



Grado en Ingeniería en Tecnologías de la Telecomunicación

Curso Académico 2018/2019

Trabajo Fin de Grado

**TALLERES DIDÁCTICOS ORIENTADOS A LA
DIFUSIÓN DEL DISEÑO 3D, EXPERIENCIAS VR
Y EL MICRO BIT 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 familia*

Agradecimientos

Agradezco el apoyo que he recibido por parte de mi familia para llegar a la finalización de este proyecto y así poner fin a esta etapa de mi vida.

AGRADECIMIENTOS

Resumen

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

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

- Beetle Blocks.
- A-Frame.
- Micro Bit.

El desarrollo de este proyecto está basado en la ejecución de 3 talleres enfocados en cada una de las mencionadas tecnologías. Los talleres a su vez forman un todo y son recursivos entre sí, de tal forma que todo lo aprendido en uno de ellos se usa en el siguiente.

En primera instancia se mostrará como crear un objeto 3D con la programación mediante bloques que ofrece Beetle Blocks, en el segundo taller se introduce A-frame y se mostrará como crear un entorno de realidad virtual en el que se inserte el objeto creado en el primer taller, por último en el tercer taller se explicará la programación y aplicación del Micro Bit como interfaz de usuario para poder moverse e interactuar dentro de la experiencia VR creada con A-Frame.

Para realizar el proyecto, primeramente se procedió a un estudio de cada una de las tecnologías y con la ayuda del tutor, se acordó que la mejor forma de introducir el conocimiento de las tres tecnologías era con una serie talleres prácticos que incidieran en cada una de las técnicas empleadas. Para la realización de los talleres, se ha escogido un formato de diapositivas en *PowerPoint* en las que se hace una introducción de las tecnologías a utilizar con una serie de ejemplos y ejercicios en los que los alumnos tienen que aplicar lo expuesto de una forma práctica.

RESUMEN

Un ejemplo de los ejercicios planteados puede ser la utilización obligatoria de una serie de bloques a la hora de crear una figura con Beetle Blocks o completar un programa de *python* dado para añadir una funcionalidad al Micro Bit.

En conclusión, al finalizar los talleres se consigue un hilo conductor entre ellos. El alumno comienza creando un objeto 3D con la programación mediante bloques de Bettleblocks, conoce y aprende la utilización de A-Frame y finalmente se le motiva a terminar la programación, en *python*, de una serie de funcionalidades básicas para el Micro Bit y poner su funcionamiento en práctica en un escenario de A-frame.

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. Micro Bit	3
1.3.1. Micro Bit como interfaz de usuario	5
1.4. Estructura de la memoria	5
2. Objetivos	9
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
3. Creación de experiencias VR	13
3.1. Tecnologías	13
3.2. Diseño 3D	13
3.2.1. Programación mediante bloques	14
3.2.2. Beetle Blocks	15

ÍNDICE GENERAL

3.2.3. Categorías de los bloques	16
3.2.4. Prácticas de aprendizaje en Beetle Blocks	17
3.2.5. Blender	21
3.3. A-frame	22
3.3.1. A-frame dentro de HTML	22
3.3.2. Uso de JavaScript dentro de A-Frame	24
3.4. BBC Micro Bit	26
3.4.1. Pyautogui	26
3.4.2. Bitio	27
3.4.3. Micropython	27
3.4.4. Programación del Micro Bit mediante bloques(Makecode)	28
4. Diseño y desarrollo de los talleres	31
4.1. Introducción y propósito de los talleres	31
4.2. Taller 1 (Creación de un objeto 3D con Beetle Blocks.)	32
4.2.1. Introducción a Bettleblocks	32
4.2.2. El Primer Objeto	34
4.2.3. Formato	37
4.2.4. Blender	37
4.3. Taller 2 (Creación de un entorno VR)	38
4.3.1. Creación de entorno VR con A-Frame	38
4.3.2. Explorando A-Frame	38
4.3.3. Animación básica con A-frame.	40
4.3.4. Llevar la experiencia VR a un smartphone	42
4.4. Taller 3(Micro Bit como IU en entorno VR)	44
4.4.1. Instalación de Bitio	45
4.4.2. Instalación de Pyautogui	47
4.4.3. Programación del Micro Bit	48
5. Conclusiones	53
5.1. Consecución de objetivos	53
5.2. Aplicación de lo aprendido	54

ÍNDICE GENERAL

5.3. Lecciones aprendidas	54
5.4. Trabajos futuros	55
Bibliografía	57
A. Guía de bloques de Beetle Blocks	59
A.1. Movimiento	59
A.2. Control	60
A.3. Colores	60
A.4. Figuras	61
A.5. Funciones	61
A.6. Crear una variable	61
A.7. Variables	62
A.8. Crear un bloque	62
A.9. Operadores	63
A.10. Imágenes de los Bloques	64
B. Programar el Micro Bit en Windows	67
C. Programas de Python	69
C.1. Testpy.py	69
C.2. interaction.py	69
C.3. move.py	70
C.4. camera.py	72
C.5. microIU.py	73
D. Talleres didácticos	77

ÍNDICE GENERAL

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 Beetle Blocks	18
3.8. Bloques e imagen de auriculares	19
3.9. Partes de los Auriculares	20
3.10. Blender	21
3.11. Exportar objeto	21
3.12. Editor Mu	27
3.13. Makecode	28
4.1. Crear perfil e iniciar Beetle Blocks	33
4.2. Inicio Beetle Blocks	34
4.3. Orientación y movimiento en Beetle Blocks	34
4.4. Bloque combinado	36
4.5. Bloques útiles	36
4.6. Sprite para crear figuras	36
4.7. inicio de Blender y exportación	37
4.8. Cubo y esfera primitives en A-Frame	39

ÍNDICE DE FIGURAS

4.9. Cursor en una escena de A-Frame	42
4.10. Ngrok	43
4.11. Funciones del Micro Bit	44
4.12. Resoluciones	50
4.13. Valores del acelerómetro	51
A.1. Colores	64
A.2. Bloques Beetle Blocks A	64
A.3. Bloques Beetle Blocks B	65
B.1. Cmd corriendo programa de micropython	68

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. A través de esta tecnología, el usuario puede sumergirse en imágenes 3D realistas generadas por ordenador, y al igual que los visores de realidad virtual, es una experiencia que luego puede enriquecerse con otros dispositivos. Se trata de una tecnología relativamente temprana, tanto en su concepción como en su aplicación, en los últimos años el crecimiento e implementación de esta tecnología ha ido en aumento. Hasta los años 90 no se consiguió simplificar los grandes simuladores de realidad virtual en dispositivos portátiles, que pueden tratarse desde cascos de realidad virtual, a gafas de plástico o cartón, con las que los *smartphones* se encargan de hacer realidad la experiencia.

Entre los años 2015 y 2016, se vivió el gran avance en el desarrollo de los dispositivos VR por parte de grandes marcas como HTC o Sony, que introdujeron en el mercado diferentes propuestas al alcance del público.

A la hora de hablar de la realidad virtual en este trabajo, se ha profundizado en las denominadas experiencias, la cuales consisten en entornos de realidad virtual que permiten al usuario transportarse al lugar de la escena, ya sea como espectadores o interactuando con ellas a través de sensores o controladores. Actualmente, el mayor uso de la realidad virtual está orientado a videojuegos, pero también se está consolidando en áreas relacionadas con la investigación médica o la educación, siendo ésta última uno de los motivos de la realización de este trabajo, ofreciendo ejemplos de su aplicación en el aprendizaje de diferentes técnicas y sistemas.

1.1.1. Utilidad VR en la educación

La introducción y utilización de la realidad virtual en la educación, es algo que está empezando a introducirse cada vez en mayor medida en la sociedad. Representa un mundo de posibilidades en el que hay sitio para todo tipo de ideas e invenciones, que pueden adaptarse a distintos tópicos durante la etapa educativa.

Durante años ha estado limitada a la formación de pilotos aéreos en simuladores con un alto coste, 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, aunque los visores de realidad virtual ofrecen la mejor experiencia, con la combinación de unas gafas de cartón como las *Cardboard* de Google y un móvil se puede ser capaz de experimentar miles de experiencias en realidad virtual. Tampoco se puede obviar su inmersión, 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 mostrado.

Declaraciones como las de Baptiste Grève, creador de la plataforma de experiencias virtuales *Unimersiv*¹, fundamentan lo positivo de esta tecnología, en las que se asegura 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 múltiples herramientas de creación y diseño 3D tanto de pago como gratuitas. Este trabajo de fin grado va a mostrar el uso de dos de ellas.

La primera y en la que más se profundiza es Beetle Blocks², una plataforma Open Source que propone un método de diseño basado en la programación por bloques, siendo más sencillo, 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

¹<https://unimersiv.com/>

²<http://BeetleBlocks.com/>

en este trabajo. La segunda, Blender,³ un software Open Source de creación 3D más clásico y complejo, cuyo uso se limitará a proporcionar ligeras modificaciones a las figuras creadas previamente con Beetle Blocks.

Más adelante en este trabajo, se describe la manera de crear objetos con Beetle Blocks en el capítulo 3.2, se ofrecen ejemplos para la creación de figuras geométricas básicas, y se proporcionan las bases para llegar a crear formas tan complejas como uno quiera debido a la versatilidad de la plataforma.

El principal objetivo de este trabajo es la creación de talleres didácticos para gente sin experiencia en el diseño 3D, por este motivo y sirviendo como apoyo este par de artículos [11] [5], se ha elegido Beetle Blocks para mostrar el diseño 3D de una forma amena y menos compleja que si se usara una herramienta más potente.

1.2.1. Entornos VR con figuras en 3D

Los entornos de realidad virtual suelen estar formados por la combinación de figuras en 3 dimensiones. Al igual que para el diseño 3D, existen múltiples programas de software que nos permiten la creación de estos entornos.

Tal y como está planteado este trabajo no se ha buscado una experiencia realista, sino una centrada en el aprendizaje de la creación de una experiencia básica que permita familiarizarse con alguno de sus principales elementos de creación. Teniendo esto en cuenta, se ha optado por A-Frame, una utilidad de creación de experiencias y entornos VR Open Source, A-frame supone el núcleo del segundo taller didáctico y una vez se ha creado el entorno de realidad virtual permite la exportación de figuras 3D, en este caso y a modo de hilo conductor con el primer taller se mostrará como insertar una figura creada con Beetle Blocks. La creación del entorno de realidad virtual con A-Frame se llevará a cabo en el capítulo 3.3

1.3. Micro Bit

Como se puede leer en su página oficial⁴, Micro Bit es una pequeña computadora programable, diseñada para hacer que el aprendizaje y la enseñanza sean fáciles y divertidos.

³<https://www.blender.org/>

⁴<https://microbit.org/es>

De una forma más técnica se trata de un sistema de Hardware embebido diseñado por la BBC, para la enseñanza informática en el Reino Unido. 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 Blueethoot y USB, 25 leds, 2 botones programables y puede ser alimentado 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 estos componentes puede verse en la Figura 1.1.

Gracias a la gran cantidad de sensores que incorpora, sólo con la placa 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. 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 facilita que una vez finalizados cada uno de los talleres, todo aquel alumno interesado pueda continuar experimentando con las herramientas y tecnologías mostradas, por ejemplo, en su propia casa.



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 aplicación en este trabajo de fin grado va a ser darle una funcionalidad de interfaz de usuario, es decir, será el controlador de la experiencia de realidad virtual creada con A-Frame.

A la hora de comenzar a programar esta funcionalidad para el Micro Bit, al alumno se le proporciona un programa de ejemplo de movimiento físico en un entorno de A-frame. Tras analizar las posibilidades que ofrece la programación del Micro Bit, se consigue implementar una serie de funcionalidades extras.

Al finalizar el último de los talleres, las funcionalidades implementadas en el Micro Bit permitirán un movimiento tanto físico como de la cámara dentro de la experiencia, basados en la combinación de los valores que ofrece el acelerómetro integrado, también se añade una funcionalidad de interacción con el objeto de Beetle Blocks insertado en el escenario de A-frame.

