se trata de convertir la figura creada con BeetleBlocks a un formato compatible con A-Frame ( *.obj* y *.mtl* ). Como es *Open Source* su documentación [8] es accesible para todo aquel que la desee y es la que se ha utilizado para este trabajo de fin de grado. La pantalla mostrada al iniciar Blender se puede ver en la Figura 3.9

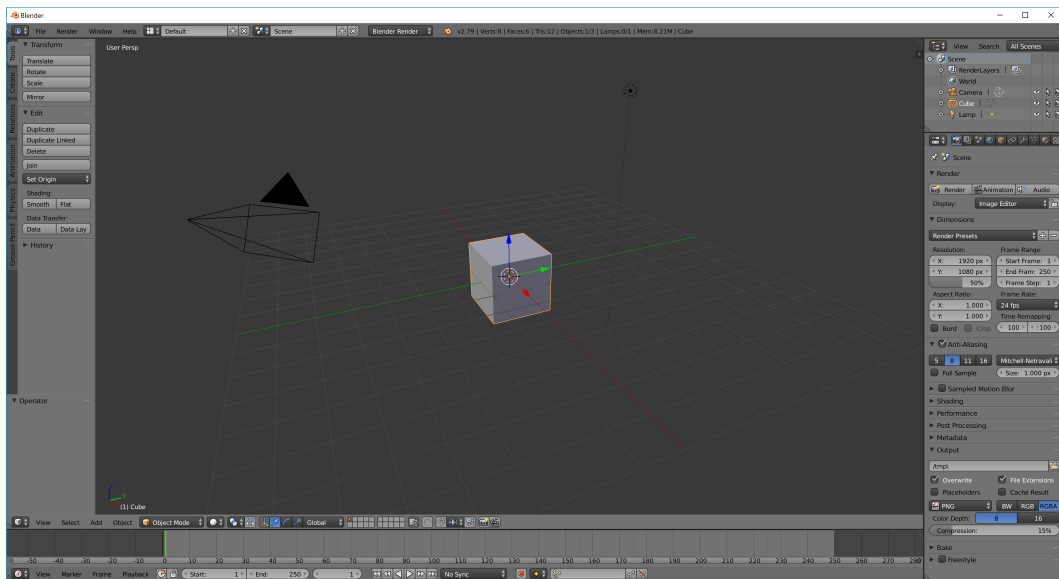


Figura 3.9: Blender

Estos son los pasos necesarios para exportar la figura en el formato correcto:

1. Borrar el cubo, posicionando el ratón encima y presionando **x** para borrarlo.
2. Se pulsa *espacio* y se introduce en el cuadro de texto **import stl**, se importa la figura de Beetle Blocks previamente creada y exportada con un formato *stl*.
3. Se pulsa *espacio* de nuevo y se introduce **export obj** en el cuadro de texto, como se ve en la Figura 4.7b se deben marcar las *Checkboxes Write Normals, Write Materials*.

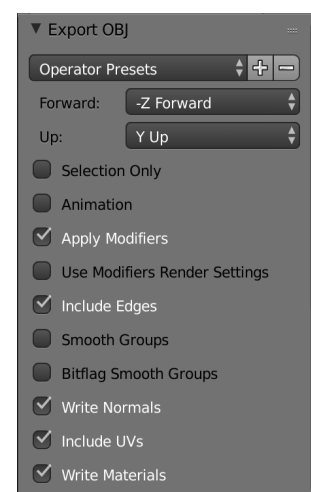


Figura 3.10: Exportar objeto

### 3.3. A-frame

A-Frame es un framework que permite crear experiencias de realidad virtual. Originalmente fue concebido dentro de Mozilla Firefox, y ahora es administrado por desarrolladores de Supermedium (Diego Marcos, Kevin Ngo) y Google (Don McCurdy). A-Frame fue diseñado para ofrecer una manera potente de desarrollar contenido VR. Al ser una plataforma independiente Open Source, A-Frame ha crecido hasta convertirse en una de las comunidades de realidad virtual más grandes.

Se integra en un documento HTML haciendolo simple de usar en primera instancia, aunque el núcleo es un potente framework ECS<sup>2</sup> para Three.js donde los desarrolladores de software pueden crear escenas WebVR y 3D usando HTML.

Por tanto se compone de elementos propios basados en HTML y de la importación de archivos en Javascript para complementar y añadir funcionalidades, todo esto se puede ver de una forma más detallada en su documentación [3].

A-Frame da soporte a la mayoría de dispositivos de realidad virtual como Vive, Rift, Windows Mixed Reality, Daydream, GearVR, Cardboard, Oculus Go, e incluso puede ser usada para realidad aumentada. El objetivo de A-Frame es definir una experiencia inmersiva completa yendo más allá del contenido en 360°, y dando soporte completo para controladores y seguimiento posicional.

#### 3.3.1. A-frame dentro de HTML

A-Frame se integra dentro del documento HTML, para usarlo es necesario importarlo en la cabecera del documento:

```
<head>
  <script src="https://aframe.io/releases/0.8.0/aframe.min.js">
  </script>
  [...]
</head>
```

---

<sup>2</sup>Entity Component System

De entre los principales conceptos y elementos que introduce A-Frame se van a destacar cuatro de ellos: las escenas <sup>3</sup>, las entidades <sup>4</sup> y los componentes <sup>5</sup> y los primitivos <sup>6</sup>.

- *Escene*: es el núcleo de A-Frame, el lugar donde se desarrolla todo, todas las entidades son creadas dentro de la escena, por lo tanto sería el “escenario” vacío de la experiencia 3D creada. Para crear una escena dentro del documento HTML hay que añadirla en el `<body>` del documento.

```
<html>
  <head> [...]
  </head>
  <body>
    <a-scene>
    </a-scene>
  </body>
</html>
```

- *Entity*: como se especifica en el patrón ECS (Sistema-entidad-componente), las entidades son objetos a los que se le añaden componentes, por defecto las entidades tienen los componentes de posición, rotación y escala, para explicarlo de una forma más sencilla las entidades representan los objetos 3D en la escena. En A-Frame las entidades se representan mediante el elemento `<entity>`. A continuación se muestra cómo añadir un objeto de tipo box, con color rojo, a la escena:

```
<body>
  <a-scene>
    <a-entity geometry="primitive: box" material="color: red">
    </a-entity>
  </a-scene>
</body>
```

---

<sup>3</sup><https://aframe.io/docs/0.9.0/core/scene.html>

<sup>4</sup><https://aframe.io/docs/0.9.0/core/entity.html>

<sup>5</sup><https://aframe.io/docs/0.9.0/core/component.html>

<sup>6</sup><https://aframe.io/docs/0.9.0/introduction/html-and-primitives.html>

- *Component*: Como se define en el patrón ECS, una componente es una porción modular de datos que se añade a una entidad para otorgarla una apariencia, comportamiento y funcionalidad determinada. Las componentes pueden ser de dos formatos, con una propiedad única o con varias. A continuación se muestra un ejemplo de ambas.

```
<body>
  <a-scene>
    <a-entity position="1 2 3"></a-entity>
    <a-entity light="type: point; color: crimson"></a-entity>
  </a-scene>
</body>
```

- *Primitives*: A-Frame proporciona una gran cantidad de objetos predeterminados como `<a-box >` `<a-sky >`, los cuales son identificados como primitivos o también nombrados primitivos en este trabajo, proporcionan una manera sencilla de usar figuras 3D para empezar a usar A-Frame, y admiten la adición de componentes. Los desarrolladores además pueden crear sus propios primitivos. Unos ejemplos de los primitivos precargados son:

- `<a-box>`
- `<a-camera>`
- `<a-circle>`
- `<a-cursor>`
- `<a-cylinder>`
- `<a-image>`
- `<a-ring>`
- `<a-plane>`

### 3.3.2. Uso de JavaScript dentro de A-Frame

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

Un módulo o paquete muy extendido entre los desarrolladores es el `aframe-extras` [?] se trata de una serie de Add-ons y helpers para A-Frame, e incluye componentes de control, modelos predefinidos, primitivos ....

Otro recurso muy utilizado para A-Frame y basado en JavaScript es la importación de components <sup>7</sup>, como se puede ver anteriormente estos nos habilitan una serie de funciones, existe una amplia biblioteca de components para importar, entre ellos hay algunos que se organizan en paquetes o módulos como superframe. <sup>8</sup>

El uso de cada component es particular y puede encontrarse en su documentación [3], pero generalmente para usar un component hay que vincularlo con una entidad `<a-entity>` y configurarlo.

