Proyecto Fin de Carrera
Ingeniería de las Tecnologías Industriales

Recreación de Planta Termosolar y flota de vehículos no tripulados para representación de simulaciones en Unity

Autor: María Tejero Rodríguez
Tutor: José María Maestre Torreblanca
Javier García Martín

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2013

Proyecto Fin de Carrera
Ingeniería en Tecnologías Industriales

Recreación de Planta Termosolar y flota de vehículos no tripulados para representación de simulaciones en Unity

Autor:
María Tejero Rodríguez

Tutor:
José María Maestre Torreblanca
Javier García Martín
Profesor titular

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Proyecto Fin de Carrera: Recreación de Planta Termosolar y flota de vehículos no tripulados para representación de simulaciones en Unity

Autor: María Tejero Rodríguez

Tutor: José María Maestre Torreblanca

Javier García Martín

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

A mis padres y a mi hermana, por su apoyo incondicional.

A mis amigos por acompañarme y animarme a avanzar durante toda esta etapa.

A mis profesores por enseñarme tanto; y en especial a mis tutores, por ayudarme a hacer posible este proyecto.

Resumen

Unity es una herramienta de desarrollo virtual en tiempo real con muchas posibilidades, el objetivo de este proyecto es el de sacar beneficio a esta herramienta en el ámbito tecnológico. Mediante el uso de las múltiples posibilidades gráficas se recreará una Planta Termosolar de colectores cilindroparabólicos. A esta simulación, se agregarán vehículos no tripulados aéreos y terrestres con el propósito de poder reproducir la trayectoria de los mismos. El comportamiento de los robots será simulado en el entorno de desarrollo integrado Visual Studio. Los puntos que conformarán el recorrido de los robots podrán variar en función de los parámetros que se manden desde Matlab mediante comunicación TCP/IP. Finalmente, la simulación se cargará sobre unas gafas de Realidad Virtual, permitiendo visualizar la recreación de una manera mucho más envolvente y a su vez dando la posibilidad de que el propio observador pueda moverse según sus preferencias con la ayuda de los controles propios de las gafas.

Abstract

Unity is a real-time 3D development tool with multiple possibilities that is exploited in this project to recreate a solar thermal power plant with parabolic trough collectors. In particular, unmanned aerial and ground vehicles are represented as they complete their routes to take irradiance measurements in the plant. These trajectories are generated in Matlab and transmitted via TCP/IP to Visual Studio, which controls the representation in Unity. Finally, the simulation can be also seen using virtual reality glasses in a much more surrounding experience for the user, who can control several aspects from the glasses' controllers.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Notación

A^*	Conjugado
c.t.p.	En casi todos los puntos
c.q.d.	Como queríamos demostrar
■	Como queríamos demostrar
e.o.c.	En cualquier otro caso
e	número e
IRe	Parte real
IIm	Parte imaginaria
sen	Función seno
tg	Función tangente
arctg	Función arco tangente
sen	Función seno
$\sin^x y$	Función seno de x elevado a y
$\cos^x y$	Función coseno de x elevado a y
Sa	Función sampling
sgn	Función signo
rect	Función rectángulo
Sinc	Función sinc
∂y ∂x	Derivada parcial de y respecto
$x_\circ$	Notación de grado, x grados.
Pr(A)	Probabilidad del suceso A
SNR	Signal-to-noise ratio
MSE	Minimum square error
:	Tal que
<	Menor o igual
>	Mayor o igual
\	Backslash
⇔	Si y sólo si

1 OBJETIVOS Y ALCANCES

El objetivo principal de este proyecto es crear un programa que permita recrear la trayectoria de vehículos terrestres y aéreos no tripulados en una Planta Termosolar, tomando los puntos que conforman la misma desde Matlab.

Para ello, es necesario establecer una comunicación entre ambos programas. Esta conexión deberá ser bidireccional; es decir, Matlab deberá enviar datos a Visual Studio pero también esperará a recibir un mensaje de confirmación por parte de este.

En Matlab se deberá tener en cuenta, hasta un máximo de diez robots, que los datos pueden corresponder tanto a un robot terrestre como a uno aéreo. El número de puntos que conformen la trayectoria podrá variar.

Como se ha comentado previamente los robots podrán ser tanto terrestres como aéreos, por ello se deberá contemplar, a la hora de procesar el mensaje obtenido por Matlab, a qué objeto mandar cada mensaje.

Se deberá conseguir que los robots sigan la trayectoria formada por los puntos y que circulen por el campo solar de colectores. Se deberá simular una Planta Termosolar con un total de treinta y cuatro lazos que permita a los vehículos no tripulados circular por debajo de los colectores, en el caso de los terrestres, y circular por las conexiones entre colectores de tuberías que van por suelo, en el caso de los aéreos. Los edificios de la Planta serán reunidos en el centro de la misma y se deberán simular las diferentes partes que forman el circuito termodinámico correspondiente, conectando todo por medio de tuberías.