Para la implementación de estas funcionalidades, se desarrolla un programa en micropython, una variación de Python adaptada al Micro Bit, que permite ejecutar programas de Python en el dispositivo. Para ello, será necesaria la importación de dos módulos, este proceso se describe en más detalle en la sección 3.4.4.

Como forma de apoyo en el aprendizaje de micropython y para el uso y metodología de diversas funciones implementadas en Python, se ha hecho uso de las siguientes documentaciones [9] [6].

1.4. Estructura de la memoria

En el capítulo de introducción de este trabajo se ofrece una visión general de la realidad virtual, así como, la descripción y significado de las experiencias VR, y además se muestra su emergente introducción en la educación como método didáctico y de gran valor práctico.

A continuación, se describen las herramientas de diseño 3D Beetle Blocks y Blender en la sección 1.2 y se fundamenta la elección de la programación de objetos en 3D mediante bloques, que resulta perfecto para introducirse en la creación y diseño de estas figuras.

La última tecnología a introducida es el Micro Bit, en esta sección también se explica la razón de incluirlo en una experiencia de A-Frame, para darle un uso como interfaz de usuario y así complementar la experiencia de realidad virtual.

En el Capítulo 2, se describen los objetivos que se presentan en este trabajo de fin de grado, en el que también se habla de la motivación para la elección de esta temática, así 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 una breve explicación del uso o práctica de cada una de ellas para ayudar a comprender su funcionamiento.

Este capítulo se puede dividir en tres partes diferenciadas.

- Sección 3.2.

Su contenido corresponde al diseño de objetos en 3D con la descripción de las dos tecnologías usadas Beetle Blocks 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 interfaz de usuario.

En el capítulo 4 se encuentra el proceso de desarrollo y creación de las tres talleres o prácticas que suponen el núcleo práctico de este trabajo. El contenido y tecnología implementada de cada uno puede verse en la Tabla 1.1 y es el siguiente:

- Primer taller, sección 4.2, el objetivo de este taller es la creación de una figura 3D mediante Beetle Blocks 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 proporciona A-frame.

- Por último un tercer taller, sección 4.4, centrado en el proceso de programación del Micro Bit, y su implementación en A-Frame como controlador de la experiencia.

	TALLER 1	TALLER 2	TALLER 3
Tecnología	Beetle Blocks	A-Frame	Micro Bit
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, donde se analizan los resultados obtenidos y se expone la consecución de los objetivos propuestos previamente. Además se pone de manifiesto, la aplicación de los conocimientos obtenidos por el alumno durante la realización de este grado, los cuales son fundamentales para la consecución de este trabajo de fin grado. Los dos últimos puntos de este capítulo corresponden a los conocimientos útiles aprendidos en consecuencia de este trabajo, así como los posibles avances y mejoras aplicables a trabajos futuros.

Las últimas secciones de este trabajo corresponden a la Bibliografía y a los Apéndices A, B, C y D.

- En el primero de ellos se describe de una forma más detallada la función y descripción de todos los bloques de Beetle Blocks.
- En el segundo de ellos se exponen los pasos necesarios para programar el Micro Bit en Windows.
- El tercero por su parte esta formado por la serie de diapositivas que forman los tres talleres, además de los programas que conforman la parte práctica del taller del Micro Bit.
- En el cuarto y último se presentan las diapositivas que conforman los talleres didácticos.

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, se describe 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 de los objetos que proporciona Beetle Blocks, sirviendo como apoyo este artículo [1] que corrobora su efectividad como método introductorio a la hora de realizar un primer acercamiento a este

campo.

Para introducir la creación de un entorno de realidad virtual la utilidad elegida es A-Frame, creado con el fin de permitir a los desarrolladores web y diseñadores de 3D y VR experiencias con HTML sin tener que conocer WebGL.

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 el 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 se interesa 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 una gran fuente de interés desde un primer momento para el alumno, lo que ha generado una gran motivación para la realización de este trabajo y ha ayudado al aprendizaje de este área. Además se ha otorgado utilidad a los conocimientos en forma de talleres para que más gente pueda aprovechar la investigación llevada a cabo.

Otra de los impulsos principales al realizar este trabajo está relacionada con los entornos de desarrollo de las 3 tecnologías principales, Beetle Blocks, A-Frame y Micro Bit, los cuales 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 del 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 grado, está orientada a 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, una realización de pequeños tutoriales y una 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.
- Adecuar los conocimientos a los lectores, en este caso, alumnos de enseñanzas medias, buscando la concordancia a sus conocimientos.
- Definir el formato del taller, siendo un taller en el que los alumnos harán experiencias guiadas pero en las que participarán activamente para mantener el vínculo 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

Para la realización de los talleres, se debe tener un conocimiento profundo de todo el software que se usa en ellos. Por ello, se ha realizado un plan de trabajo en el que primeramente se estudian las 2 tecnologías Beetle Blocks y A-Frame.

Gracias a la ayuda del tutor mediante tutorías a través de Hangouts 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, sin embargo este primer planteamiento fue descartado 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 Beetle Blocks, 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 tener una experiencia completa de realidad virtual incorporando el Micro Bit como un controlador y una figura de Beetle Blocks al escenario de A-frame al completar los 3 talleres.

El formato de los talleres elegido es una presentación con PowerPoint. Cada taller, cuenta con una serie de diapositivas en las que se introduce la tecnología, se establecen unos pasos previos y una serie de ejemplos prácticos para ayudar a la comprensión y uso durante la práctica. Además dentro de la presentación, se intercalan una serie de ejercicios en los que se requiere utilizar lo aprendido, para que los alumnos pongan en práctica por ellos mismos las diferentes herramientas. Un ejemplo de lo anteriormente expuesto, se puede ver en el primer taller cuando se pide diseñar un objeto en Beetle Blocks con el requerimiento obligatorio de usar una serie de bloques dados.

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.

Beetle Blocks 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 Beetle Blocks, 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 la evolución si no se tiene experiencia de ningún tipo con la programación. En este trabajo de fin de grado aparece tanto en Beetle Blocks 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 está 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.

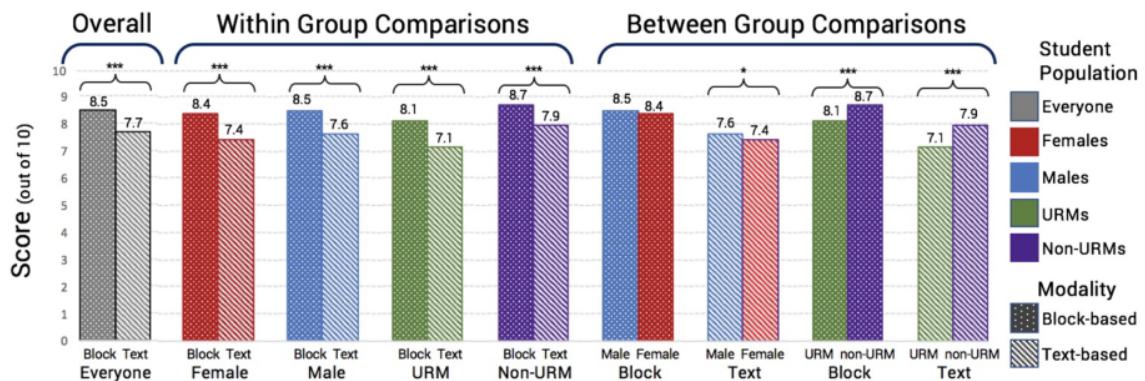


Figura 3.1: Puntuacion media de la evaluación de estudiantes a preguntas de programación

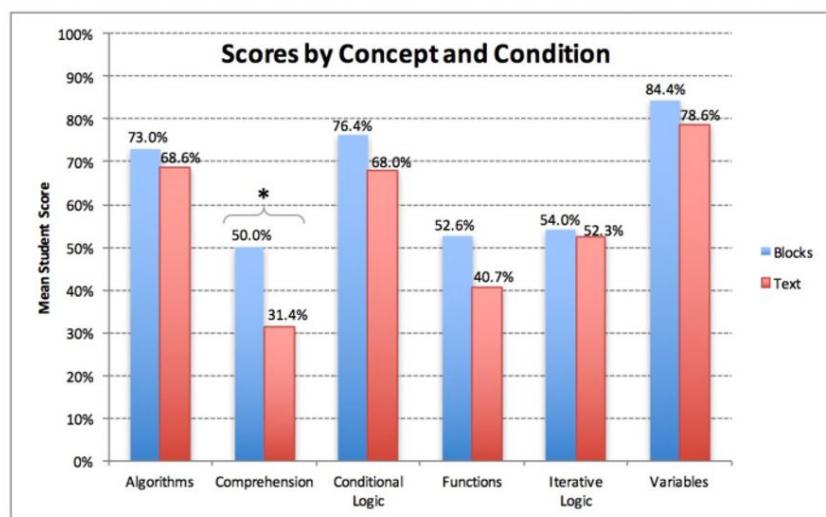


Figura 3.2: Tasa de rendimiento medio para estudiantes en diferentes conceptos

3.2.2. Beetle Blocks

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 [?], Beetle Blocks es más que una herramienta: es también una puerta de entrada muy atractiva al mundo de la programación. El editor de Beetle Blocks está basado en Scratch e implementado usando Snap! y ThreeJS.

Para empezar a usar la herramienta basta con acceder a la web ¹ y hacer click sobre “Run Beetle Blocks”. La pantalla mostrada al arrancar Beetle Blocks 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.

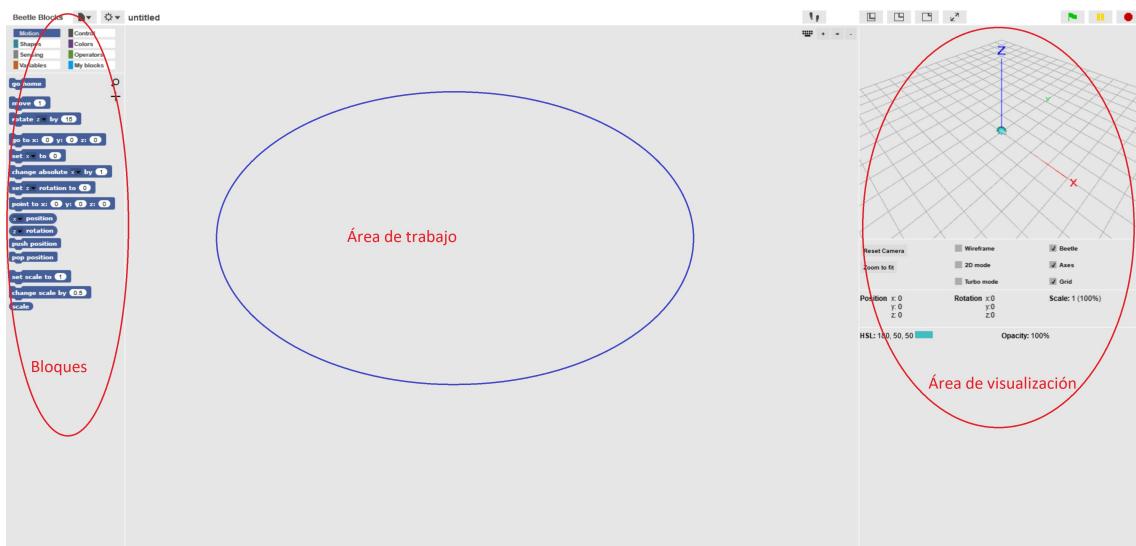


Figura 3.3: Inicio Beetle Blocks

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

¹<http://Beetle Blocks.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 Beetle Blocks 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 moverlos 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.