Tomando como ejemplo particle system, su importación y uso sería de la siguiente forma:

- Se importa el componente en el index.html de la misma manera que ya se mostró con A-Frame:

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

- Y a continuación se vincula particle system con una entidad

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

---

<sup>7</sup><https://aframe.io/docs/0.9.0/introduction/entity-component-system.html>.

<sup>8</sup><https://github.com/supermedium/superframe>.

### 3.4. BBC Micro Bit

La última tecnología en introducir en este trabajo de fin de grado es el Micro Bit, ya se expone en el capítulo 1.3 su definición y características, su función y objetivo en este trabajo es programar las funcionalidades necesarias para que se convierta en el controlador del escenario de A-Frame previamente creado.

Dentro del proceso de implementación de esta funcionalidad en el MicroBit cabe destacar dos herramientas fundamentales *pyautogui* y *bitio*, son unos módulos de micropython que aportan diversas funcionalidades.

Para el inicio del desarrollo y estudio de esta funcionalidad se tuvo la ayuda de esta práctica<sup>9</sup>, que se encuentra en el repositorio de GuitHub del tutor. En ella se puede observar el uso de estas dos herramientas y fue idónea para servir de punto de partida por parte del alumno.

El siguiente paso fue llevar a cabo una serie de test y pruebas con Bitio y Pyautogui para aprender sus distintas funcionalidades. En las siguientes secciones 3.4.1 y 3.4.2 se expone más detalladamente documentación y función de estos dos módulos.

#### 3.4.1. Pyautogui

Para aprender a utilizar pyautogui se han utilizado estas documentaciones [7]

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 mapear las pulsaciones de las teclas del teclado o movimientos del ratón a las interacciones con el Micro Bit en forma de gestos y pulsaciones de los botones.

El código fuente se encuentra en el siguiente repositorio de GitHub<sup>10</sup>.

#### 3.4.2. Bitio

Bitio es la segunda herramienta que se ha usado para conseguir convertir nuestro Micro Bit en un controlador, se trata de una librería de Micro Bit para Python, nos permite correr código

---

<sup>9</sup>[github.com/sarehp/microbit-aframe](https://github.com/sarehp/microbit-aframe)

<sup>10</sup><https://github.com/asweigart/pyautogui>



de python en nuestro PC y poder interactuar directamente con el Micro Bit.

Para aprender su uso se siguió la documentación [10] que se encuentra en un repositorio de GitHub creado por David Whale, el encargado de desarrollarlo.

En la sección *Getting Started* se exponen los pasos necesarios para instalarlo en el programa de python, además se explica como hacer la conexión con el Micro Bit y una serie de ejemplos de las funciones propias de bitio como pueden ser leer los valores del acelerómetro o detectar cuando se pulsa uno de los botones.

### 3.4.3. Micropython

Mycropython es el lenguaje de programación propio del Micro Bit. En este trabajo de fin de grado todas las acciones se han desarrollado en este lenguaje y a la hora de realizar los programas que corren las acciones de la placa se ha contado con un apoyo en su documentación [9]. Esta basado en Python [5] se usa de una forma prácticamente idéntica. Cualquier editor de texto es válido para programar los scripts, el recomendado oficialmente es el editor Mu Figura 3.11. 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.

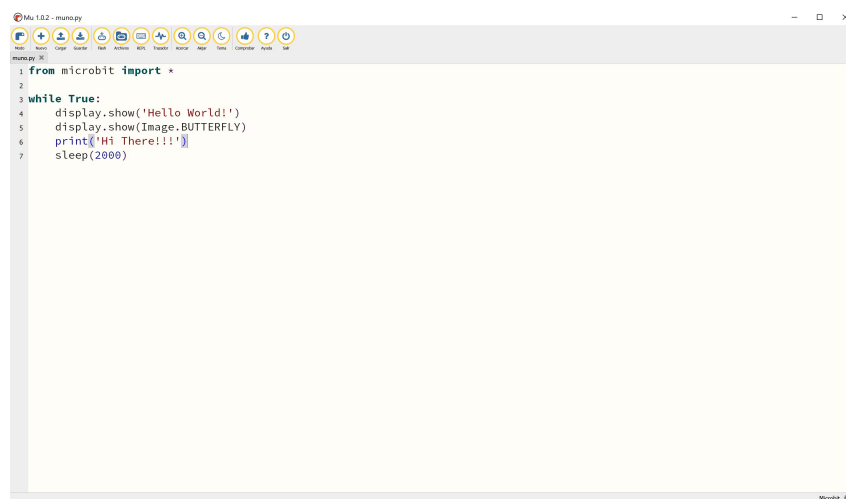


Figura 3.11: Editor Mu

Si no se cuenta con el editor de Mu la forma de cargar los programas en el MicroBit es mediante una terminal, para ello ya sea en Windows como en Linux es necesario tener una compilación de Python instalada en el sistema operativo y lanzar el programa.

### 3.4.4. Programación del Micro:bit mediante bloques(Makecode)

Makecode es un editor para programar el Micro:bit que usa el método de programación por bloques [1] [2].

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 Makecode<sup>11</sup> están disponibles numerosos tutoriales para familiarizarse con su uso. Aunque en este trabajo Makecode no ha sido utilizado para el desarrollo de las funcionalidades, se profundizó en cada uno de los tutoriales ya que ofrecen una mejor visión global de las posibilidades del Micro:bit, pudiendo programarse sencillos juegos de una forma rápida y amena.

En la siguiente figura 3.12 se puede ver su aspecto inicial al arrancarlo, posee 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 y crear los *sprites* (3). Como complemento se puede hacer una previsualización del resultado de la ejecución de los bloques creados(1).

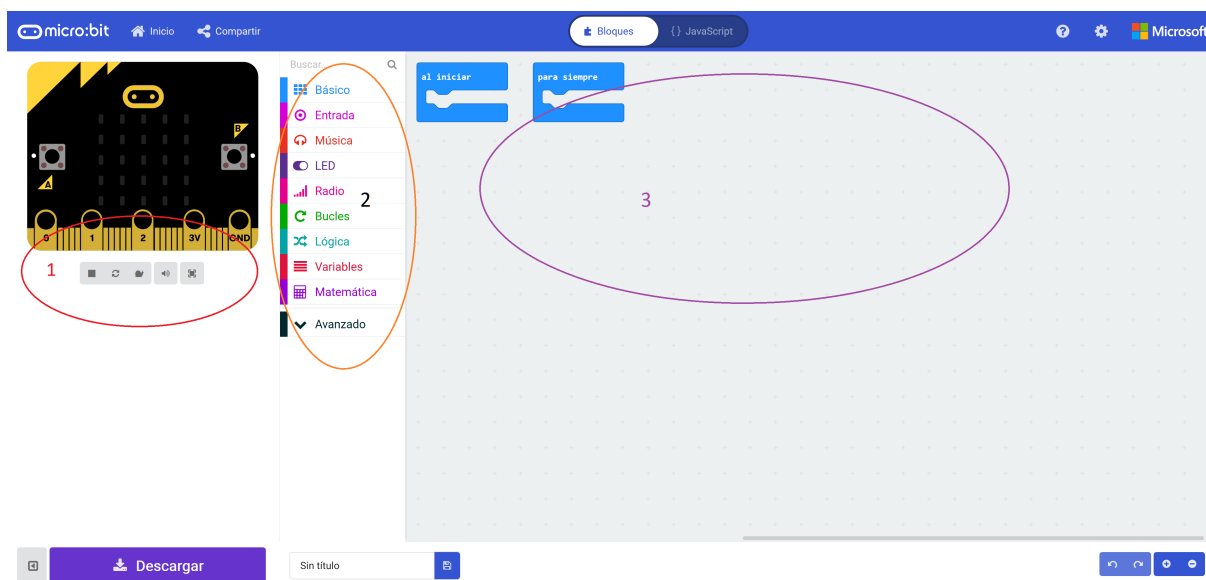


Figura 3.12: Makecode

<sup>11</sup><https://makecode.microbit.org/>

# Capítulo 4

## Diseño y desarrollo de los talleres

### 4.1. Introducción y propósito de los talleres

El principal objetivo de este trabajo de fin de grado es introducir todas estas herramientas que hemos explicado previamente a alumnos de enseñanzas medias, y para ello se han desarrollado una serie de tres talleres, estos son autoconclusivos pero a su vez cada uno de ellos utiliza lo aprendido en el anterior, resultando en que una vez finalizados los tres se tenga una experiencia de realidad virtual completa creada con A-Frame en la que se introduce un objeto creado con BeetleBlocks y que puede ser controlada con un Micro:bit para moverse por ella.