Finalmente, se exportará el programa a un dispositivo de realidad virtual, dando la posibilidad al observador de poder moverse por la planta utilizando mandos.

2 UNITY

2.1 Unity

Unity es una herramienta de desarrollo de videojuegos, lo que se conoce como un motor de desarrollo o motor de juegos, creada por la empresa Unity Technologies. El término motor de videojuego o "*game engine*", hace referencia a un software el cual tiene una serie de rutinas de programación que permiten el diseño, creación y el funcionamiento de un entorno interactivo.

Se trata de una herramienta que no engloba únicamente motores para el renderizado de imágenes, de física 2D/3D, de audio, de animaciones y otros motores, sino que engloba además herramientas para trabajar en equipo de manera remota, herramientas de navegación NavMesh para Inteligencia Artificial o soporte de Realidad Virtual.

Dentro de su página web (ver Figura 1) en la sección "*Made with Unity*", se pueden ver diferentes juegos que han sido creados con este software; entre los que se pueden encontrar algunos conocidos como "*Hollow Knight*"[1].

Además de para crear videojuegos, se ha utilizado para crear experiencias de Realidad Virtual interactivas e incluso miniseries, como "*Baymax Dreams*", producida por Disney junto con Unity, donde se ha utilizado el editor para procesar y previsualizar en tiempo real todos los capítulos de la miniserie.

Por tanto, Unity3D es un software que permite el desarrollo de simulaciones y que además, permite su adaptación para diversas plataformas, en concreto más de 25 medios, mediante un editor visual y programación vía scripting. Otro punto fuerte de Unity se trata de la gran comunidad de usuarios que tiene detrás. Esto permite acceso a multitud de documentación, foros y comunidades donde se preguntan y resuelven dudas. [1]

A modo de documentación, Unity cuenta con un manual online donde explica como aprender a usar los diferentes servicios disponibles.[2]

2.2 Proceso de instalación

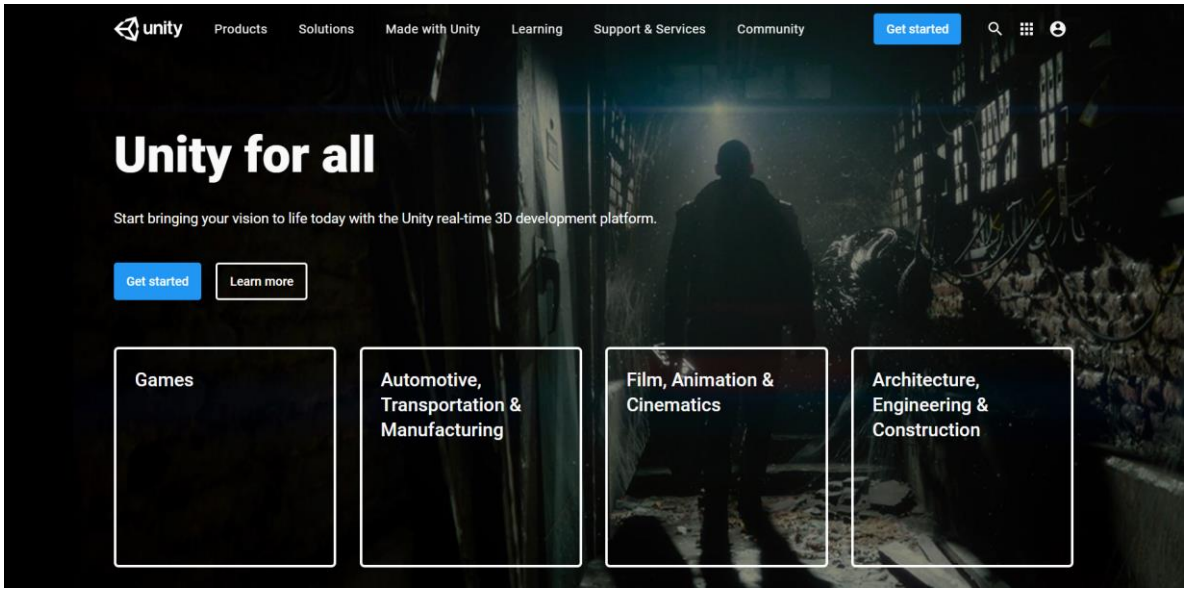

Figura 1. Página web *http://unity.com*

Para descargar el software, se accederá a la página principal de Unity (ver Figura 1). Dentro de la web se

[1] Hollow Knight es un videjuego realizado por la desarrolladora de videojuegos indies, Team Cherry.

pueden adquirir productos y además, tener acceso a cursos, soluciones, la comunidad o servicios que se ofrecen. Tras pulsar "*Get Started*", la página mostrará los diferentes niveles de licencia que en función de la magnitud del proyecto será necesario adquirir. Una vez elegido el nivel, se podrá escoger entre dos modos de inicio: "*First-time Users*", que te da la oportunidad de crear un primer juego a modo de tutorial; y "*Returning Users*", enfocado para los que ya manejan el software y quieren comenzar a trabajar tras la descarga.