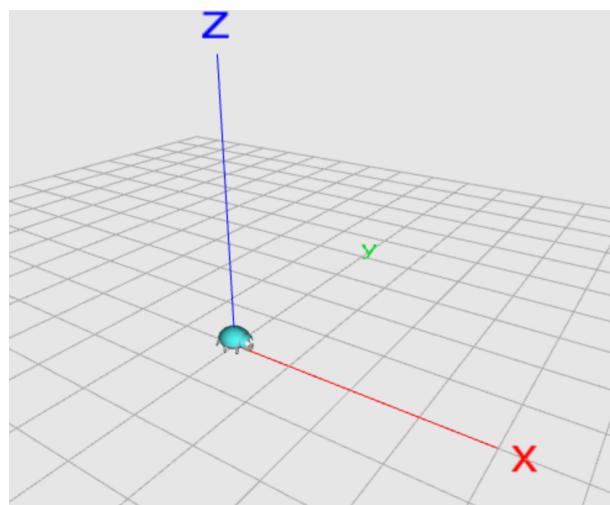


Figura 3.4: Escarabajo del área de visualización 3D

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 Beetle Blocks

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. Por último, se expone el proceso de creación de un objeto más complejo con el objetivo de justificar los conocimientos aprendidos, y que además sirva como figura práctica para la realización del segundo taller.

- **Movimiento por los ejes**

Dependiendo de como sea la orientación de las figuras dibujadas es necesario moverse por los ejes y orientar el escarabajo como puntero de orientación para dibujar. El bloque que se usa para moverse y situarse donde se desea 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 ($18^\circ * 20 = 360^\circ$).

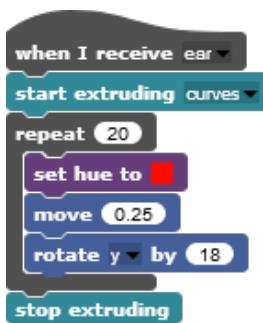


Figura 3.6: Bloque para crear círculos

■ 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 creaciones más complejas combinándolas entre ellas.



Figura 3.7: Bloques de las figuras predeterminadas de Beetle Blocks

■ Auriculares

Como parte final en el aprendizaje de Beetle Blocks, se ha creado una figura más compleja para corroborar lo aprendido. La figura en cuestión son unos auriculares, el objeto y bloques que lo componen se pueden ver en la Figura 4.3

Desde un primer momento, se ha pensado que esta figura sirva como práctica para el taller, por lo tanto, su complejidad tampoco debía ser muy elevada. El requisito era que usara los bloques que son considerados más importantes para aprender el manejo y uso de Beetle Blocks.

Con estos bloques se pretende que los alumnos además de aprender a diseñar objetos 3D en Beetle Blocks, aprendan algunos fundamentos de la programación tradicional, como pueden ser el uso de bucles, crear funciones o la inicialización de del escenario.

- Posicionamiento, fundamental para mover el escarabajo a un punto exacto.
- Movimiento, para aprender a moverse en los 3 ejes.
- Dibujo de líneas/curvas, son los bloques de creación.
- Bucle, recurso básico en programación.
- Conjunto de bloques “función”, con el objetivo de mostrar como organizar los bloques y controlar la ejecución.

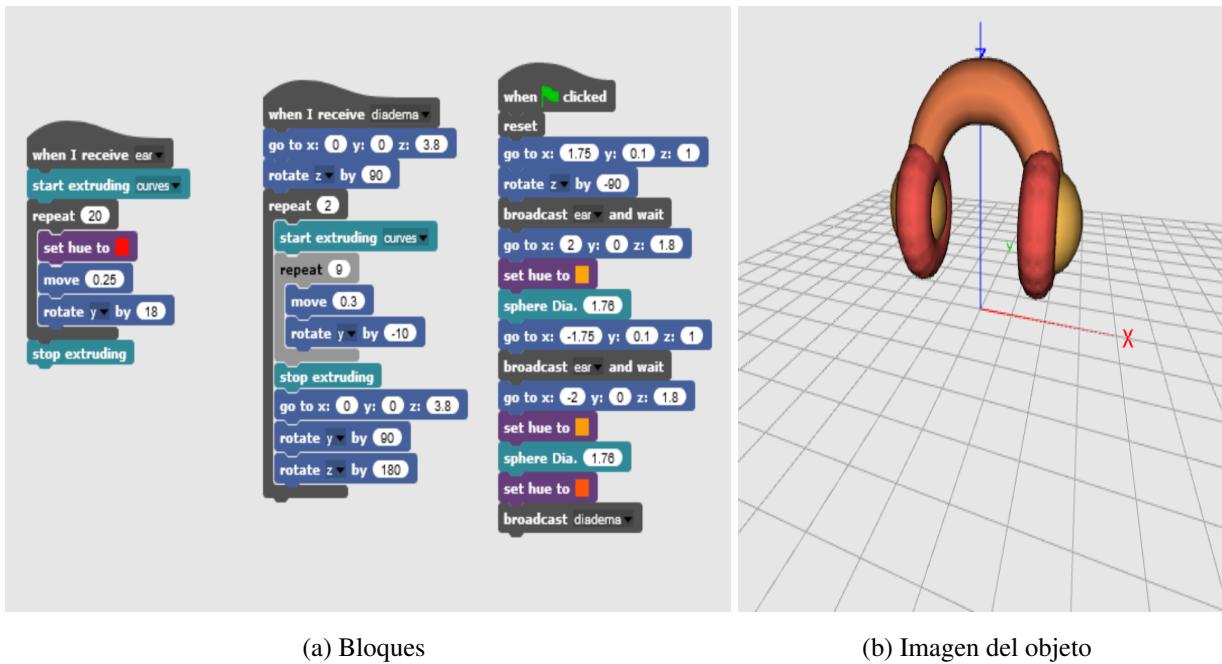


Figura 3.8: Bloques e imagen de auriculares

A continuación y apoyándose en la Figura 3.8a se analiza la composición de los bloques de los auriculares y su función. Tanto el bloque central, como el de la izquierda representan 2 funciones, uno para crear la diadema y otro para crear una orejera, más adelante se explica su función al detalle.

El bloque de la derecha por tanto representa lo que vendría a ser el "main" si se mirara como un fichero en cualquier lenguaje de programación. Los primeros dos bloques, pueden ser llamados bloques de control", con el primero de ellos se restringe la ejecución de los bloques anidados hasta que se pulse sobre ellos, el segundo sirve para hacer un reset al plano y dejarlo vacío. Tras unos bloques de posicionamiento y rotación, se invoca por primera vez el conjunto de bloques encargado de dibujar una de las orejeras, su metodología consiste en crear una circunferencia, en rojo en el dibujo. Una vez terminado se vuelve a posicionar el escarabajo, en este caso en el centro del círculo creado por la función *ear* y ayudándose del bloque que dibuja una esfera se rellena el interior formando así una de las orejeras, en amarillo en la Figura 3.9a.

El siguiente paso es replicar otra orejera paralela e invertida a la actualmente creada, por tanto se mueve el escarabajo a la misma posición pero, en este caso, a la parte negativa del eje X se vuelve a invocar la función *ear* y posteriormente se dibuja la esfera, se puede ver el

resultado en la Figura 3.9a. Por último solo queda crear la diadema, en naranja en la Figura 3.8b, que viene a ser una curva que une ambas, en el último bloque del “main” se invoca la función central, que es el encargado de dibujarla.

En primera instancia, este conjunto de bloques, posiciona el escarabajo entre las dos orejeras, sin embargo lo sitúa en una posición superior en la parte positiva del **eje Z**. Se pueden observar dos bucles, el bucle interior dibuja media curva en el eje positivo del eje x que une el centro con la orejera derecha, Figura 3.9b, el bucle exterior repite este proceso en la parte negativa del eje x, dando finalmente el resultado observado en la Figura 3.8b.

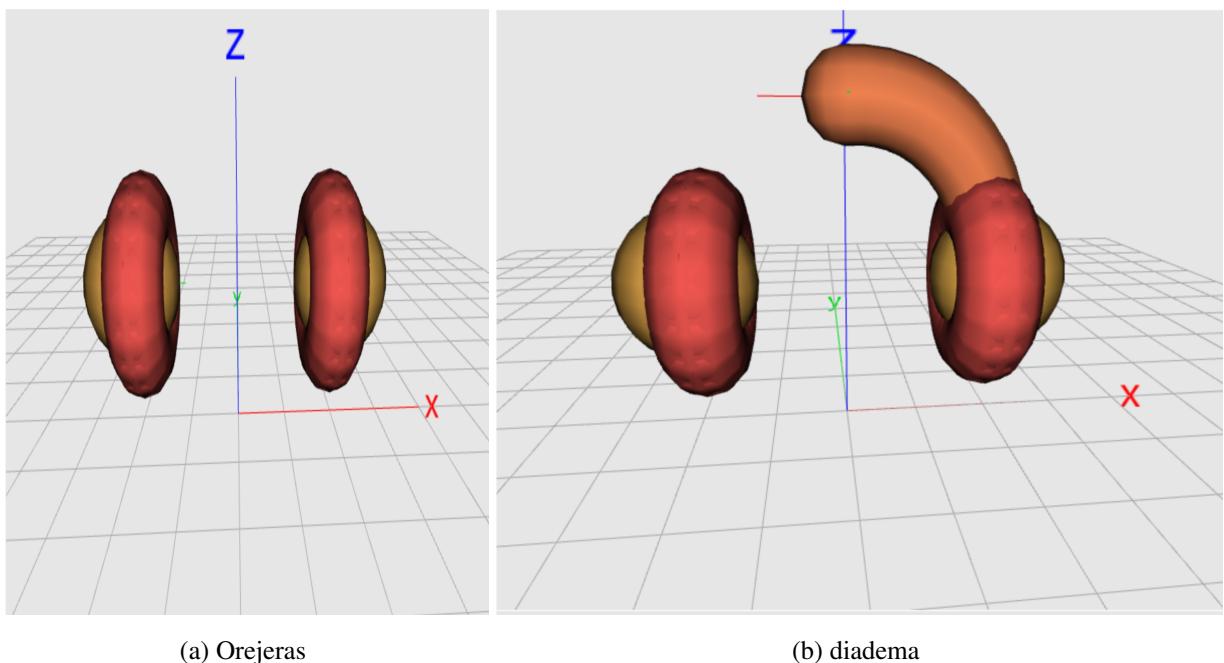


Figura 3.9: Partes de los Auriculares

3.2.5. Blender

Blender es una potente herramienta OpenSource 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 Beetle Blocks 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 3.10

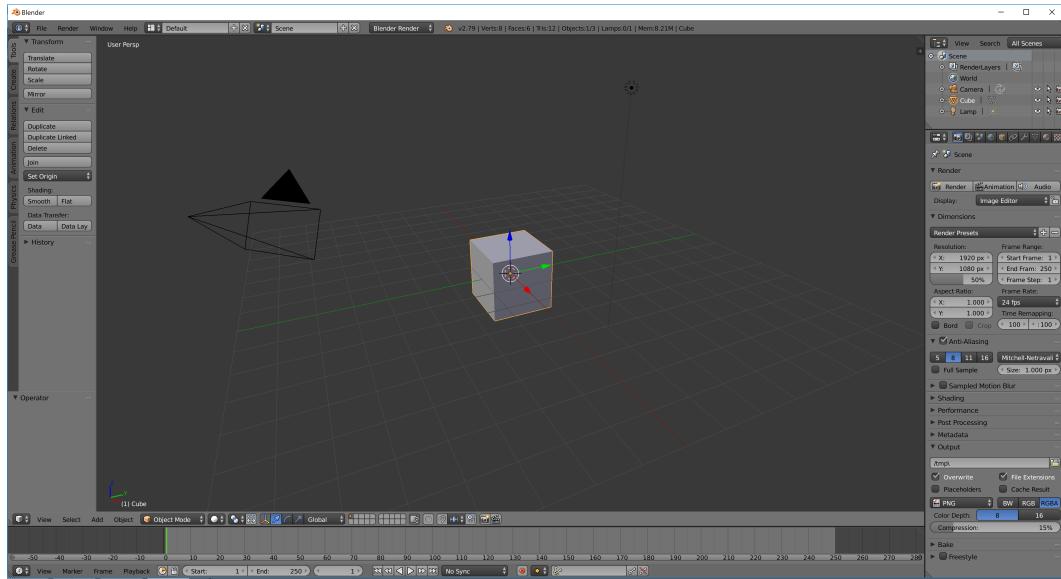


Figura 3.10: Blender

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*.