El desarrollo y contenido de los talleres es el siguiente:

- **Taller 1:** Creación de un objeto 3D con Beetleblocks.

Este taller se centra en la creación de un objeto 3D con Beetleblocks, además se aprende a exportarlo y a darle el formato adecuado para usarlo en el siguiente taller.

- **Taller 2:** Diseñar una experiencia VR con A-frame.

En este taller se aprende a usar A-Frame, creando un entorno de realidad virtual en el que se añade el objeto creado anteriormente con BeetleBlocks.

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

En este taller se convierte el Micro:bit en un controlador para la experiencia de realidad virtual creada con A-Frame, el Micro Bit permite moverse por la escena, mover la cámara y una pequeña interacción con el objeto.

## 4.2. Taller 1 (Creación de un objeto 3D con Beetleblocks. )

Este primer taller pretende enseñar a crear un objeto en 3D con BeetleBlocks y su accesible método de programación mediante bloques. Además en la segunda parte del taller se introduce Blender otra herramienta de diseño 3D que permite dar el formato correcto al objeto para usarlo en un entorno VR creado con A-Frame, que será el núcleo del segundo Taller4.3.

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. Introducción a Beetleblocks

Beetleblocks 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 la primera figura, se debe crear una cuenta en Beetleblocks para de una forma accesible poder acceder a los proyectos guardados.

- El primer paso es ir a la web de BeetleBlocks<sup>1</sup> y en la esquina superior derecha pulsar sobre la opción *Log in* para crear el usuario.
- Una vez hecho esto se puede comenzar a usar BeetleBlocks, para ello se debe pulsar en *Run BeetleBlocks* como se ve en la Figura 4.1.



Figura 4.1: Crear perfil e iniciar BeetleBlocks

---

<sup>1</sup><http://BeetleBlocks.com>

### 4.2.2. El Primer Objeto

Una vez abierto BeetleBlocks se muestra una pantalla como la de la Figura 4.2

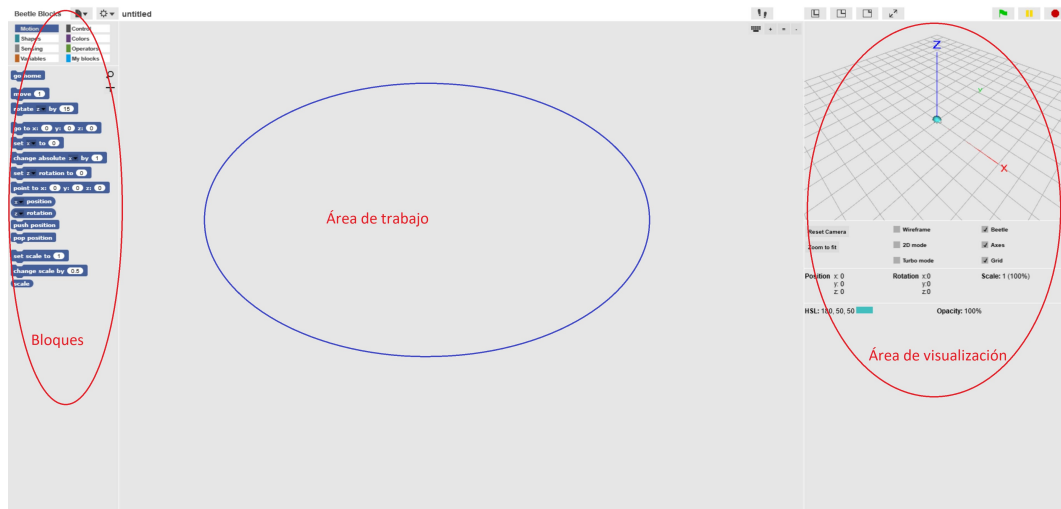


Figura 4.2: Inicio Beetle Blocks

A la izquierda se tiene el menú con los diferentes bloques ordenados por categorías. En el centro se encuentra el área de trabajo en el que se colocan los bloques, y a la derecha una previsualización en 3D del resultado de los bloques colocados en el área de trabajo en ella se puede observar un escarabajo, que resulta muy importante, porque indica la orientación actual respecto a los 3 ejes, siendo su cabeza la dirección hacia la que se está “apuntando”, en la Figura 4.5a la orientación es hacia la parte positiva del eje  $x$  y por lo tanto si se ordena un movimiento positivo con valor 1 el escarabajo avanzaría una casilla en ese eje Figura 4.5b.

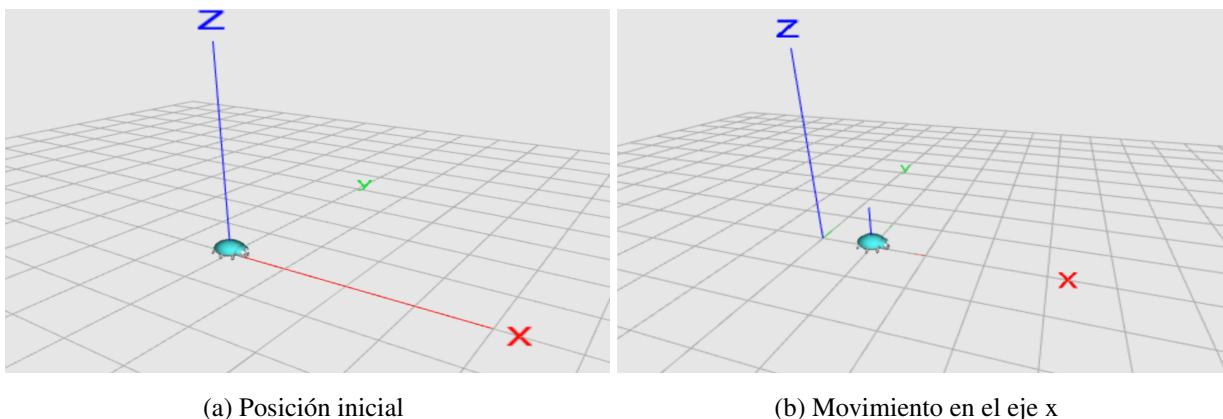










Figura 4.3: Orientación y movimiento en Beetle Blocks

Sabiendo esto ya se puede empezar a crear/programar el primer objeto. A continuación se exponen las ordenes y funciones útiles para familiarizarse con Beetleblocks y crear un primer objeto:

1. Dibujar una línea: usando la orden  que se encuentra en la pestaña de Shapes
2. Creación de una variable: en la pestaña *Variables* se encuentra la opción *make a variable* que nos permite crear una variable global para utilizar.
3. Al crear una variable x, se le puede asignar un valor gracias al bloque  en el que se introduce el valor deseado.
4. Un elemento muy común en la programación son los bucles estos se representan en BeetleBlocks con el bloque , para continuar con la creación de la figura se debe repetir el proceso de la creación de las líneas de una figura geométrica, en el hueco del bloque se inserta la variable creada, esto provoca que todos los bloques que son insertados a posteriori se repiten el número de veces que valor tenga la variable.
5. Dentro del bucle se deben insertar 2 bloques:
  - El bloque  este permite el movimiento y al haber ordenado previamente dibujar se “pintará una línea que tendrá la longitud igual al valor que insertemos en su hueco. En este caso se debe insertar la variable y no un número concreto, más adelante se justifica este paso. El resultado por tanto debe ser el siguiente: 
  - El bloque  permite girar la dirección en la que está orientado el escarabajo, el valor que debe introducirse en cada hueco debe ser en grados. En este caso se quiere girar tantos grados como lados tenga la figura geométrica que se quiere conseguir, para hacerlo de una forma muy sencilla el bloque de división debe contener el número total de grados de un círculo entre los lados de la figura en este caso la variable x previamente creada, quedando de la siguiente manera  sólo queda combinar ambos bloques .