2.3 Unity Hub

El archivo a descargar será el ejecutable "*UnityHubSetup*" que instalará Unity Hub; el cual se trata de una aplicación escritorio independiente la cual permite gestionar de manera ordenada y centralizada todas las versiones de Unity que se tengan instaladas, y los diferentes proyectos que se estén realizando. Una vez se hayan seguido y completado todos los pasos de la instalación, se podrá ejecutar el programa.

En la Figura 2 se puede observar la interfaz de Unity Hub. En ella se tienen tres pestañas: "*Projects*", en la que se encontraran todos los proyectos en los que se haya trabajado anteriormente y también donde se podrá crear uno nuevo; "*Learn*", donde hay diferentes tutoriales y proyectos con los que empezar a aprender cómo utilizar Unity; "*Install*", esta pestaña indica las últimas versiones de Unity y permitirá actualizar la herramienta.

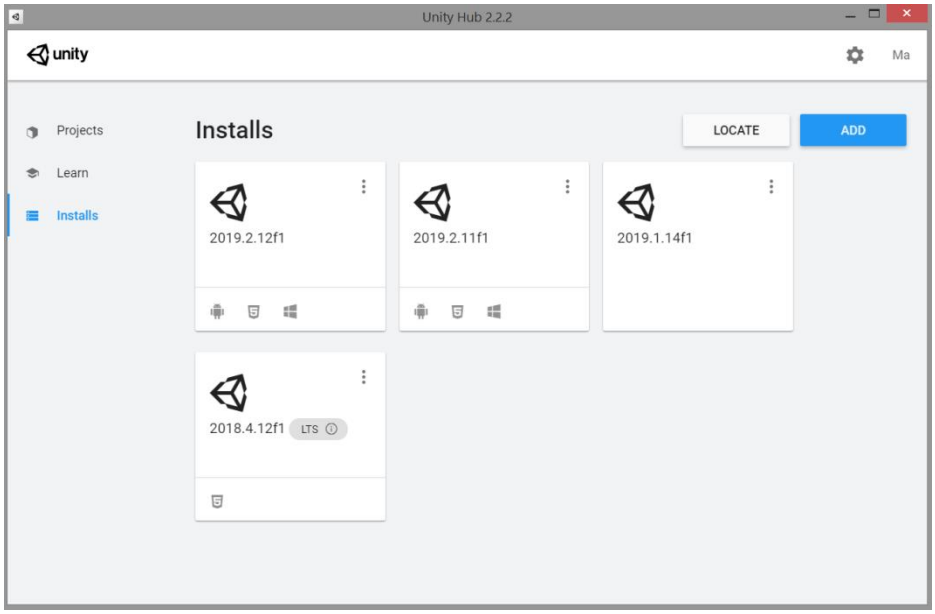

Figura 2. Interfaz Unity Hub.

Para instalar Unity Editor, que será el software donde se trabajará, se tiene que pulsar sobre la pestaña "*Installs*". En primer lugar, se deberá pulsar sobre el botón "*Add*", lo cual abrirá una nueva ventana que permitirá escoger la versión de Unity Editor que se quiera descargar. Para continuar con el proceso, se deberá presionar sobre "*Next*". A continuación, se permitirá seleccionar los diferentes módulos que se quieran instalar con esa versión de Unity Editor. Si no se instala en ese mismo momento, permite agregarlos posteriormente. Una vez se hayan elegido todos los módulos necesarios, se debe pulsar sobre "*Done*" y comenzará automáticamente la instalación.

En el caso de que se quisiese añadir algún módulo que durante el proceso de instalación no se haya añadido, se debe pulsar sobre los tres puntos de la casetilla de la versión a la que queramos añadir dichos módulos. Finalmente, al pulsar "*Add modules*", tal y como se muestra en la Figura 3, se podrá elegir entre los módulos cuales añadir.