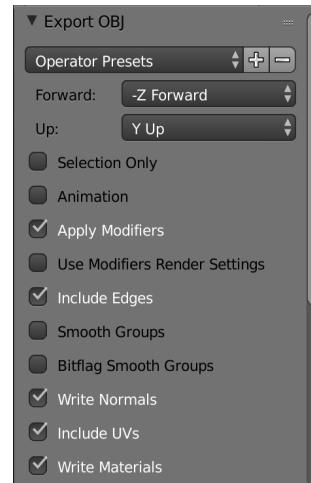


Figura 3.11: Exportar objeto

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 haciéndolo simple de usar en primera instancia, aunque el núcleo es un potente framework ECS² 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 360º, 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>
```

²Entity Component System

De entre los principales conceptos y elementos que introduce A-Frame se van a destacar cuatro de ello las escenas³, las entidades⁴ y los componentes⁵ y los primitivos⁶.

- *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 las 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 como 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>
```

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

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

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

⁶<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 primitives 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 desarrollados además pueden crear sus propios primitivos. Unos ejemplos de los primitivos incluidos 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` [4], se trata de una serie de Add-ons y helpers para A-Frame, e incluye componentes de control, modelos predefinidos, primitives

Otro recurso muy utilizado para A-Frame y basado en JavaScript es la importación de components⁷, para habilitar o añadir una serie de funciones en el entorno creado con A-frame. Existe una amplia biblioteca de components para importar, entre ellos hay algunos que se organizan en paquetes o módulos como superframe.⁸

Aunque el uso de cada component es particular y puede encontrarse en su documentación [3], generalmente para usar un component hay que vincularlo con una entidad `<a-entity>` y añadirle una serie de elementos para su configuración.

A continuación, se muestra un ejemplo con el component *particle system*, en él se dan los pasos necesarios para su importación y posterior uso ligado a una entidad:

- 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>
```

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

⁸<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 Micro Bit 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 ⁹, que se encuentra en el repositorio de GitHub 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 ¹⁰.

⁹github.com/sarehp/Micro Bit-aframe

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

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, permite correr código de Python en el ordenador mientras se interactúa 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]. Está basado en Python [6] 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 3.12. 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.

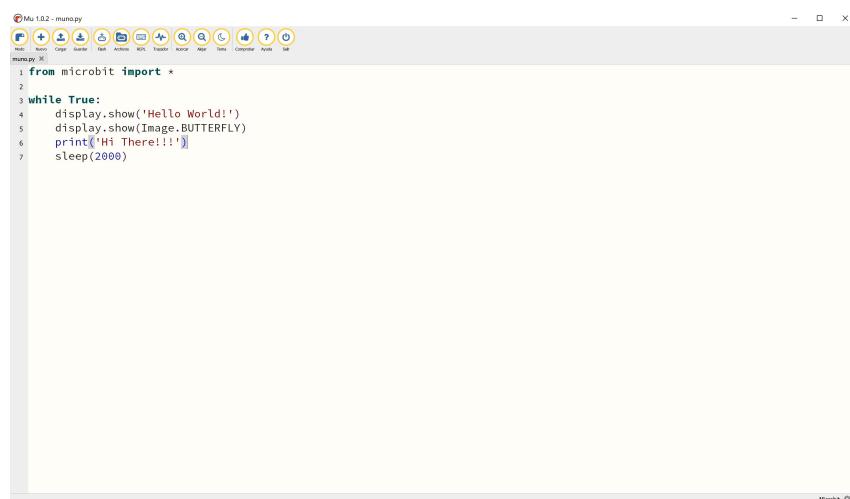


Figura 3.12: Editor Mu

Si no se cuenta con el editor de Mu la forma de cargar los programas en el Micro Bit 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)

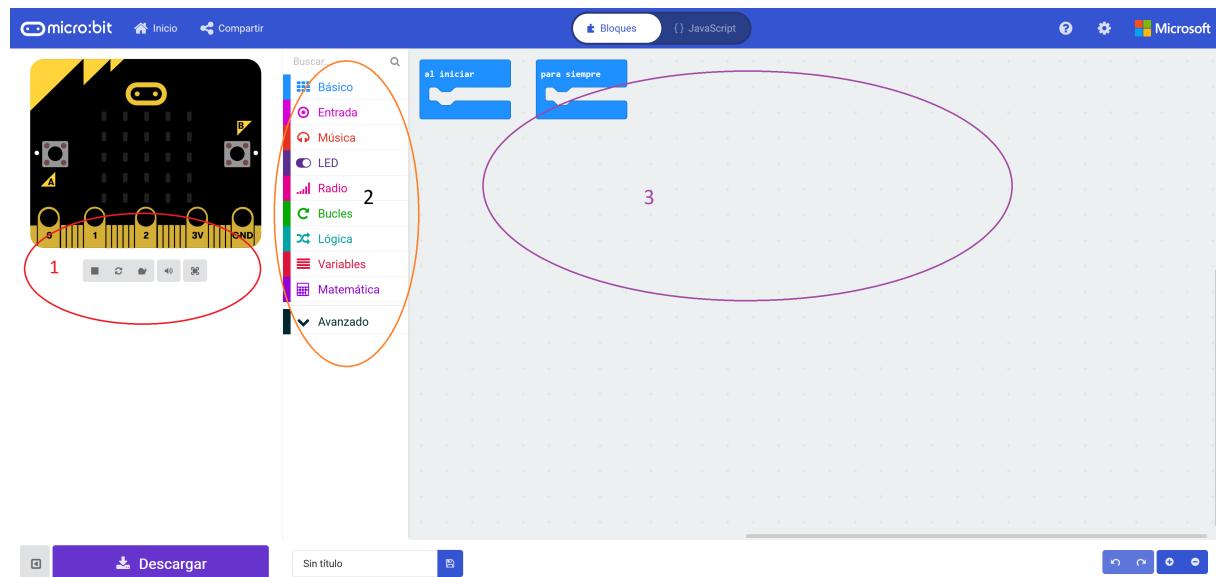


Figura 3.13: Makecode

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 Beetle Blocks divide sus bloques en grupos, que aportan distintas funciones, acciones para los LEDs, Lógica, Matemática ... etc

En la web de Makecode¹¹ 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 3.13 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).

¹¹<https://makecode.Micro Bit.org/>

Como complemento se puede hacer una previsualización del resultado de la ejecución de los bloques creados(1).

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 autoconclusivos, pero además a modo de hilo conductor, cada uno de ellos utiliza lo aprendido en el anterior. Por tanto 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 Beetle Blocks y que puede ser controlada con un Micro Bit para moverse por ella, y de tal forma el alumno aprende acerca de las tres tecnologías implementándolas en un proyecto global.

El desarrollo y contenido de los talleres es el siguiente:

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

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 Beetle Blocks.

- **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 Beetle Blocks.)

Este primer taller pretende enseñar a crear un objeto en 3D con Beetle Blocks 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 Beetle Blocks¹ 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 Beetle Blocks, para ello se debe pulsar en *Run Beetle Blocks* como se ve en la Figura 4.1.

¹<http://BeetleBlocks.com>



Figura 4.1: Crear perfil e iniciar Beetle Blocks

4.2.2. El Primer Objeto

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

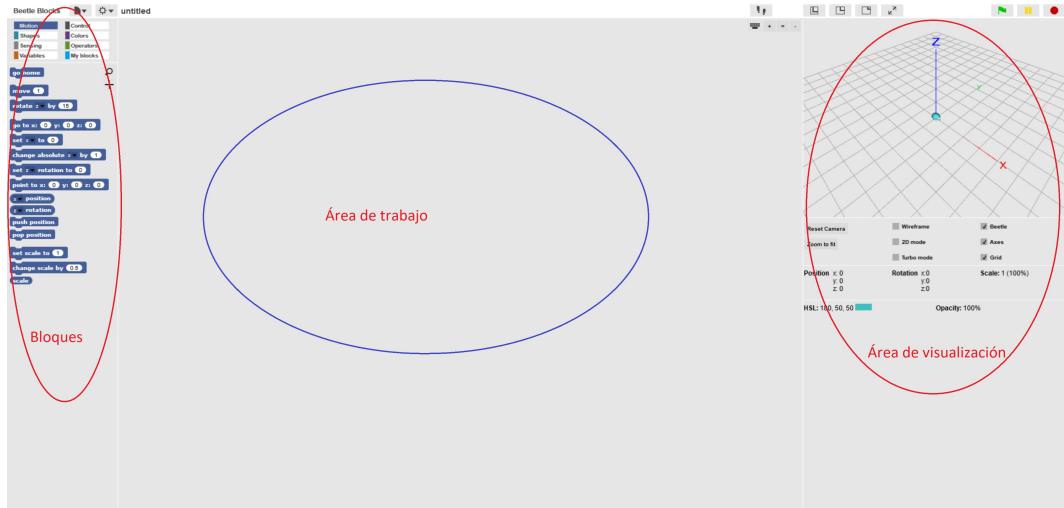
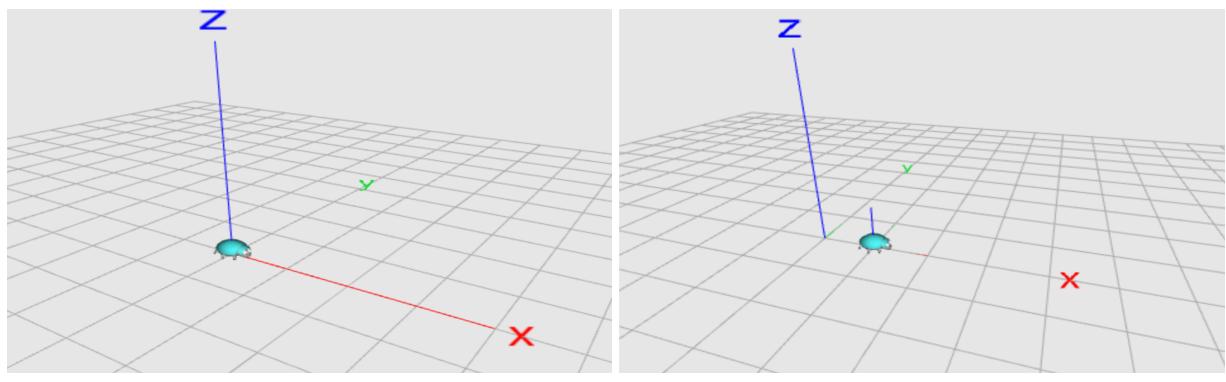


Figura 4.2: Inicio Beetle Blocks

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 previvulización 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.



(a) Posición inicial

(b) Movimiento en el eje x

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 órdenes y funciones útiles para familiarizarse con Beetle Blocks 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 Beetle Blocks 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 la ejecución de los bloques que son insertados posteriormente se repiten un número de veces igual al 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 escara-bajo, 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 Beetle Blocks, 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 Beeetleblocks, 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 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*.

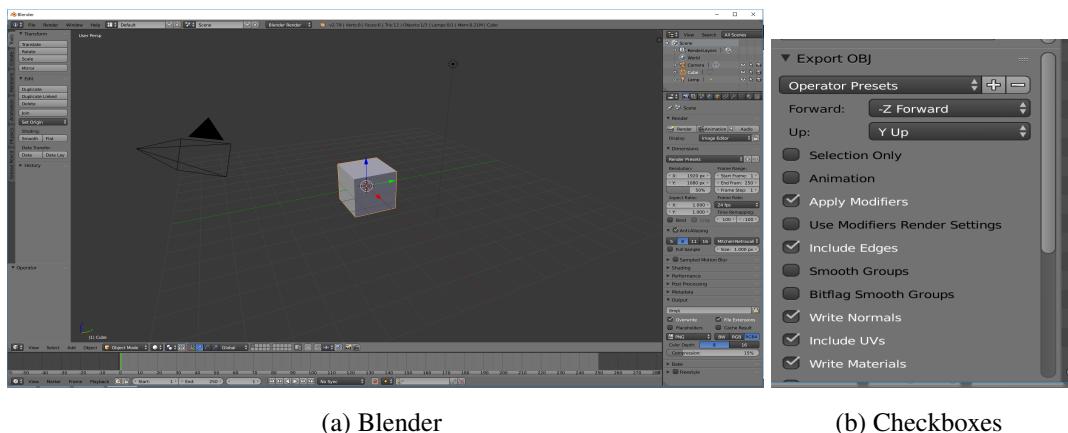


Figura 4.7: inicio de Blender y exportación

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 Beetle Blocks 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 el taller están basados 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 primitive, 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. Para usarlos basta con nombrarlos en el *index.html* como si se tratara de una entidad, a los primitives 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 HMTL como *<a-scene>*.

A continuación se expone un ejemplo en el que se introducen una serie de primitives, entre ellos un cubo y una esfera, dentro de la escena de A-Frame, 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>
```