- Por último se combinan los tres bloques, tanto el bucle como los dos resultantes anteriores, el *sprite* o conjunto de bloques final debería quedar como en la Figura 4.4.



Figura 4.4: Bloque combinado

6. Dos bloques que resultan muy útiles para crear cualquier objeto son los de la Figura 4.5 el primero inicia la serie de bloques anidados al hacer click sobre ellos, esto permite un control de la ejecución. Y el segundo que nos deja el plano en blanco, proporciona una manera de inicializar el escenario a su valor por defecto, para prevenir resultados no deseados.



(a) Evento de Click (b) reset

Figura 4.5: Bloques útiles

7. Finalmente, el resultado de la combinación de los diferentes bloques expuestos se puede ver en la Figura 4.6, permite formar un *sprite* que dibuje figuras geométricas con el número de lados que se le da a la variable *x* creada.

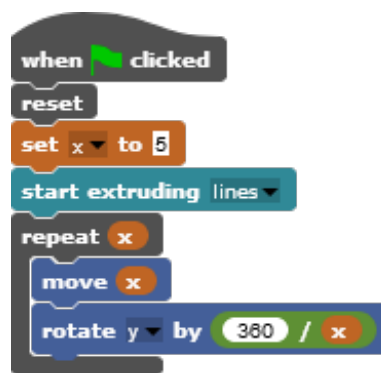


Figura 4.6: Sprite para crear figuras

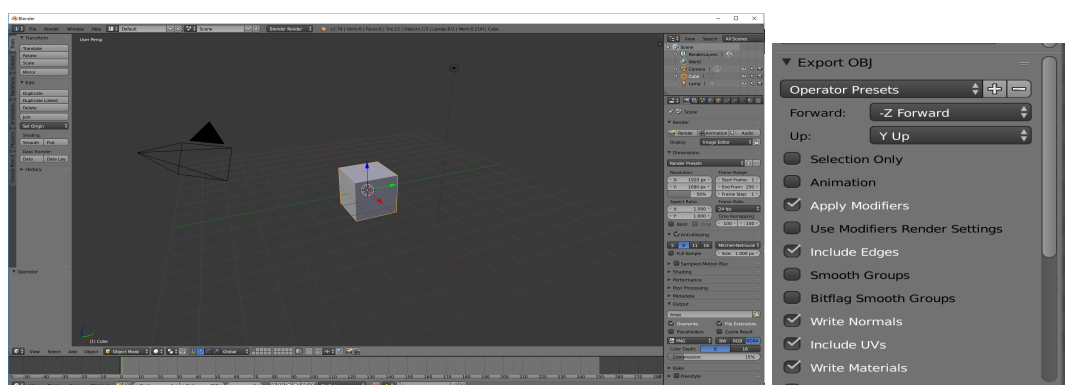
### 4.2.3. Formato

El último paso por exponer es la exportación de la figura final de entre los diferentes formatos que ofrece Beetleblocks, ninguno es compatible con A-Frame, en la siguiente sección veremos como conseguirlo gracias a Blender, pero en este paso es necesario exportarlo en formato *.stl*.

### 4.2.4. Blender

Una vez que se ha creado un objeto con Beetleblocks el paso final es conseguir que tenga una extensión *.obj* que es la apropiada para poder usar el objeto en el siguiente taller, en el que mostrará la manera de crear una experiencia de realidad virtual con A-frame. De entre las múltiples maneras de conseguirlo, en este trabajo se ha optado por usar Blender un software OpenSource de diseño 3D. La pantalla de inicio de Blender se puede ver en la Figura 4.7a.

- Los pasos a seguir para exportar la figura en *.obj* y *.mtl* son los siguientes:
  1. Borrar el cubo, posicionando el ratón encima y presionando **x** para borrarlo.
  2. Se pulsa *espacio* y se introduce en el cuadro de texto **import stl**, se importa la figura de Beetle Blocks previamente creada y exportada con un formato *stl*.
  3. Se pulsa *espacio* de nuevo y se introduce **export obj** en el cuadro de texto, como se ve en la Figura 4.7b, se deben marcar las *Checkboxes Write Normals, Write Materials* y exportar el objeto a la ruta deseada, si todos los pasos se han realizado correctamente, deben aparecer dos nuevos archivos *xxx.obj* y *xxx.mtl*.



(a) Blender

(b) Checkboxes

Figura 4.7: inicio de Blender y exportación



## 4.3. Taller 2 (Creación de un entorno VR )

En este segundo taller se enseñará a crear una experiencia de realidad virtual con A-Frame, además se añadirá el objeto previamente creado con BeetleBlocks en el primer taller, y se indicaran los pasos a seguir para visualizar el escenario de A-Frame en un smartphone.

### 4.3.1. Creación de entorno VR con A-Frame

Para crear la experiencia de realidad virtual con A-Frame el primer paso es crear un documento `index.html` y el repositorio en el que se van a incluir los archivos necesarios. Todos los comandos de terminal descritos en este taller se basan en un terminal Linux, con los siguientes se procede a crear el directorio de la práctica y el archivo `html`.

```
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. Simplemente para empezar a usarlo se debe importar en la cabecera de `index.html` de esta manera:

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

### 4.3.2. Explorando A-Frame

Una vez se ha cargado la versión de A-Frame se puede dar paso a un primer acercamiento. Un ejemplo sencillo para aprender a usar A-Frame consiste en cargar un primitivo, plantillas de elementos que están dentro del código de A-Frame, gracias a ellos se puede crear un cubo, un cilindro, una esfera . . . de una forma muy sencilla. A la hora de usarlos en el `index.html` basta con nombrarlos como si se tratara de una entidad, a los primitivos se le pueden dar diferentes propiedades. Siempre que se carga una figura en A-Frame tiene que ser dentro de la escena, esta se identifica en el documento HTML con `<a-scene>`.

A continuación se expone un ejemplo, en el que se introducen una serie de primitivos como un cubo y una esfera dentro de la escena de A-Frame, y el resultado puede verse en la Figura 4.8.

```
<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>
```

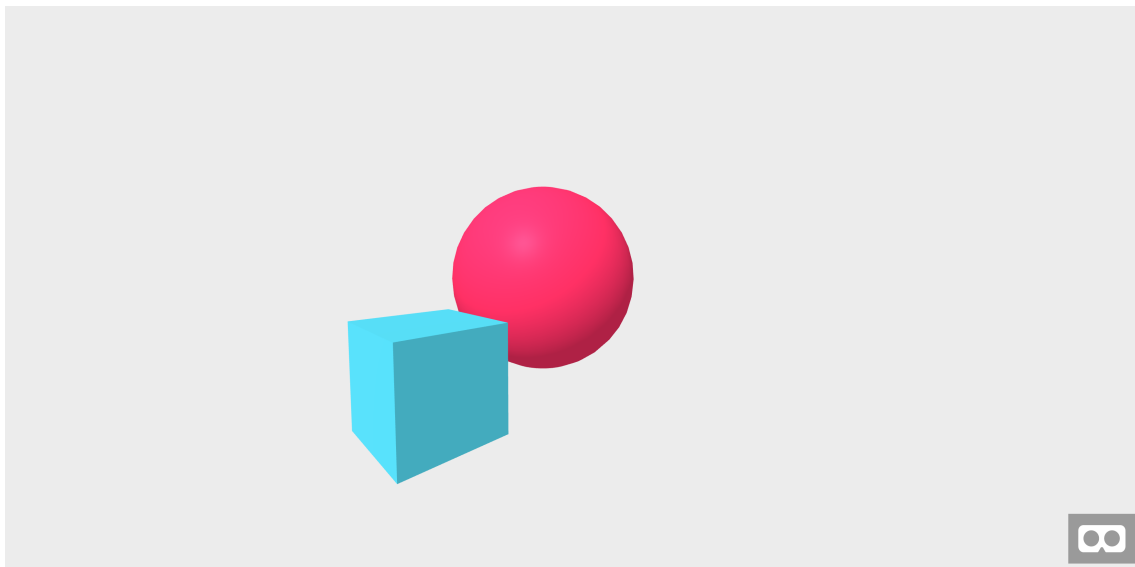


Figura 4.8: Cubo y esfera primitives en A-Frame

Además es posible crear nuestros primitivos propios, algo que se puede ver en la siguiente sección de este taller, para ello es necesario exportar un archivo JavaScript.

Ahora que ya se ha explicado la metodología para cargar objetos en A-Frame, el siguiente paso a realizar es cargar el objeto creado con BeetleBlocks, ya con el formato *.obj* que se le ha dado previamente con Blender.

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

1. Se inicializan las variables *.obj*, *.mtl*, y *textura*.

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

</a-entity>.
```

2. Se cargan el *.obj*, *.mtl*, y la *textura*. Para la textura vale cualquier *.jpg*, es recomendable que se trate de algún tipo de imagen que sea una textura. En la propiedad *src* se indica la ruta de cada archivo a importar. A continuación se puede ver un ejemplo:

```
<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>
```

### 4.3.3. 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.

El ejemplo expuesto aquí consigue aportarle una animación al objeto que consiste en una rotación al hacer click sobre él, para hacerlo se debe importar el paquete `aframe-animation-component`.