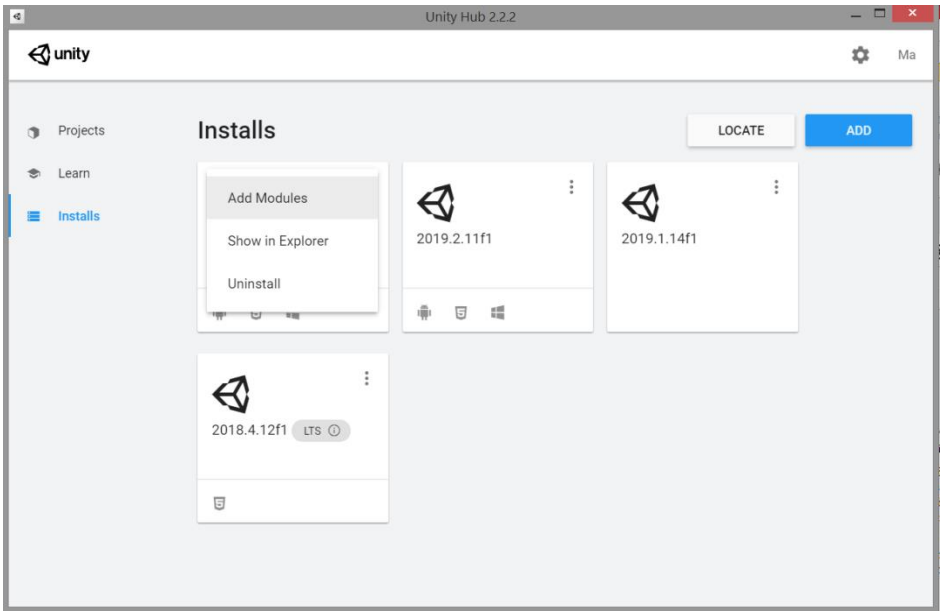

Figura 3. Selección de añadir módulos con versión previamente instalada.

Para acceder a los ajustes de Unity Hub, se tendrá que acceder a la tuerca que hay en la parte superior derecha de la ventana. Aquí, además de cambiar el idioma y la dirección, se puede cambiar también la licencia.

En el caso de querer crear un nuevo proyecto, bastará con entrar en la pestaña "*Projects*" y pulsar sobre "*New*"; lo que abrirá una nueva ventana, Figura 4, que permitirá realizar la configuración inicial del nuevo proyecto.

En "*Project Name*" se le asigna un nombre al proyecto. Este nombre será el que lleve la carpeta donde se guarde toda la información. Esta carpeta se creará automáticamente donde se le indique en el apartado "*Location*". Para cambiar la ubicación, se debe pulsar sobre los tres puntos seguidos, esto abrirá una nueva ventana donde se permite seleccionar de manera manual la dirección preferente. Otra opción sería escribir sobre la línea la propia dirección.

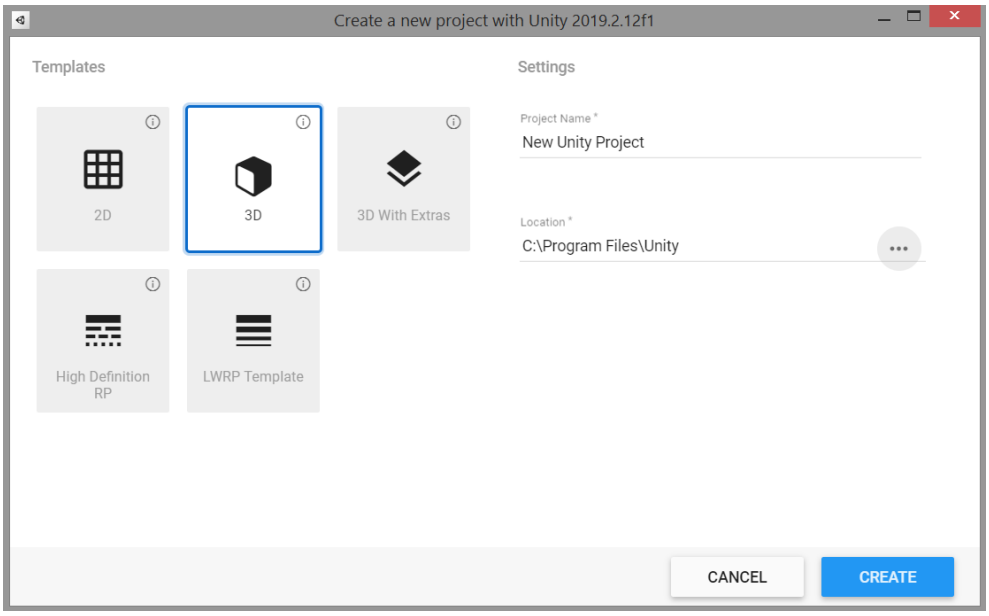

Figura 4. Ventana de creación de un nuevo proyecto.

Otra alternativa de elección disponible es el tipo de plantillas o "*Templates*". Esta selección proporciona configuraciones preseleccionadas en función de los tipos de proyectos más comunes en Unity. El tipo predeterminado de plantilla es el "*3D*"; aunque si fuese preciso, también se da la posibilidad de pasarlo a 2D una vez dentro del proyecto ya creado.