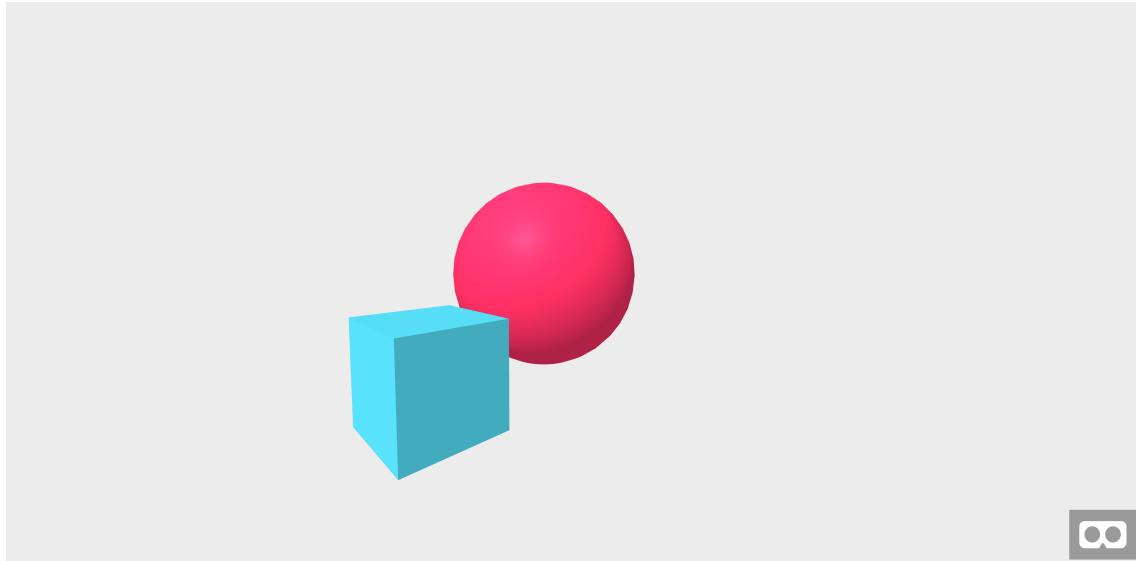


Figura 4.8: Cubo y esfera primitives en A-Frame

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

Una vez explicada la metodología para cargar objetos en A-Frame, el siguiente paso a realizar es cargar el objeto creado con Beetle Blocks, 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** en la sección de assets. 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-assets>
  <a-asset-item id="Beetle-obj" src="models/cascos3.obj"></a-asset-item>
  <a-asset-item id="Beetle-mtl" src="models/cascos3.mtl"></a-asset-item>
  
</a-assets>
```

2. Se cargan el **.obj**, **.mtl**, y la **textura**, además se le pueden pasar diferentes propiedades como su posición:

```
<a-entity
  obj-model="obj: #Beetle-obj;" mtl="#Beetle-mtl"
  material="src: #Beetle-texture" position="1.5 1 2.5" >
</a-entity>.
```

4.3.3. Animación básica con A-frame.

A-Frame permite añadir algo de animación a los objetos que integran la escena, mediante el identificador **<a-animation>**, se tiene acceso a una serie de animaciones básicas, pero en este caso se ha optado por la importación de archivos JavaScript, que funcionan como módulos de A-Frame.

El ejemplo expuesto aquí consigue aportarle una animación al objeto, se produce una rotación al hacer click sobre él, para hacerlo se debe importar el paquete aframe-animation-component².

De la misma manera que se hace con A-Frame se importa en la cabecera **<head>** del documento HTML.

²<https://github.com/aframevr/aframe/blob/master/docs/components/animation.md>

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

La animación forma parte de las propiedades del objeto *<animation>* se suele indicar una vez se establezcan las propiedades básicas del objeto, representado en A-Frame como *<a-entity>*, este módulo también puede añadir animación a los primitives de A-Frame, a continuación se expone un ejemplo en el que se añade una animación de rotación:

```
<a-entity obj-model="obj:#Beetle-obj" mtl="src:#Beetle-mtl"
material="src:#Beetle-texture" position="1.5 1 2.5"

animation="property: rotation; loop: 5; to: 0 360 0;
startEvents: click"
</a-entity>
```

Como se puede observar, a la hora de declarar una animación primero se debe indicar el tipo, en este caso rotación y posteriormente, se establecen una serie de modificadores, para este ejemplo el número de veces que se quiere repetir y la dirección de la rotación en alguno de sus ejes, la lista completa de argumentos que funcionan como modificadores se puede consultar en la documentación anteriormente citada.

El último de los argumentos, no forma parte de la animación propiamente dicha, se trata de un identificador de evento, en este caso el click del ratón, que permite que la animación sólo se ejecute cuando se hace un click sobre el objeto.

Cuando se está inmerso en un entorno de realidad virtual, no se dispone del puntero del ratón para hacer click sobre el objeto. El puntero sería el equivalente al centro de la mirada, para conseguir simular que se está mirando al objeto sin necesidad de tener un visor puesto, es necesario crear un cursor “virtual”, que además resulta de gran ayuda cuando se utilice un visor, ya que ayuda a una orientación más rápida a la hora de moverse por un entorno de realidad virtual.

El cursor se crea una vez se define la posición inicial de la cámara, en este caso se le da un color rojo de forma que resalte con el fondo de la escena. Una vez aclarado este punto, la creación del cursor resulta muy sencilla y el resultado final puede verse en la Figura 4.9:

```
[ . . . ]
<a-camera position="1.5 1 6.5">
  <a-cursor color="#FF0000">
</a-camera>
[ . . . ]
```

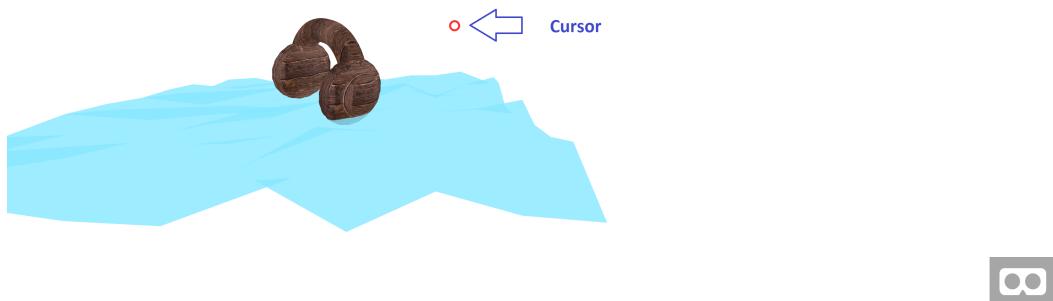


Figura 4.9: Cursor en una escena de A-Frame

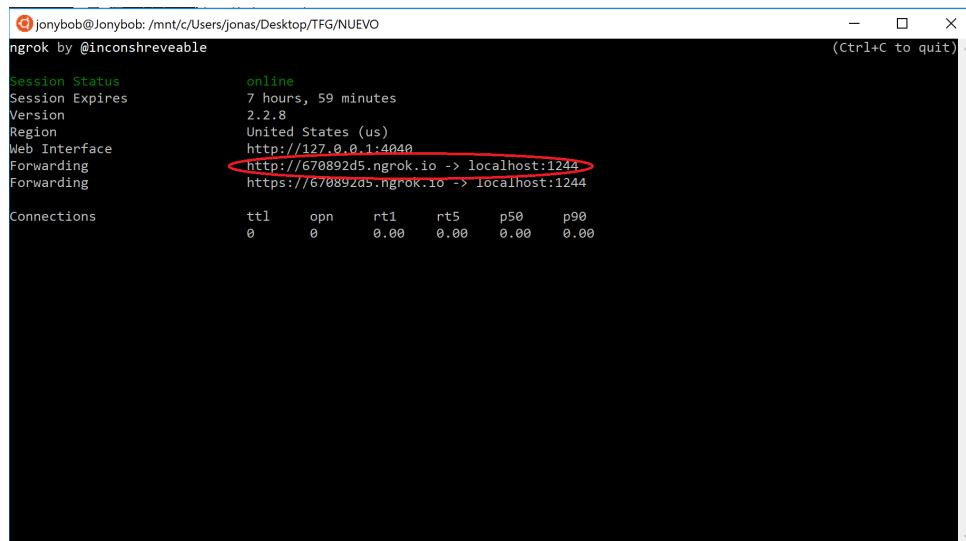
4.3.4. Llevar la experiencia VR a un smartphone

A continuación se expone un método para visualizar el entorno de A-Frame en un smartphone, si por ejemplo, no se cuenta con un visor físico para el ordenador. Para hacer esto se ha hecho uso de la utilidad **ngrok**, que permite la exportación de un servidor local a Internet, proporcionándole una IP pública y haciendo posible su acceso a través del navegador de un teléfono móvil. Uno de los problemas que se pueden encontrar con ngrok es que la experiencia no puede ser muy compleja, debido a que la transmisión de información no proporciona la velocidad normal de un servidor estándar.

- 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, lanzar el servidor de forma local:

```
python -m SimpleHTTPServer 1234
```
 2. Instalar ngrok, simplemente descargando el .zip y descomprimiéndolo en la carpeta raíz.
 3. Despues de esto es necesario ejecutarlo pasando 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.



A terminal window titled "jonybob@Jonybob: /mnt/c/Users/jonas/Desktop/TFG/NUEVO" showing the output of the ngrok command. The output includes session status, expiration time, version, region, and two forwarding entries. The second forwarding entry, which is the public URL, is circled in red.

```
jonybob@Jonybob: /mnt/c/Users/jonas/Desktop/TFG/NUEVO
ngrok by @inconschreivable
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      rti      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 programar el Micro Bit para convertirlo en el controlador de una experiencia de realidad virtual creada con A-Frame.

De la misma manera que en el segundo taller se utilizaba la figura creada en el primero para insertarla en A-frame, en este caso se parte de la escena creada en el segundo taller para controlar el movimiento dentro de ella con el Micro Bit. Conviene aclarar que el Micro Bit debe funcionar en cualquier entorno de A-Frame que permita movimiento, independientemente de como sea la experiencia en sí.

Una vez finalizado el taller, las funciones implementadas en el Micro Bit serán:

- 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.