Lo importamos en la cabecera `<head>` del documento de la misma forma que se hace con A-Frame.

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

</script>
```

Para usar la animación `<a-animation>` es necesario invocarla dentro del objeto, representado en A-Frame como `<a-entity>`, de la siguiente manera:

```
<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>
```

Como se puede observar uno de los atributos de `<a-animation>` es `begin=click`, este permite que la animación sólo se ejecute cuando se hace click sobre el objeto. Al estar en un entorno de realidad virtual, no se dispone del ratón y lo que funciona sería el equivalente del es el centro de la mirada, para conseguir simular apuntar al objeto con el puntero del ratón es necesario crear un cursor “virtual.”<sup>en</sup> 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 se define la posición inicial de la cámara, en este caso se le da un color rojo para que resalte con el fondo de la escena.

Una vez aclarado este punto la creación del cursor y centrar la cámara es muy sencillo, y debe hacerse dentro del objeto:

```
[...]

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

  <a-cursor color="#FF0000">

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

El resultado final puede verse en la Figura 4.9

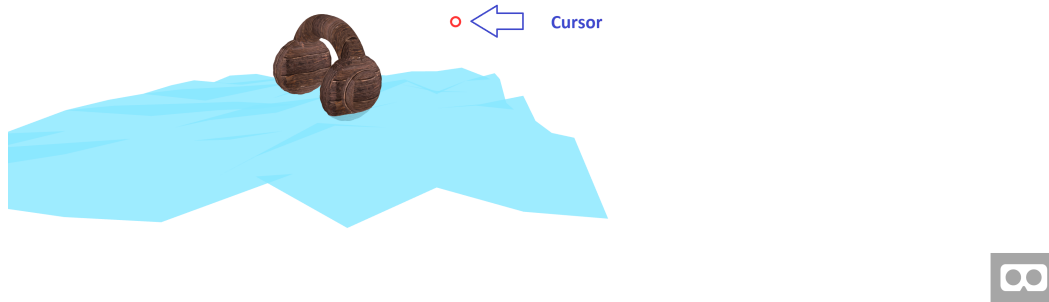


Figura 4.9: Cursor en una escena de A-Frame

#### 4.3.4. Llevar la experiencia VR a un smartphone

Con todo esto se consigue crear un entorno de realidad virtual con A-Frame de forma local. Pero para visualizar un entorno de realidad virtual, es necesario un visor, la forma más rápida y sencilla de ver obtener uno, es a través de un smartphone, para exportar el index.html a internet y poder visualizarla desde el navegador web del teléfono se va a contar con la ayuda una utilidad llamada ngrok.

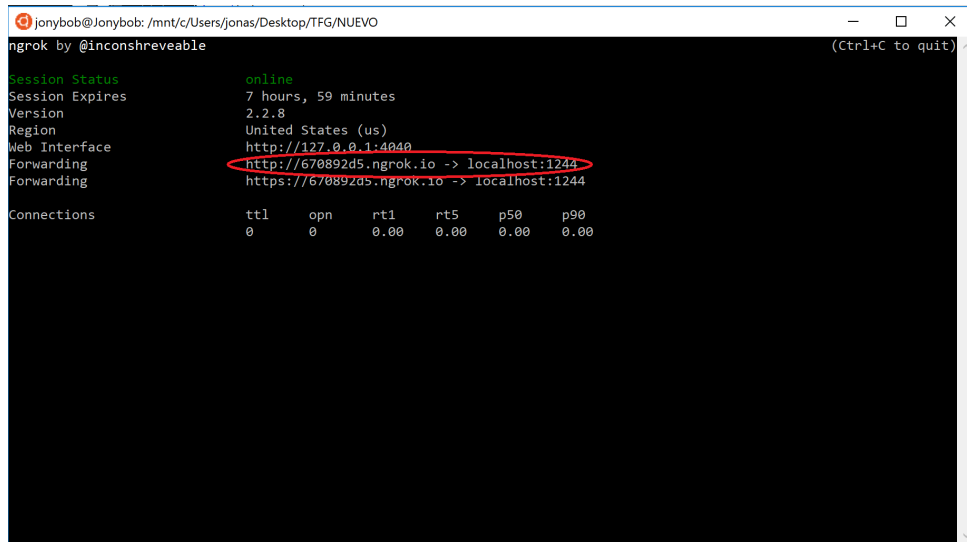
- Estos los pasos necesarios para usar ngrok y exportar la escena de A-Frame:
  1. En una terminal y estando en carpeta la raíz del proyecto, se lanza el servidor de forma local:

```
python -m SimpleHTTPServer 1234
```

2. Instalar ngrok, simplemente descargando el .zip y descomprimiéndolo en la carpeta raíz.
3. Después de esto es necesario ejecutarlo pasándolo como argumento el puerto del servidor local previamente creado.

```
./ngrok http 1234
```

4. Ngrok devuelve una pantalla, entre los datos que proporciona se encuentra la url pública de la aplicación, marcada en rojo en la Figura 4.10.
5. Introduciendo esta url en un navegador compatible, por ejemplo Google Chrome, se puede visualizar la experiencia de A-Frame en un smartphone u otro dispositivo compatible.



The image shows a terminal window titled "jonybob@Jonybob: /mnt/c/Users/jonas/Desktop/TFG/NUEVO". The terminal output displays the Ngrok session status and public URL. The public URL is highlighted with a red circle.

```
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
```

Figura 4.10: Ngrok

## 4.4. Taller 3(Micro Bit como IU en entorno VR)

En este tercer taller se mostrará como convertir nuestro Micro Bit 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 Micro Bit 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 Micro Bit 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 Micro Bit.

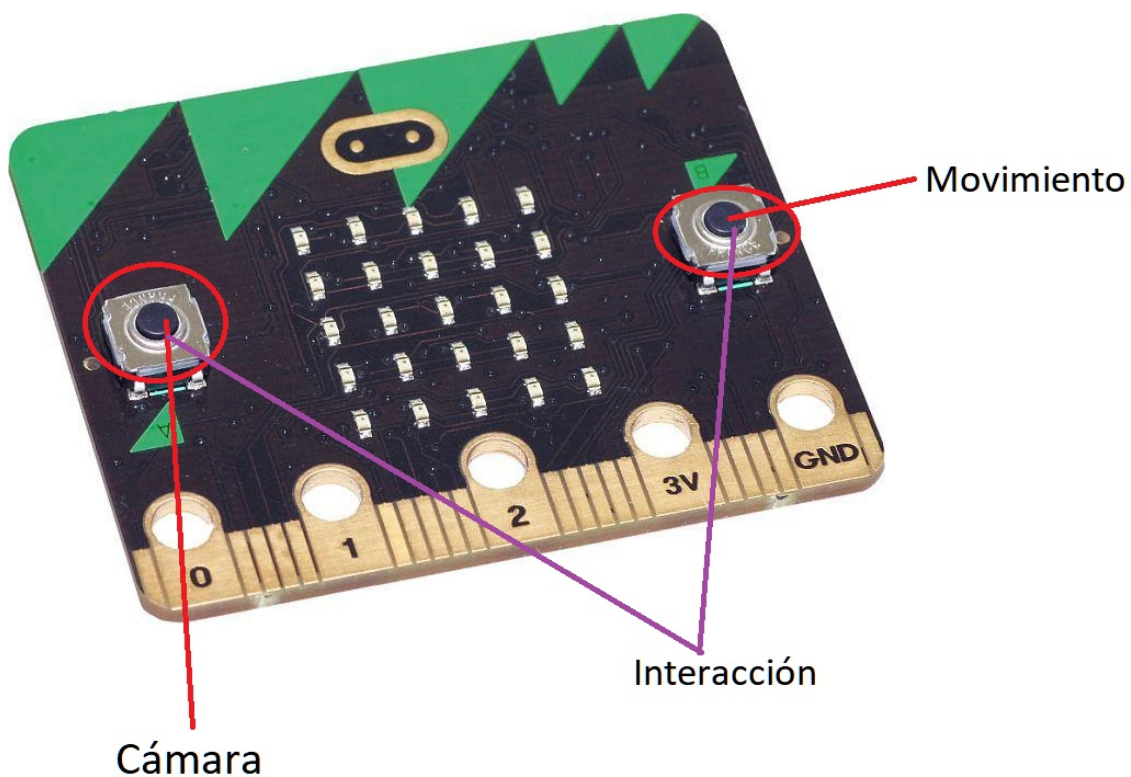


Figura 4.11: Funciones del Micro:bit

Para programar el Micro:bit 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, Micro:bit 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<sup>2</sup>.