Los juegos 3D, como por ejemplo "*Timelapse*" [2] en la Figura 5, crea la tridimensionalidad con materiales y texturas renderizadas en la superficie de los GameObjects para que aparezcan como entornos, personajes u objetos sólidos que compondrán el mundo virtual. La cámara puede moverse dentro y alrededor de la escena libremente, con luces y sombras proyectadas alrededor del mundo de manera realista. También se tiene en cuenta la perspectiva, por lo que los objetos aparecen más grandes en la pantalla a medida que se acercan a la cámara.

También existe la plantilla "*2D*", como se puede ver en la Figura 6 en el juego "*Cuphead*"[3] desarrollado en Unity, en los que se usan gráficos llamados sprites, que es geometría no tridimensional. En la pantalla aparece como una imagen plana, y no habrá por tanto perspectiva.

La plantilla "*3D with Extras*" configura los ajustes del proyecto para que las aplicaciones 3D usen el renderizado y también las funciones de procesamiento posterior. Esto incluye una nueva forma de postprocesamiento, varios ajustes de inicio de desarrollo y contenido de ejemplo.

Por otro lado, existe la opción de plantilla "*High Definition RP*" que se suele utilizar para proyectos de alta gama. Esta plantilla está construida utilizando "*Scriptable Render Pipeline*", que permite adaptar el proceso de renderizado a la plataforma de destino para que se pueda optimizar el rendimiento para hardware específico. Incluye además iluminación y materiales de base física.

Figura 5. Timelapse, juego 3D desarrollado en Unity.

Figura 6. Cuphead, juego 2D desarrollado en Unity.

[2] Time lapse es un juego desarrollado por la empresa NightDive Studios, LLC.
[3] Cuphead es un juego de plataformas desarrollado y producido por StudioMDHR.

2.4 Interfaz de Unity

Una vez se haya creado o abierto el nuevo proyecto, aparecerá la interfaz de Unity Editor (ver Figura 7). Como se puede observar, Unity dispone de diversas ventanas independientes con las que se debe familiarizar para poder hacer un completo uso del programa.

Comenzando por la ventana denominada A, se encuentra la llamada Ventana Escena, la cual permite configurar y navegar por la simulación. A su lado se localiza la ventana B, llamada la Ventana del juego, la cual permite tener una visión de cómo quedará la simulación. Justo debajo, se encuentra la Ventana del Proyecto, C, en la que se dispondrá de acceso a todas las carpetas como los "*Assets*" o los "*Packages*". A la izquierda, se encuentra la ventana D, conocida como Ventana de Jerarquía. En ella se encontraran todos los GameObject que haya en la Escena. Si se seleccionase algún objeto de la lista en la Ventana de Jerarquía, se podrá visualizar sus características y opciones de configuración en la Ventana Inspector, la E. Finalmente, en la parte superior y como letra F, estará la barra de herramientas que cuenta con un conjunto de opciones que son las más utilizadas a la hora de trabajar, como es el botón de Play que inicia la simulación.

Unity permite añadir más ventanas con distintos tipos de funciones, como puede ser la Ventana de Animación que sirve para crear animaciones. En el caso de que se abriera una nueva ventana o que se quisiera cambiar la distribución de las ventanas ya existentes, se pueden arrastrar a la zona donde se quisiera que se ubiquen, o añadirlos dentro de una ventana presente y que aparezca como una pestaña en esta misma. Esta última opción permite alternar entre ventanas ambas ventanas según la necesidad. Por ejemplo, en la Figura 7 la Ventana de la Consola aparece como pestaña al lado de la Ventana del Proyecto. En cuanto se pulsase sobre Consola esta sustituiría la de Proyecto, la cual aparecería como pestaña justo como actualmente aparece Consola. [3]