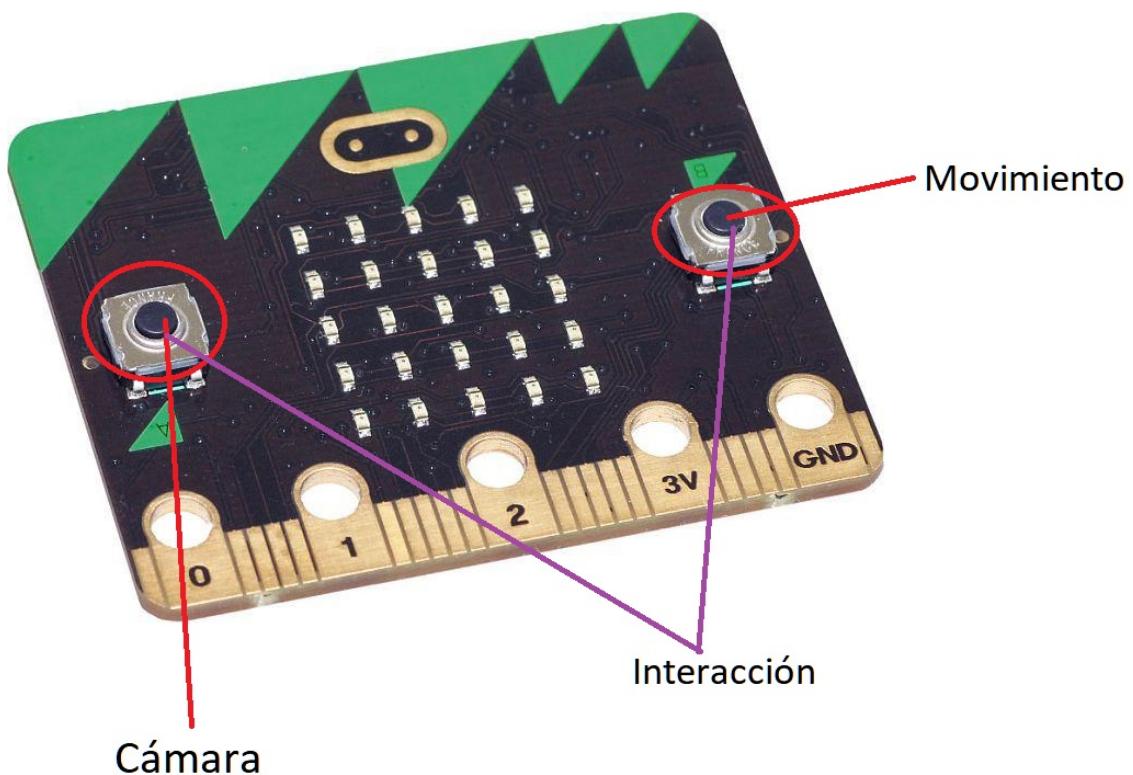


Figura 4.11: Funciones del Micro Bit

El primer paso antes de programar el Micro Bit es preparar el entorno. Como ya se ha expuesto 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 es abordado en el Apéndice B del trabajo.

Los requerimientos para empezar a programar el Micro Bit son:

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

Como editor de texto se puede utilizar aquel que sea más cómodo para el usuario, entre ellos el editor Mu que recomienda la documentación de Micro Bit. Respecto a la instalación de bitio y pyautogui, son abordadas en las siguientes secciones.

4.4.1. Instalación de Bitio

El primer paso es instalar Bitio, esta guía de instalación esta basada directamente en su repositorio de GitHub³. Como la propia documentación indica la mejor forma de usar bitio en un entorno educativo es copiar la carpeta ‘src/Micro Bit’ 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 Micro Bit, al principio del programa .py mediante la siguiente línea.

```
import microbit
```

Dentro de la carpeta ‘src’ se encuentran una serie de programas que permiten probar las distintas funcionalidades de bitio. Una forma de verificar que bitio está instalado correctamente es hacer una prueba con uno de los programas de prueba que se encuentran en esta carpeta.

³<https://github.com/whaleygeek/bitio>

Tomando como prueba *button.py*, si bitio está correctamente instalado, se obtienen estos valores de la salida estándar.

```
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
```

Si se analiza la salida, bitio sigue una serie de pasos al ejecutar un programa. Primero, se escanean los dispositivos conectados al ordenador, con el Micro Bit desconectado del ordenador, a continuación, se solicita al usuario conectarlo y tras confirmar, se establece la conexión al encontrar un nuevo dispositivo conectado.

Para verificar la conexión con la placa y que el programa está ejecutándose en la terminal se debe mostrar:

Now running your program

En el ejemplo expuesto anteriormente el programa devuelve una línea de texto cada vez que se pulsa el botón A.

4.4.2. Instalación de Pyautogui

El último paso antes de empezar a programar el Micro Bit es instalar pyautogui, para así poder mapear las funciones de las teclas del teclado y el ratón, respecto a los valores del acelerómetro del 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 hechos tanto en Linux como en Windows 10.

De la misma manera que se hizo con bitio, el método de instalación que se describe aquí esta basado en la propia documentación⁴

Los comandos a ejecutar para instalar pyautogui en Linux son:

```
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 y de la misma manera que con bitio, pyautogui es totalmente funcional al importarlo 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.

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

⁴<https://github.com/asweigart/pyautogui>

Su contenido es el siguiente.

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

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 para convertirlo en el controlador de la experiencia.

Para la realización del taller y la programación de las distintas funcionalidades del Micro Bit se ha optimizado el código del siguiente repositorio de GitHub⁵ proporcionado por el tutor, para implementar el movimiento físico. Para las otras dos funciones se ha utilizado la documentación tanto de bitio como de pyautogui.

Como se considera que la programación de la funcionalidades es demasiado compleja para la duración de los talleres, se ha optado por que los alumnos completen el código de una serie de programas que se adjuntan con el guión del taller, que se corresponden con las distintas funciones a implementar.

A continuación, se describe el estudio e implementación llevado a cabo con cada una de las funciones por parte del alumno:

A) Movimiento

B) Control de la cámara.

C) Interacción.

⁵<https://github.com/sarehp/microbit-aframe>

A) Movimiento

Para programar el movimiento dentro de A-Frame, se parte del código facilitado en el repositorio de GitHub, añadiéndole un paso previo que consiste en centrar la cámara antes de comenzar el movimiento y de esta forma facilitar la orientación.

```
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 creada en el anterior taller, consiste en una pequeña animación del objeto 3D en el escenario creado con A-Frame. La animación comienza cuando se hace click sobre el objeto. La función a implementar en el Micro Bit es hacer click con el ratón sobre el objeto cuando se pulsen los dos botones simultáneamente. Hay una función en `pyautogui` que permite esta orden.

```
pyautogui.click()
```

El método de detección elegido en este caso es la pulsación simultánea de los dos botones. Existe una función específica para detectar la pulsación de cada uno de los botones del Micro Bit, por tanto, basta con indicar con un *AND* la pulsación de ambos en el bloque *If* de la siguiente manera:

```
Micro Bit.button_a.is_pressed() and Micro Bit.button_b.is_pressed():
```

C) Control de la Cámara

En primera instancia, para programar el control de la cámara se estudiaron las diferentes funciones que ofrece pyautogui para interactuar con la pantalla y el ratón.

El primer paso para implementar el movimiento de la cámara con el acelerómetro del Micro Bit, es entender como se maneja la cámara en A-Frame con un ratón. 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.

La forma de implementar esta serie de acciones, es mapear el movimiento del ratón por la pantalla, en base a los valores que ofrece el acelerómetro del Micro Bit.

Para comenzar, es necesario extraer la resolución de la pantalla. 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 en esta memoria se describe la programación de la funcionalidad en base a una pantalla con una resolución Full HD (1920x1080), 1920 pixeles de ancho por 1080 pixeles de alto.

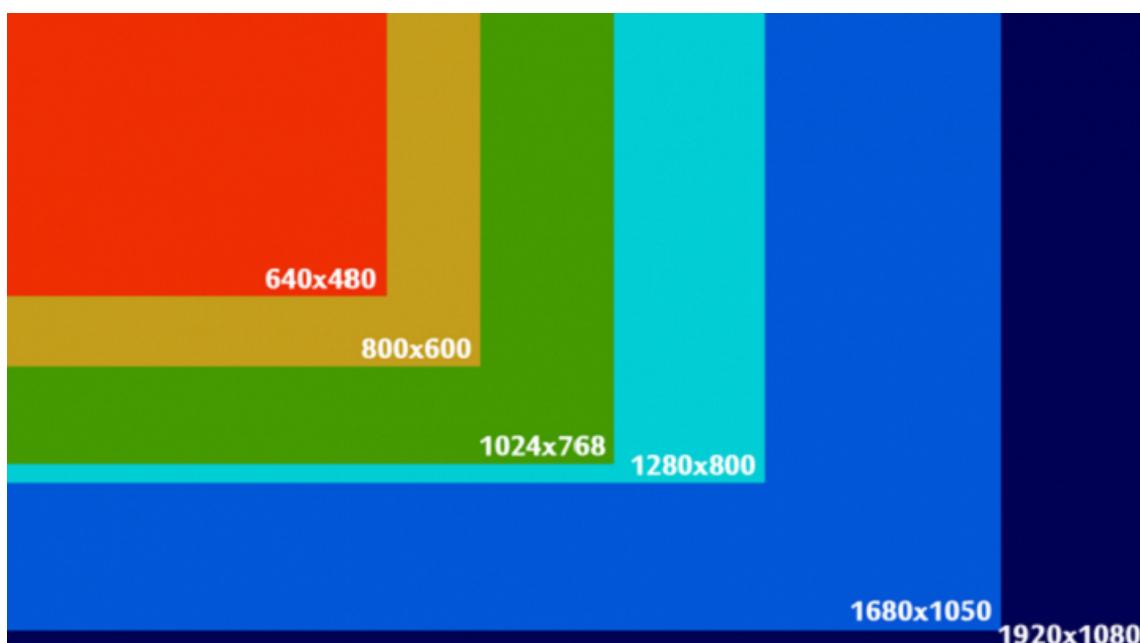


Figura 4.12: Resoluciones

Ahora se necesita saber como funciona el acelerómetro del Micro Bit. Con las funciones *Micro Bit.accelerometer.get_x()* y *Micro Bit.accelerometer.get_y()* se extraen los valores del acelerómetro los cuales oscilan de 1000 a -1000 en cada uno de los ejes, Figura 4.13.

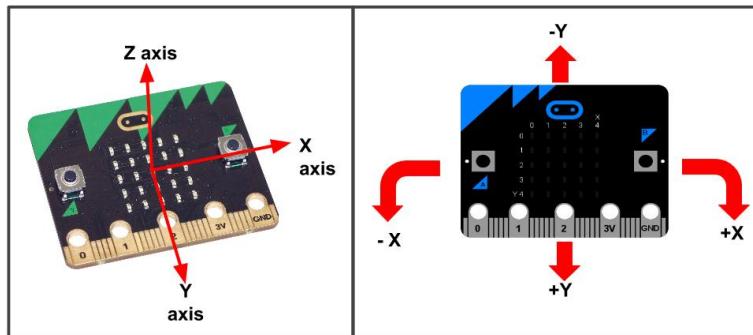


Figura 4.13: Valores del acelerómetro

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

Una vez se tienen claras las medidas 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.

Para mover la cámara en A-Frame es necesario mover el ratón mientras se mantiene pulsado el botón izquierdo del mismo. Para trasladar estas órdenes al Micro bit, son necesarias estas dos funciones de pyautogui.

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

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

Sabiendo esto, basta con implementar un método de sensibilidad en el acelerómetro, parecido al ya implementado con el movimiento físico donde se establecía un límite para paliar pequeñas desviaciones en los ejes.

```
x = Micro Bit.accelerometer.get_x()
y = Micro Bit.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)

```

La función de este código es mover el ratón a unas coordenadas concretas, en base a los valores que ofrece el acelerómetro en el **eje x** y el **eje y**. Las variables *SENX* y *SENY*, ofrecen una sensibilidad en los valores de los ejes del acelerómetro, en este caso, han sido fijadas a 180 y 280 respectivamente para prevenir desvíos no deseados, de una forma parecida al Threshold establecido en la función del movimiento implementada en el Micro Bit.