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
```

---

<sup>2</sup><https://github.com/whaleygeek/bitio>



```
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 Micro:bit 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 Micro:bit.

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 Micro:bit

Una vez se tiene el entorno preparado y todas las herramientas instaladas se puede proceder a programar el Micro:bit 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 4.11

- 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.

## B) Interacción

La siguiente función en ser implementada es la interacción, en este caso su programación es muy sencilla.

La interacción que fue creada en el anterior taller consistía en una pequeña animación del objeto 3D dentro del escenario creado por A-Frame. La animación es ejecutada mediante un click del ratón cuando se enfoca directamente al objeto.

Por tanto la función a implementar en el Micro:bit es el click del ratón. Hay una función en `pyautogui` que permite esta orden.

```
pyautogui.click()
```

El método de detección que ha sido elegido en este caso es la pulsación simultánea de los dos botones para hacer click. Para capturar una pulsación de un botón Micro:bit posee una función específica para cada uno por lo que en el bloque `if` se deberá incluir las dos en un *and* de la siguiente forma:

```
microbit.button_a.is_pressed() and microbit.button_b.is_pressed() :
```

## C) 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 Micro:bit, 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 Micro:bit. 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 4.12 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)

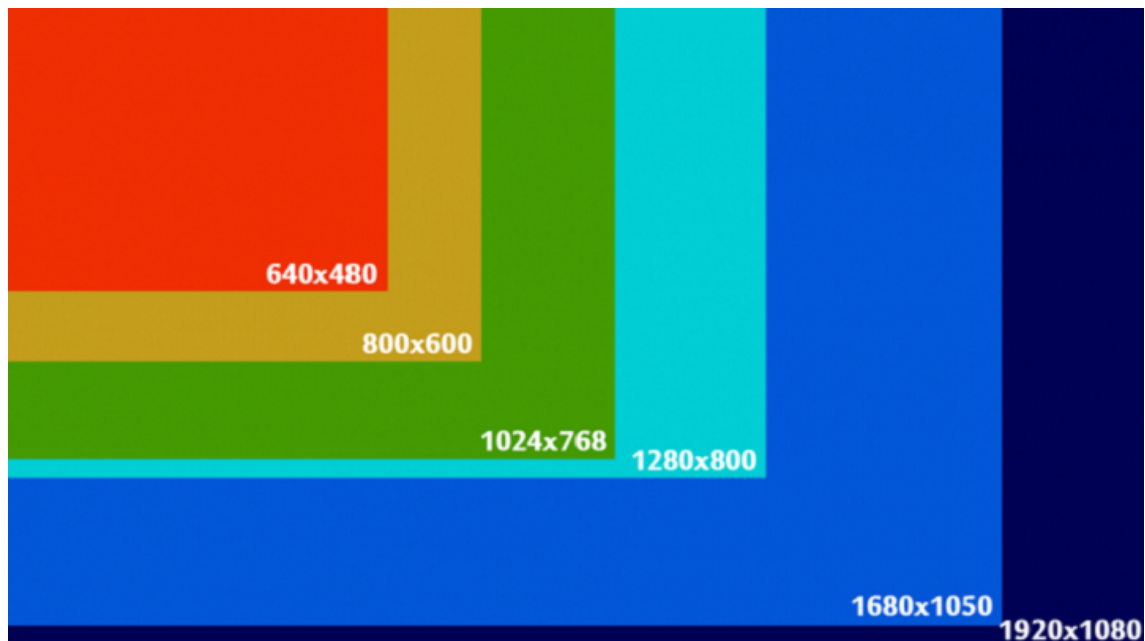


Figura 4.12: Resoluciones

Tomando como base por tanto la resolución Full HD tenemos 1920 pixeles de ancho y 1080 pixeles de alto. Ahora se necesita saber como funciona el acelerómetro del Micro:bit. 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 4.13.

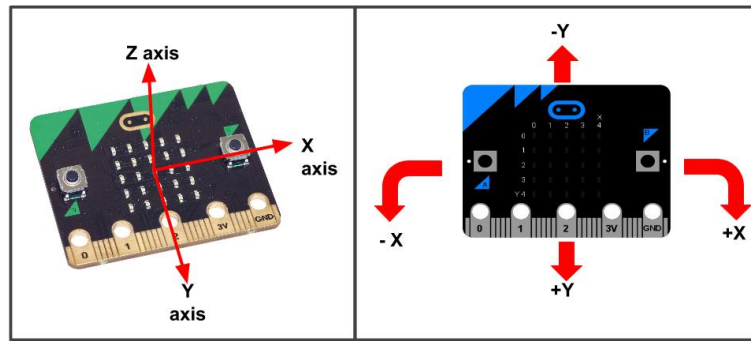


Figura 4.13: Valores del acelerómetro

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.

Una vez se tiene clara la medida del ancho de la pantalla en la que se ejecuta la experiencia de realidad virtual y los valores máximos del acelerómetro el siguiente paso es mover la cámara.

Como se puede ver anteriormente para mover la cámara en A-Frame es necesario mover el ratón mientras se mantiene pulsado el botón izquierdo del mismo para hacer esto, son necesarias estas dos funciones de pyautogui.

```
pyautogui.mouseDown()
pyautogui.moveRel(x, y)
```

Con *pyautogui.mouseDown()* se consigue dejar pulsado el ratón y con *pyautogui.moveRel(x,y)* se mueve el ratón en función de su posición actual en la pantalla.

Una vez se sabe esto, basta con implementar un método de sensibilidad en el acelerómetro parecido al ya creado con el movimiento físico.

```
x = microbit.accelerometer.get_x()
y = microbit.accelerometer.get_y()

if x > SENX:
    pyautogui.moveRel(Height*x, None)
elif x < -SENX:
    pyautogui.moveRel(Height*x, None)
if y > SENY:
```

```
pyautogui.moveRel(None, y*Width)
elif y < -SENY:
pyautogui.moveRel(None, y*Width)
```

Con este código se consigue mover el ratón en todas las direcciones en base a los valores del acelerómetro en el **eje x** y el **eje y**. Las variables *SENX* y *SENY*, representan la sensibilidad en los ejes del acelerómetro del Micro:bit, en este caso fueron fijadas a 180 y 280 respectivamente para prevenir movimientos, de una forma parecida al Threshold, en la anterior función implementada en el Micro:bit del movimiento.

Por último se puede observar que el valor del acelerómetro es multiplicado por dos variables **Height** y **Width**, Estas son variables que ayudan a graduar el grado de sensibilidad del movimiento del ratón y son resultado de la división del ancho y el alto de la resolución de la pantalla usada, por un factor múltiplo de los valores ancho y alto de una pantalla Full HD. Para las pruebas realizadas en este proyecto su valor es el siguiente.

```
screenWidth, screenHeight = pyautogui.size()
Factor_y = 1920*6
Factor_x = 1080*6

Width = screenWidth / Factor_y
Height = screenHeight/ Factor_x
```

Cambiando el valor de estas variables es posible ajustar la sensibilidad del movimiento en función de la pantalla usada, basta con cambiar el múltiplo por uno menor en este caso fue 6 debido a que la pantalla era 4k y la densidad de píxeles era muy elevada, despues de diversas pruebas se confirmó como un valor apropiado para tener un control óptimo e la experiencia, en el que la cámara no se mueva excesivamente rápido.