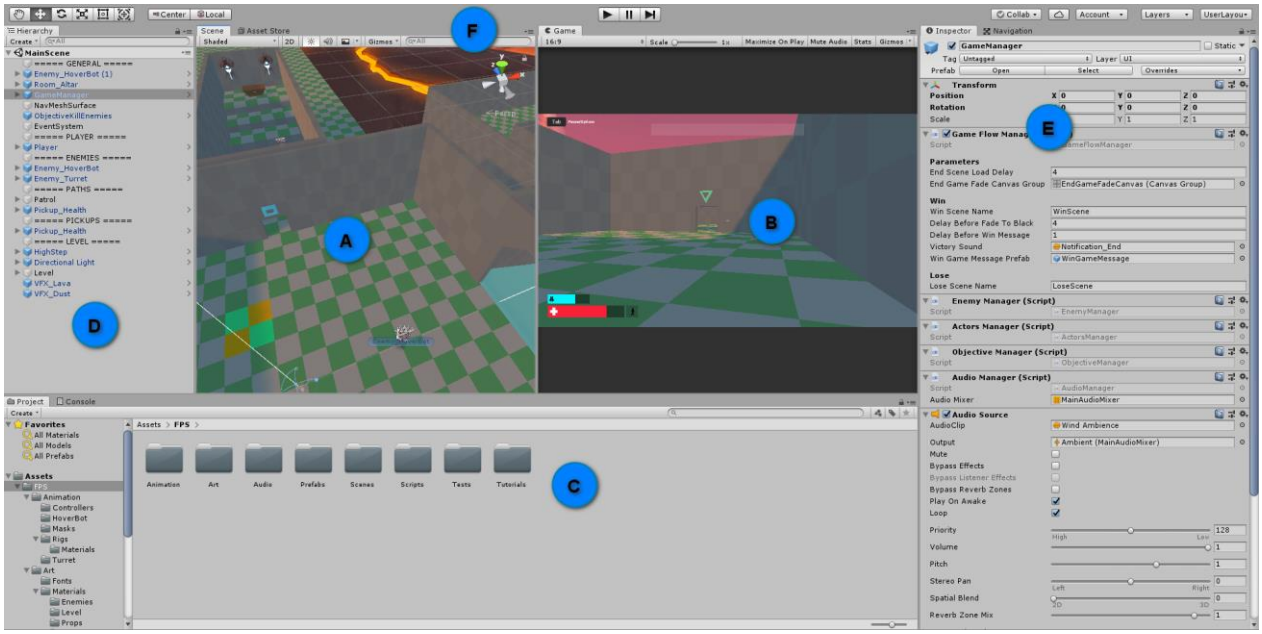

Figura 7. Interfaz de Unity.[4]

[4] Imagen obtenida del tutorial accesible desde Unity Hub, "FPS Microgame".

2.4.1 **Ventana Escena**

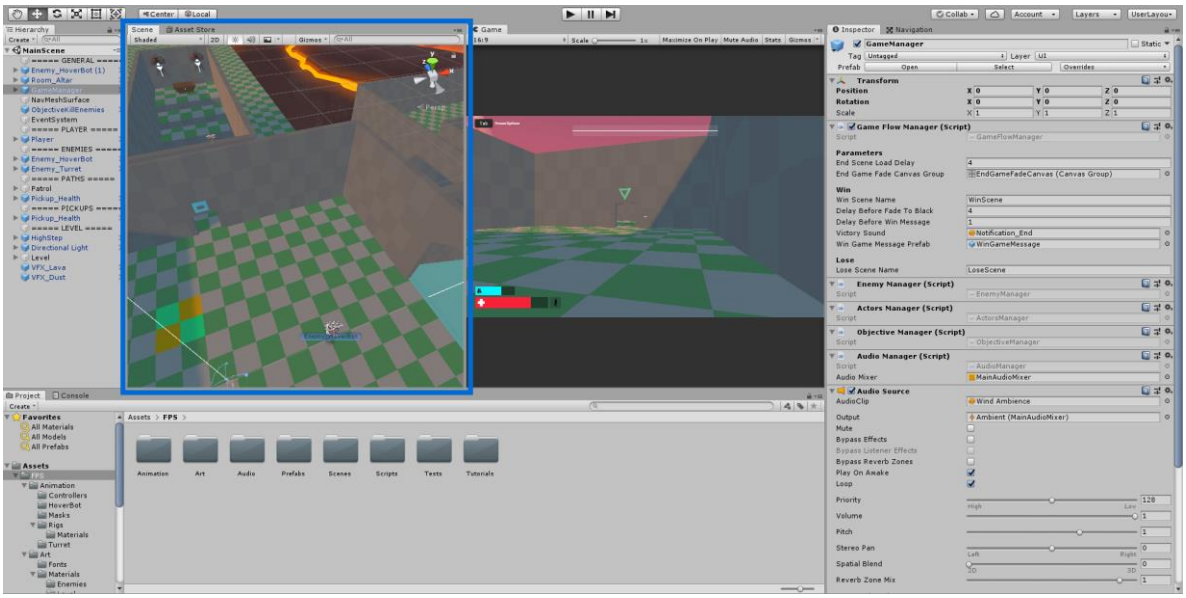

Figura 8. Ventana Escena.

Permite visualizar y editar la escena, la cual puede estar en 2D o 3D en función del tipo de proyecto en el que se esté trabajando. En el caso de la Figura 8, se ha estado trabajando en un proyecto 3D y por tanto, la escena será tridimensional.

Otra particularidad de esta ventana es que se pueden seleccionar un solo o varios GameObjects desde la misma. Una vez que se haya seleccionado, aparecerá el propio objeto en la Ventana Inspector (ver Figura 26), y también aparecerá remarcado en la Ventana de Jerarquía (ver Figura 24). De la misma manera, si se seleccionase más de un objeto en la escena, todos saldrán remarcados en la Ventana de Jerarquía, pero en la Ventana Inspector solo estará el último que se haya seleccionado.