Por último, se puede observar que el valor del acelerómetro es multiplicado por dos variables **Height** y **Width**. Estas variables, ayudan a graduar el grado de sensibilidad del movimiento del ratón ayudando a la fluidez del movimiento. Son resultado de la división del ancho y el alto de la resolución de la pantalla, por un factor múltiplo de los valores ancho y alto de la pantalla.

```

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. En este caso es 6, debido a que la pantalla usada en las pruebas es 4K y la densidad de píxeles es muy elevada, tras una serie de pruebas, se ha establecido como un valor apropiado para tener un control óptimo de la experiencia, en el que la cámara no se mueva excesivamente rápido y exista una fluidez.

Capítulo 5

Conclusiones

En la última parte de este trabajo de fin de grado se analiza la consecución de los objetivos propuestos, las asignaturas cursadas durante el grado que han sido aplicadas, así como los conocimientos aprendidos por el alumno en la realización de este proyecto, así como las 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. Introducir la creación de objetos 3D y entornos VR.
3. Implementar una funcionalidad del Micro Bit en el entorno de A-Frame.

Para la consecución del primer objetivo, se llevó a cabo una descripción de la realidad virtual así como de las posibles utilidades y aplicaciones posibles para mejorar la educación en diferentes ámbitos, promoviendo 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 introducir a los alumnos el diseño 3D mediante la programación mediante bloques y la creación de una experiencia de realidad virtual propia, ha sido conseguido gracias a Beetle Blocks y A-Frame dos herramientas simples de entender basadas en la

programación por bloques y HTML/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, siendo la programación del Micro Bit usando micropython y consiguiendo la transformación en el controlador de la experiencia previamente creada con A-Frame. Este es el objetivo más complejo a la hora de ser impartido a alumnos en fase de acceso a la universidad, debido a la necesidad de una serie de conocimientos base de Python. Por tanto, el tercer taller se ha orientado de tal forma que, a medida que se desarrolla el guión, los alumnos deben completar el código de una serie de programas basados en las distintas funcionalidades a implementar en el Micro Bit. Esto hace que se deba tener un mínimo de conocimientos acerca de Python, pero facilita la comprensión a la hora de asentar las bases de la programación.

5.2. Aplicación de lo aprendido

Las principales asignaturas impartidas que están relacionadas con el desarrollo de este trabajo de fin de grado son:

1. INGENIERÍA DE SISTEMAS DE INFORMACIÓN
2. DESARROLLO DE APLICACIONES TELEMÁTICAS
3. SERVICIOS Y APLICACIONES TELEMÁTICAS

En ellas el alumno aprendió la programación con Python y JavaScript, además de aprender la aplicación de utilidades y conocimientos relacionados con HTML5.

5.3. Lecciones aprendidas

Los conocimientos aprendidos por el alumno en este trabajo de fin de grado corresponden al núcleo de los tres talleres desarrollados, es decir, la creación de objetos en 3D, la creación de experiencias de realidad virtual y la programación del Micro Bit.

Se puede afirmar, 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 posteriormente, aplicarlo en la redacción de los guiones 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, por ejemplo, la realidad aumentada.
- Mejoras en la experiencia, programando un sencillo juego que haga uso del Micro Bit dentro de A-Frame, podría contemplarse como un trabajo mucho más elaborado si se introdujera más de un Micro Bit y aumentara 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] D. M. donmccurdy. A-frame extras, Feb. 2019.
<https://github.com/donmccurdy/aframe-extras>.
- [5] E. R. Duks Koschitz, Bernat Ramagosa. Beetle blocks a new visual language for designers and makers. 2016.
- [6] P. S. Foundation. The python standard library, 2019.
<https://docs.python.org/2/library/index.html>.
- [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. whaleyleegeek. Bitio - a micro:bit io device, Apr. 2019.
<https://github.com/whaleyleegeek/bitio>.

- [11] J. G. Yeste. Beetle blocks y un caso practico. 2016.

Apéndice A

Guía de bloques de Beetle Blocks

Beetle Blocks 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 # Resetea todos los objetos renderizados.
- When Green Flag clicked # Ejecuta el código anidado al hacer click.
- When space(↓) key pressed
- When I receive (↓)
- Broadcast (↓)
- [Checkbox] message
- Warp
- Wait (n) secs # Espera un periodo de tiempo.
- Wait until <boolean>
- Forever # Corre siempre los códigos insertados.
- Repeat (n) # Bucle.
- Repeat until <boolean> # Repite el código hasta que se de una condición.
- If <boolean> # Condicional.
- If <boolean> else # Condicional completo.
- Report [string]
- Stop all(↓) # Para la ejecución de lo especificado en el selector.
- Stop all but this script # Para la ejecución de todo menos lo especificado.
- Run (...)
- Launch (...)
- Call (input)
- Run (...) w/continuation
- Call (...) w/continuation
- Pause all # Para el renderizado.

A.3. Colores

- Set hue(↓) to (n) # Establece un color para el dibujado
- Change hue(↓) by (n) # Cambia el color
- 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(↓) # Permite dibujar líneas o curvas
- Stop drawing # Para de dibujar.
- Start extruding curves(↓)
- Stop extruding
- Set extrusion Dia. to (n)
- Change extrusion Dia. by (n)

A.5. Funciones

- Request user ... [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>
- (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 que 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.

A.7. Variables

- set (\downarrow) to (...) # Establece el valor de la variable.
- change (\downarrow) by (n) # Reemplaza la variable por el valor. dado.
- show variable (\downarrow) # Muestra la variale introducida.
- hide variable (\downarrow) # Oculta la variable introducida.
- script variables a(...) <increase>
- list (...) <increase>
- (...) in front of list
- item 1(\downarrow) of list
- all but first of list
- length of list
- list contains thing(...)
- add thing(...) to list # Añade a la lista el elemento introducido.
- delete 1(\downarrow) of list # Borra el elemento solicitado.
- insert thing(...) at 1(\downarrow) of list # Inserta el elemento en la posición específica de la lista.
- replace item (\downarrow) of list with(...) # Reemplaza un elemento por otro en la lista.

A.8. Crear un bloque

En la sección de crear un bloque, se tiene un único bloque hasta que el usuario crea alguno. Para crear un bloque, es necesario hacer click en el botón correspondiente, esto abre un cuadro de diálogo 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.9. Operadores

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

A.10. Imágenes de los Bloques



Figura A.1: Colores

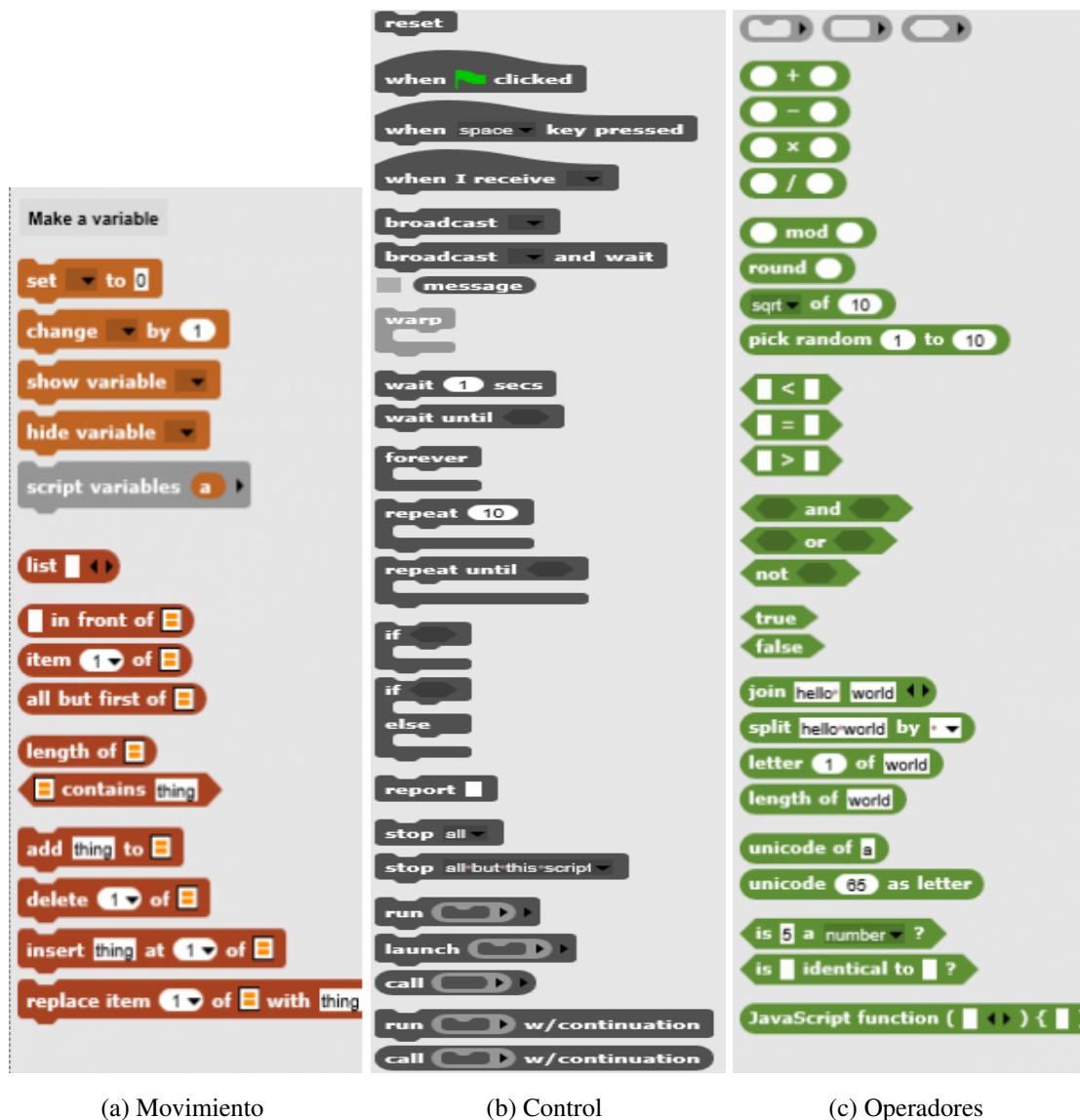


Figura A.2: Bloques Beetle Blocks A

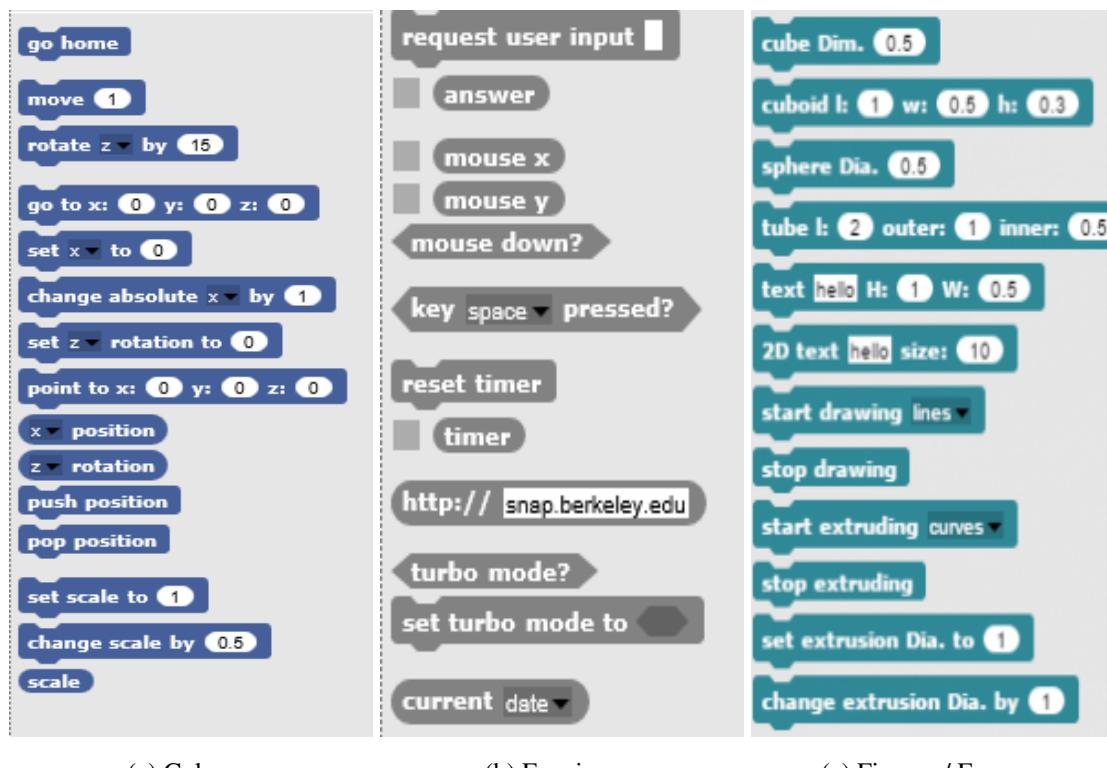


Figura A.3: Bloques Beetle Blocks B

Apéndice B

Programar el Micro Bit en Windows

Para usar el Micro Bit en windows, es necesario instalar una compilación de Python, en Windows 10 basta con buscar en la tienda Python 3.7¹ e instalar la aplicación.

Una vez instalado Python, el siguiente paso es instalar bitio y pyautogui, pero antes es necesario instalar el driver *mbed serial driver*², para que la terminal de windows pueda tener acceso al serial port del Micro Bit.

La instalación de bitio es igual que la detallada para Linux, basta con descargar el repositorio a la raíz del proyecto, o si se desea tener un entorno más controlado descargar solo la carpeta /microbit.

Para pyautogui es necesario ejecutar una serie de pasos.

- Instalar pip, para ello es necesario descargar primero el archivo get-pip.py en el directorio del proyecto.