# Capítulo 5

## Conclusiones

En la última parte de este trabajo de fin de grado se analiza la consecución de los objetivos propuestos. los conocimientos aprendidos durante el grado que han sido aplicados, los conocimientos aprendidos por el alumno en la realización de este mismo así como los trabajos futuros en base a posibles ideas o funcionalidades a implementar en un futuro.

### 5.1. Consecución de objetivos

En el capítulo 2 se exponen una serie de objetivos:

1. Visibilidad de las experiencias de realidad virtual en la educación
2. Conseguir crear una experiencia VR propia
3. Implementar una funcionalidad del Micro:bit en el entorno de A-Frame

Para la consecución del primer objetivo se llevo a cabo una descripción de la realidad virtual así como las posibles utilidades y aplicaciones posibles para mejorar la educación en diferentes ámbitos ayudando así a la introducción de una parte práctica en el aprendizaje de los alumnos lo que conlleva una mayor retención de los conocimientos.

Respecto al objetivo de conseguir crear una experiencia de realidad virtual propia y a que a su vez sea posible para alumnos que vayan a acceder a la universidad, es conseguido gracias a BeetleBlocks y A-Frame dos herramientas simples de entender basadas en la programación por bloques y javascript respectivamente pero que ofrecen múltiples posibilidades de creatividad y en el caso de A-Frame la creación de experiencias virtuales a través de la docencia.

Por último la parte más compleja como es la programación del Micro:bit usando micropython para transformarlo en el controlador de la experiencia previamente creada con A-Frame. Como ya se ha dicho es la parte más compleja a la hora de ser impartida a alumnos en fase de acceso a la universidad ya que es necesario una serie de conocimientos de python, por tanto el tercer taller se ha orientado en un guión guiado que ayude ala comprensión con pequeñas partes prácticas para asentar las bases de la programación.

## **5.2. Aplicación de lo aprendido**

Para la realización de este trabajo de fin de grado las principales asignaturas relacionadas con el mismo son :

1. INGENIERIA DE SISTEMAS DE INFORMACION
2. DESARROLLO DE APLICACIONES TELEMATICAS
3. SERVICIOS Y APLICACIONES TELEMATICAS

En ellas el alumno aprendió la programación con python, javascript además de conocimientos y aplicación de utilidades relacionadas con HTML5.

## **5.3. Lecciones aprendidas**

Los conocimientos aprendidos por el alumno en este trabajo de fin de grado corresponden a la creación de objetos en 3D, la creación de experiencias de realidad virtual así como la programación del Micro Bit.

Con esto se puede concluir que toda la parte práctica de los talleres ha servido a su vez para que el alumno realizara una labor de estudio y aprendizaje de las tecnologías aplicadas, para la creación posterior de los talleres didácticos.

## 5.4. Trabajos futuros

En este apartado se detallan las posibles mejoras e ideas que podrían ser útiles en un futuro:

- Mejorar la animación del objeto 3D ya sea usando un Software externo o explotando las posibilidades de A-Frame.
- Aumentar el número de funcionalidades del Micro Bit dentro de A-Frame, usando eventos de movimiento como sacudirlo o el uso de la brújula interna.
- Introducir alguna funcionalidad extra de A-Frame en el segundo taller como por ejemplo la realidad Aumentada.
- Mejoras la experiencia programando un sencillo juego que haga uso del Micro Bit dentro de A-Frame, esto podría contemplarse como un trabajo mucho más elaborado si se quieren introducir más de un Micro Bit y aumentar la complejidad del juego en cuestión.



# Bibliografía

- [1] Computing education research blog. Aug. 2018.
- [2] U. W. David Weintrop. Comparing block-based and text-based programming in high school computer science classrooms. 2017.
- [3] D. M. donmccurdy. A-frame proxy-controls component, 2016.  
<https://aframe.io/docs/0.9.0/introduction/>.
- [4] E. R. Duks Koschitz, Bernat Ramagosa. Beetle blocks a new visual language for designers and makers. 2016.
- [5] P. S. Foundation. The python standard library, 2019.  
<https://docs.python.org/2/library/index.html>.
- [6] B. Romagosa. El zoo de la computacion. 2015.
- [7] A. Sweigart. Pyautogui documentation, 2014.  
<https://pyautogui.readthedocs.io/en/latest/index.html>  
<https://buildmedia.readthedocs.org/media/pdf/pyautogui/latest/pyautogui.pdf>.
- [8] B. D. Team. Blender 2.79 manual.  
<https://docs.blender.org/manual/en/latest/index.html>.
- [9] R. the Docs. P ucl tutorial: Bbc micro:bit micropython, 2016.  
<https://microbit-challenges.readthedocs.io/en/latest/index.html>.
- [10] D. W. whaleygeek. Bitio - a micro:bit io device, Apr. 2019.  
<https://github.com/whaleygeek/bitio>.
- [11] J. G. Yeste. Beetle blocks y un caso practico. 2016.



# Apéndice A

## Guía de uso Beetleblocks

BeetleBlocks tiene la mayoría de categorías en común con Scratch. Cualquier categoría, se puede ocultar pulsando el botón derecho. SI se quiere hacer un reinicio de un bloque se debe pulsar click derecho en el área de trabajo y seleccionar "show primitives". Por último todas las letras "**n**" de esta guía corresponden a un valor numérico y los puntos suspensivos (...) a un cuadro de entrada de texto

### A.1. Movimiento

- Go home
- move (n) # Se Mueve n veces.
- rotate z(eje) by (n)
- go to x:(n)y:(n)z:(n)
- set x(eje) to (n)
- change absolute x(eje) by (n)
- set z(eje) rotation to (n)
- point to x:(n)y:(n)z:(n)
- x(eje) position
- z(eje) rotation
- push position
- pop position
- set scale to (n)
- change scale by (n)
- scale

## A.2. Control

- Reset # Resets all rendered graphics.
- When Green Flag clicked # Runs the code below it when clicked.
- When space(↓) key pressed
- When I receive (↓)
- Broadcast (↓)
- [Checkbox] message
- Warp
- Wait (n) secs # Waits for a period of time.
- Wait until <boolean>
- Forever # Runs code inserted into it forever.
- Repeat (n) # Repeats code inserted into it a selected amount of times.
- Repeat until <boolean> # Repeats code inserted into it until a specified thing happens.
- If <boolean> # If something specified happens it will run the inserted code.
- If <boolean> else # If something specified happens it will run the code inserted into the if area but if it does not happen it will run the code inserted into the else area.
- Report [string]
- Stop all(↓) # Stops the specified ↓.
- Stop all but this script # Stops the script specified in the ↓
- Run (...)
- Launch (...)
- Call (input)
- Run (...) w/continuation
- Call (...) w/continuation
- Pause all # Pauses rendering.

## A.3. Colores

- Set hue(↓) to (n) # Sets selected option to selected n
- Change hue(↓) by (n) # Changes selected option by selected n
- Color (↓)



## A.4. Figuras

- Cube Dim. (n)
- Cuboid l: (n) w: (n) h:(n)
- Sphere Dia. (n)
- Tube l:(n) outer: (n) inner: (n)
- Text [string] H: (n) W: (n)
- 2D text [string] size: (n)
- Start drawing lines(↓) # Starts drawing the specified ↓ option.
- Stop drawing # Stops drawing.
- Start extruding curves(↓)
- Stop extruding # Stops extruding.
- Set extrusion Dia. to (n)
- Change extrusion Dia. by (n)

## A.5. Funciones

- Request user ... [text] # Requests ...ted text
- [ [Checkbox]] answer
- [Checkbox] mouse x
- [Checkbox] mouse y
- <mousedown>
- <key space(↓) pressed?>
- reset timer
- [Checkbox] timer
- [http://(url)]
- <turbo mode?>
- set turbo mode to <boolean input> # Sets turbo mode to inputted boolean (current date(↓))

## A.6. Crear una variable

Cuando se pulsa en el botón de crear una variable se despliega el cuadro de creación en el se introduce el nombre de la variable a crear y se puede elegir si esta variable puede ser usada por todos los sprites o solo para algunos en concreto.