Dentro de la ventana se encuentran diferentes herramientas a utilizar; entre ellas el "*Scene Gizmo*" (ver Figura 9), la cual se puede observar en la esquina superior derecha de la Ventana Escena. Este útil muestra la orientación de la cámara de la propia ventana. Además, permite modificar el ángulo de visión y el modo de proyección. Para ello, basta con pulsar en cualquiera de los brazos cónicos, los cuales están etiquetados como "X", "Y" e "Z", y se ajustaría la cámara al eje que representa el brazo cónico pulsado. También se podría pulsar con el botón derecho en el cubo para que aparezca un menú desplegable con una serie de opciones de vistas más comunes y seleccionar la deseada. Para volver al ángulo de visión predeterminado, basta con pulsar con el botón derecho sobre el cubo y seleccionar, dentro del menú desplegable, la opción Free. También existe la opción de cambiar entre Perspectiva e Isométrica, al pulsar sobre el texto debajo del Gizmo o pulsando con el botón izquierdo del ratón sobre el cuadrado. En la Figura 10 y en la Figura 11 se puede ver la misma escena pero con los dos tipos de vistas disponibles, se puede visualizar por tanto que la mayor diferencia existente entre ambas se da en que en la vista en perspectiva, al contrario que en la isométrica, se considera cierto ángulo en altura para mejorar la visualización.

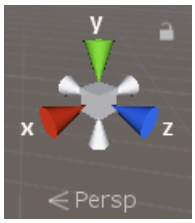

Figura 9. Gizmo.

Figura 10. Vista en Perspectiva.

Figura 11. Vista en Isométrica.

Para moverse por la escena se puede utilizar las flechas del teclado. Las flechas hacia arriba y hacia abajo mueven la cámara hacia adelante y hacia atrás en la dirección que se esté mirando. Y las flechas izquierda y derecha desplazan la vista hacia la izquierda y derecha respectivamente. Para moverse más rápido se puede mantener presionada la tecla Mayúscula con una flecha a la vez. Otra forma de moverse por la escena sería con las diferentes opciones que ofrece la barra de herramientas que se explicarán en el apartado 2.4.2.

Por otro lado existe la opción "*Flythrough*" o "volar a través", donde se puede recorrer en primera persona la escena, siendo esta una forma similar a como se verá en la simulación. Para entrar en este modo basta con pulsar y mantener presionado el botón derecho del ratón. Si se quisiese mover la vista, se usará el mouse y para moverse izquierda/derecha/adelante/atrás se usarán las teclas "W"," A"," S" y "D" respectivamente. También se usarán las teclas "Q" y "E" para moverse hacia arriba y hacia abajo. Al igual que se mencionaba anteriormente con las flechas, se podrá ir más rápido si se mantiene presionada la tecla Mayúscula. Este modo está diseñado para la Perspectiva, en el caso de que se estuviese en el modo Isométrico solo se conseguiría que la cámara orbitase. Tampoco está disponible por tanto en el modo 2D.

La ventana tiene una barra de control (ver Figura 12), con diferentes opciones que ayudan al control y la visualización de la escena.

Figura 12. Barra de control de la Ventana Escena.

La primera opción despliega un menú con el que se busca configurar el modo de visualización de texturas o la iluminación. En primer lugar se encuentra "*Shading Mode*" que se refiere al sombreado, donde te permite elegir tres opciones: "*Textured*", muestra las superficies con sus texturas visibles; "*Wireframe*", dibuja mallas con una representación de líneas; y "*Textured Wire*", muestra los meshes con textura y con un modo de líneas de dibujo superpuesto.

La siguiente opción se trata de "*Miscellaneous*" permite seleccionar diferentes opciones que influyen sobre todo en el color: "*Shadow cascades*", muestra cascadas de sombras de luz direccionales; "*Render Paths*", muestra la ruta de representación para cada GameObject usando un código de color (azul implica "*deferred shading*", verde, "*deferred lighting*", amarillo,"*forward rendering*" y el rojo "*vertex lit*"); "*Alpha Channel*", renderiza los colores con alpha, que se trata de un modo de tratar los colores; "*Overdraw*", renderiza los objetos como siluetas transparentes cuyos colores transparentes se acumulan haciendo más fácil identificar lugares donde un objeto se dibuja encima de otro; "*Mipmaps*", muestra tamaños ideales de texturas usando un código de colores (rojo indica que la textura es demasiado grande y azul que la textura puede aumentarse); y por último "*Texture Stream*", tiñe los GameObjects de verde, rojo o azul en función de su estado en el sistema.

Con "*Deferred*" se permite ver los elementos ("*Albedo*", "*Specular*", "*Smoothness*" y "*Normal*") del sombreado de manera aislada. En "*Global Illumination*" se pueden elegir opciones para visualizar el sistema de Iluminación: "*UV Charts*", "*Systems*", "*Albedo*", "*Emissive*", "*Irradiance*", "*Directionality*", "*Baked*", "*Clustering*" y "*Lit Clustering*".