```
$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

- Ejecutarlo con la ayuda de una terminal de Windows:

```
$ python get-pip.py
```

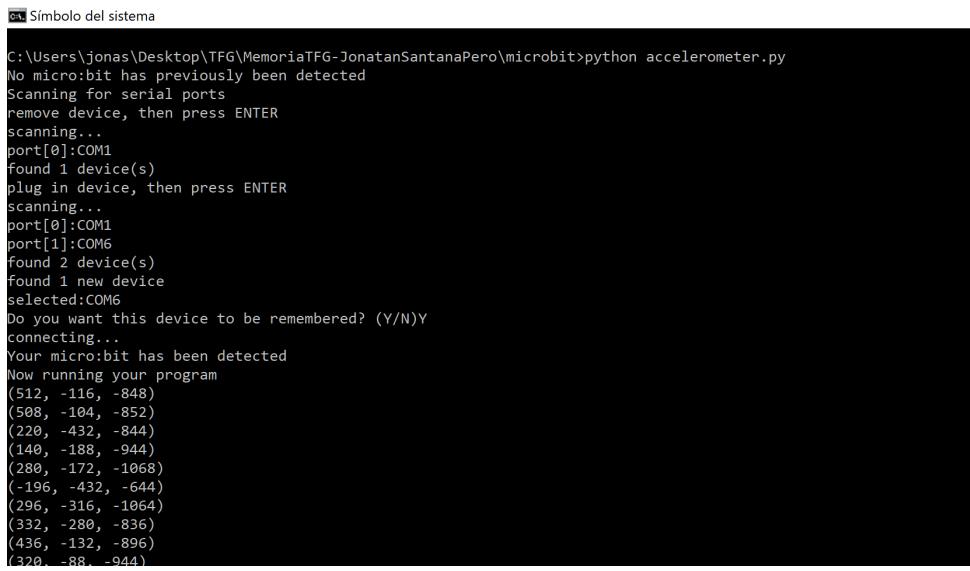
- Por último instalar pyautogui

```
$ pip install pyautogui
```

¹<https://www.microsoft.com/en-us/p/python-37/9nj46sx7x90p?activetab=pivot:overviewtab>

²<https://os.mbed.com/handbook/Windows-serial-configuration>

Una vez instalado Python, bitio y pyautogui ya es posible ejecutar cualquier programa para el Micro Bit desde una terminal de windows como se puede ver en la Figura B.1.



```
C:\Símbolo del sistema
C:\Users\jonas\Desktop\TFG\MemoriaTFG-JonatanSantanaPero\microbit>python accelerometer.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)Y
connecting...
Your micro:bit has been detected
Now running your program
(512, -116, -848)
(508, -104, -852)
(220, -432, -844)
(140, -188, -944)
(280, -172, -1068)
(-196, -432, -644)
(296, -316, -1064)
(332, -280, -836)
(436, -132, -896)
(320, -88, -944)
```

Figura B.1: Cmd corriendo programa de micropython

Apéndice C

Programas de Python

C.1. Testpy.py

```
1 import microbit
2 import pyautogui
3
4 while True:
5     if microbit.button_a.was_pressed():
6         microbit.display.show("PULSA A")
7         screenWidth, screenHeight = pyautogui.size()
8         pyautogui.moveTo(screenWidth / 2, screenHeight / 2)
9         microbit.sleep(500)
10        microbit.display.clear()
```

C.2. interaction.py

```
1 import pyautogui
2 import microbit
3
4 #Function
5 def Interaction():
6     if % %and % %
7         print("Button A & B pressed , click mode")
8         %
9
```

```

10 #MAIN
11 while True:
12     Interaction()
13     microbit.sleep(500)

```

C.3. move.py

```

1 import pyautogui
2 import microbit
3
4 #Global variables
5 THRESHOLD=200.0
6
7 #Functions
8 def Move():
9     while True:
10         %%%= microbit.accelerometer.get_x()
11         y = microbit.accelerometer.get_%%%0
12
13         if x > THRESHOLD:
14             pyautogui.keyDown(% %)
15         elif x < -THRESHOLD:
16             pyautogui.keyDown(% %)
17         else:
18             pyautogui.keyUp("right")
19             pyautogui.keyUp("left")
20
21         if y > % %
22             pyautogui.keyDown(% %)
23         elif y < -% %:
24             pyautogui.keyDown("down")
25         else:
26             pyautogui.keyUp(% %)
27             pyautogui.keyUp(% %)
28
29
30

```

```
31 def Center_Cam():
32     screenWidth, screenHeight = pyautogui.size()
33     pyautogui.mouseDown()
34     pyautogui.moveTo(screenWidth / 2, screenHeight / 2)
35     pyautogui.mouseUp()
36
37 #MAIN
38 while True:
39
40     Center_Cam()
41     Move()
42
43     microbit.sleep(500)
```

C.4. camera.py

```
1 import pyautogui
2 import microbit
3 import msvcrt, sys
4
5 #Global variables
6 THRESHOLD=200.0
7 SENX=180.0
8 SENY=280.0
9
10 #Functions
11 def Camera(Width, Height):
12     while True:
13         % %=% microbit.accelerometer.get_%_()
14         % %
15
16         if % %> SENX:
17             pyautogui.moveRel(% %*x, None)
18         elif % %< -SENX:
19             pyautogui.moveRel(% %*x, None)
20         if % %> SENY:
21             pyautogui.moveRel(None, y*% %)
22         elif % %< -SENY:
23             pyautogui.moveRel(None, y*% %)
24
25         if microbit.button_b.is_pressed():
26             pyautogui.%
27             break
28
29
30 def Center_Cam():
31     screenWidth, screenHeight = pyautogui.size()
32     pyautogui.mouseDown()
33     pyautogui.moveTo(screenWidth / 2, screenHeight / 2)
34     pyautogui.mouseUp()
35
```

```

36 #MAIN
37 while True:
38     #Interaction
39     if %and %:
40         print("Buttons A & B pressed , click mode")
41     #Camera
42     elif %:
43         print("Button A pressed")
44         print("camera mode, press A again to exit")
45         screenWidth , screenHeight = pyautogui.size()
46         Factor_y = 1920
47         Factor_x = 1080
48         Width = screenWidth / Factor_y
49         Height = screenHeight/ Factor_x
50         %mouseDown()
51         Camera(Width , Height)
52     #Movement
53     elif %:
54         print("Button B pressed")
55         print("movement mode, press B again to exit")
56         Center_Cam()
57         Move()
58         microbit.sleep(500)

```

C.5. microIU.py

```

1 import pyautogui
2 import microbit
3 import msvcrt , sys
4
5 #GLOBAL
6 THRESHOLD=200.0
7 SENX=180.0
8 SENY=280.0
9 AYUDA="""        HELP
10      Press A to move the camera ,
11      Press B to move ,

```

```
12     Press A & B to interact"""
13
14 #FUNCTIONS
15 def Help_Windows():
16     ch = msvcrt.getwch()
17     if ch == 'q':
18         sys.exit()
19     elif ch == 'h':
20         print(AYUDA)
21     else:
22         print ("Wrong Key Pressed")
23         print("Press H to help , Q to exit")
24
25 def Help_Linux():
26     ch = input("Press H to help , Q to exit")
27     if ch == 'q':
28         sys.exit()
29     elif ch == 'h':
30         print(AYUDA)
31     else:
32         print ("Wrong Key Pressed")
33         print("Press H to help , Q to exit")
34
35 def Camera(Width):
36     while True:
37         x = microbit.accelerometer.get_x()
38         y = microbit.accelerometer.get_y()
39
40         if x > SENX:
41             pyautogui.moveRel(Height*x, None)
42         elif x < -SENX:
43             pyautogui.moveRel(Height*x, None)
44         if y > SENY:
45             pyautogui.moveRel(None, y*Width)
46         elif y < -SENY:
47             pyautogui.moveRel(None, y*Width)
```

```
49     if microbit.button_b.is_pressed():
50         pyautogui.mouseUp()
51         break
52 def Move():
53     while True:
54         x = microbit.accelerometer.get_x()
55         y = microbit.accelerometer.get_y()
56
57         if x > THRESHOLD:
58             pyautogui.keyDown("right")
59         elif x < -THRESHOLD:
60             pyautogui.keyDown("left")
61         else:
62             pyautogui.keyUp("right")
63             pyautogui.keyUp("left")
64
65         if y > THRESHOLD:
66             pyautogui.keyDown("up")
67         elif y < -THRESHOLD:
68             pyautogui.keyDown("down")
69         else:
70             pyautogui.keyUp("down")
71             pyautogui.keyUp("up")
72
73         if microbit.button_a.is_pressed():
74             break
75
76 def Center_Cam():
77     screenWidth, screenHeight = pyautogui.size()
78     pyautogui.mouseDown()
79     pyautogui.moveTo(screenWidth / 2, screenHeight / 2)
80     pyautogui.mouseUp()
81
82
83
84
85
```

```
86 #MAIN
87 while True:
88     if msvrt.kbhit():
89         Help_Windows()
90         #Help_Linux()
91
92     elif microbit.button_a.is_pressed() and microbit.button_b.is_pressed():
93         print("Button A & B pressed , click mode")
94         pyautogui.click()
95
96     elif microbit.button_a.is_pressed():
97         print("Button A pressed , camera mode, press B to exit")
98         screenWidth, screenHeight = pyautogui.size()
99         Factor_y = 1920*6 # Factor 6 porque la pantalla es 4k
100        Factor_x = 1080*6
101        Width = screenWidth / Factor_y # Ajusto la sensibilidad
102        Height = screenHeight/ Factor_x
103        pyautogui.mouseDown()
104        Camera(Width)
105
106    elif microbit.button_b.is_pressed():
107        print("Button B pressed , movement mode, press A to exit")
108        #Centramos la posicion de la camara antes de movernos
109        Center_Cam()
110        Move()
111
112    #print(microbit.accelerometer.get_values())
113    microbit.sleep(500)
```

Apéndice D

Talleres didácticos