## Variables

- set (↓) to (...) # Sets inputted variable to inputted text/n
- change (↓) by 1(n) # Changes inputted variable by inputted n
- show variable (↓) # Shows the inputted variable
- hide variable (↓) # Hides the inputted variable
- script variables a(...) <increase>
- list (...) <increase>
- (...) in front of list
- item 1(↓) of list
- all but first of list
- length of list
- list contains thing(...)
- add thing(...) to list # Adds inputted text/n to the list
- delete 1(↓) of list # Deletes inputted item from the list
- insert thing(...) at 1(↓) of list # Inserts inputted text/n in the inputted area of the list
- replace item (↓) of list with(...) # Replaces inputted item of list with inputted item

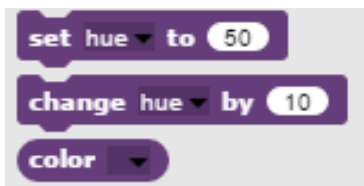
## A.7. Crear un bloque

La sección de crear un bloque consta de uno único hasta que el usuario crea alguno. Para crear un bloque, es necesario hacer click en el botón correspondiente, esto abre un cuadro de dialogo que ofrece las siguientes opciones:

1. Selección de categoría, el color de los bloques es acorde a la categoría elegida.
2. Editar el nombre del bloque creado, esté queda escrito en el propio bloque.
3. Selección el tipo de bloque entre tres: Command, Reporter y Predicate
4. Opción de hacer el bloque visible/disponible para el sprite actual o para el área de trabajo al completo.

## A.8. Operadores

- $(n) + (n)$
- $(n) - (n)$
- $(n) \times (n)$
- $(n) / (n)$
- $(n) \bmod (n)$
- $\text{round}(n)$
- $\text{sqrt}(\downarrow) \text{ of } (n)$
- pick random  $(n)$  to  $(n)$
- $(...) < (...)$
- $(...) = (...)$
- $(...) > (...)$
- $\langle \text{boolean} \rangle$  and  $\langle \text{boolean} \rangle$
- $\langle \text{boolean} \rangle$  or  $\langle \text{boolean} \rangle$
- not  $\langle \text{boolean} \rangle$
- $\langle \text{true} \rangle$
- $\langle \text{false} \rangle$
- join  $(\text{text})—(\text{text})—(\text{text}) < >$
- split  $(\text{text})$  by  $(\downarrow)$
- letter  $(n)$  of  $(\text{text})$
- length of  $(\text{text})$
- unicode of  $(\text{letter})$
- unicode  $(n)$  as a letter
- is  $(...)$  a  $(\downarrow)$  ?
- is  $(...)$  identical to  $(...)$  ?
- JavaScript function



(a) Colores

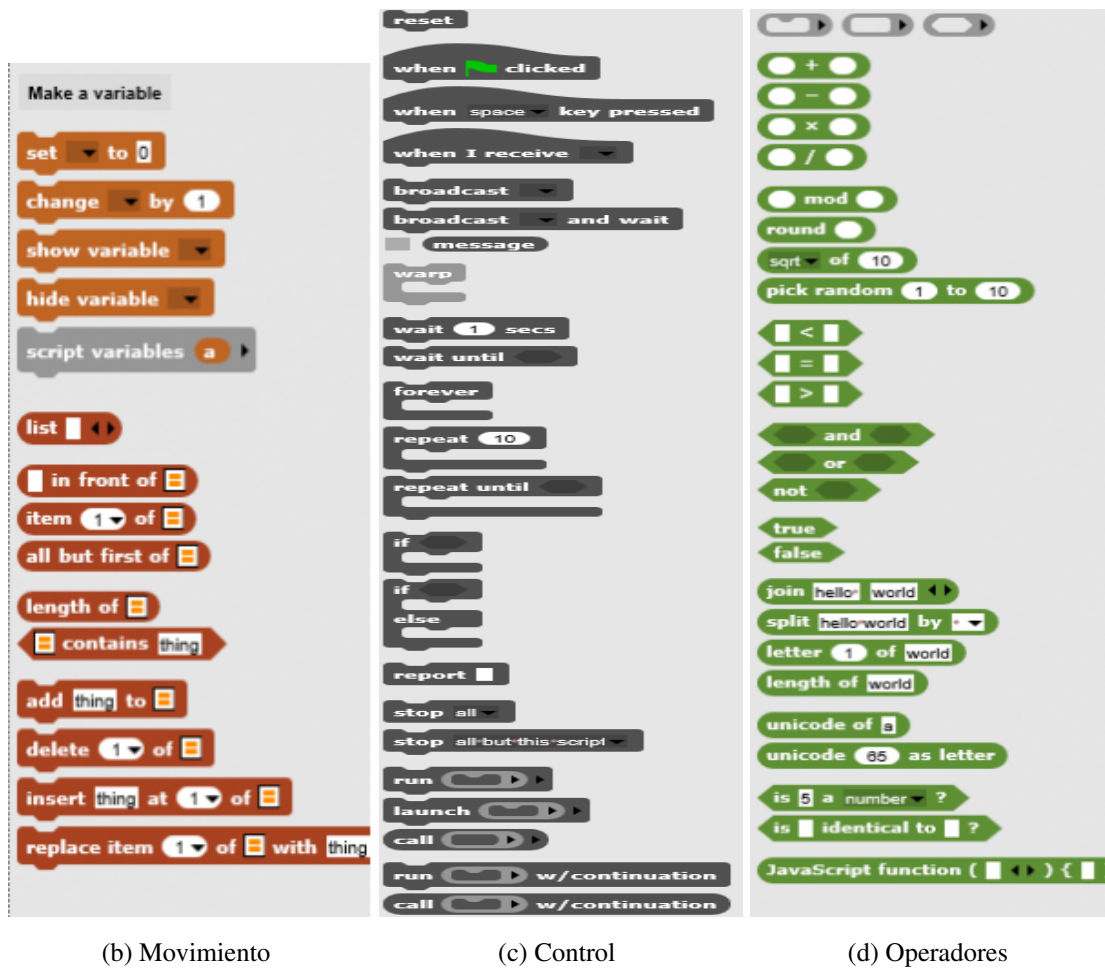


Figura A.1: Bloques Beetle Blocks A

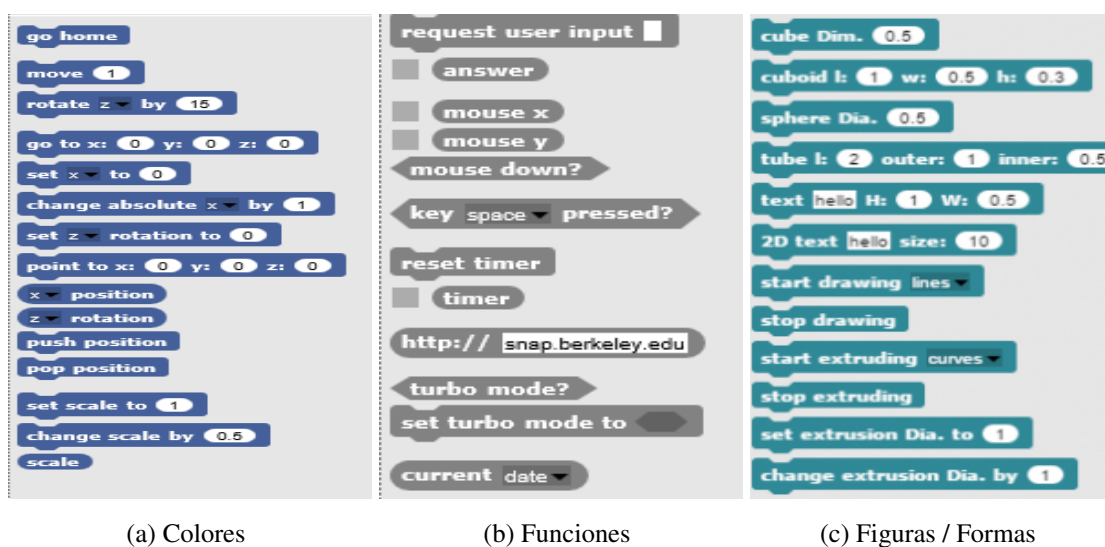


Figura A.2: Bloques Beetle Blocks B

## **Apéndice B**

### **Auriculares en Beetle Blocks**

En este segundo apéndice se explica el proceso de desarrollo y creación de los auriculares descritos en la práctica de BeetleBlocks.