Por último, con "*Material Validator*" se puede elegir entre "*Albedo*" y "*Metal Specular*". Esto permite verificar si los materiales que tengan base física cumplen los valores recomendados.

El siguiente interruptor en la Figura 12 se trata del interruptor 2D/3D, en el que se puede variar entre ambos modos. En el modo 2D, la cámara está orientada mirando hacia el eje z en dirección positiva, apuntado el eje x hacia la derecha y el eje y hacia arriba. El siguiente botón es el de luz, donde se puede encender o apagar la iluminación de la escena. La siguiente opción en la barra de control de la Ventana Escena es el del audio y de nuevo es un interruptor que enciende o apaga el sonido.

El pulsador que va a continuación, con forma de una imagen con una montaña, se trata del botón de efectos y permite habilitar o deshabilitar efectos de la ventana tales como "*Skybox*", el cual se refiere al fondo de la escena; "*Fog*", desvanecimiento gradual de la vista; "*Flares*", destellos en la vista; y "*Animated Materials*", que como el propio nombre indica define cuando aparecerán las animaciones. Además, el propio botón de efectos puede usarse como un interruptor para habilitar o deshabilitar todos los efectos a la vez.

La penúltima opción se trata del pulsador Gizmo, el cual despliega un menú con diferentes opciones de cómo se muestran los objetos, iconos y gizmos en la escena.

Por último hay un cuadro de búsqueda que permite rastrear elementos en la ventana por sus nombres y tipos. El conjunto de elementos que coinciden con el filtro de búsqueda también se muestra en la Ventana de Jerarquía, explicada en el apartado 2.4.4.

2.4.2 **Barra de herramientas**

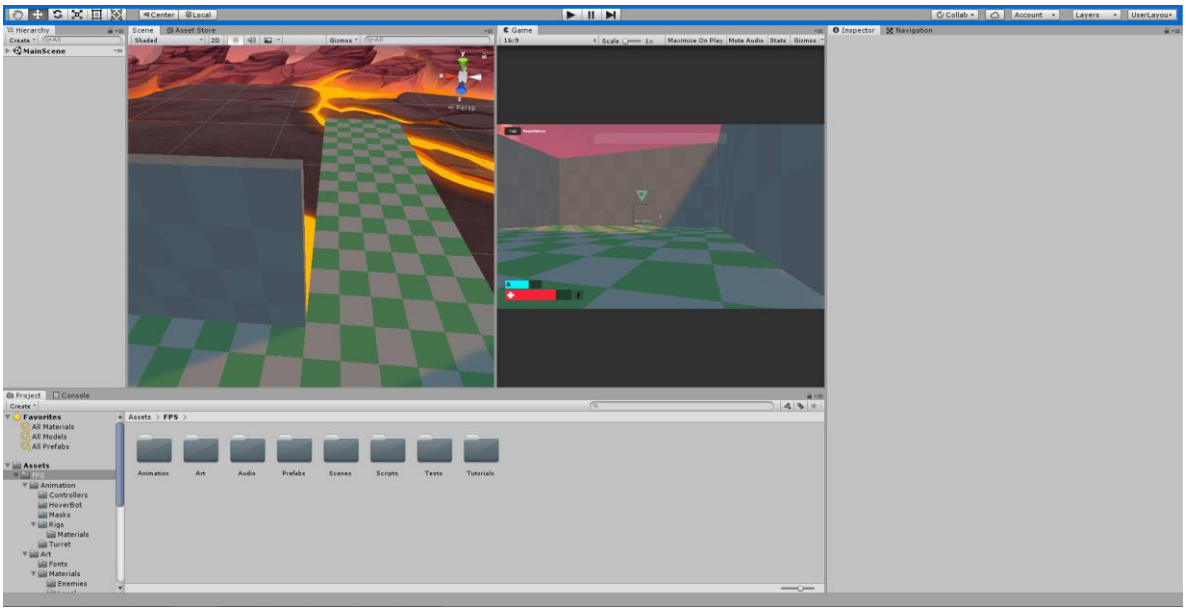

Figura 13. Barras de herramientas.

La barra de herramientas o "*Toolbar*", aparece marcada por un rectángulo en la Figura 13. Su uso principal es el de proporcionar acceso a las funciones de trabajo más esenciales de Unity. A la izquierda se encuentran las herramientas básicas para manipular la escena y los GameObjects que haya dentro de ella. En el centro, se encuentran los botones de reproducción. Y por último a la derecha se encuentran los botones que dan acceso a diferentes servicios de Unity y unos pulsadores que permiten configurar el diseño de la interfaz. A continuación se va a proceder a estudiar estos botones detenidamente.

Figura 14. Herramientas básicas de la Barra de Herramientas.

En la Figura 14 se pueden identificar las herramientas básicas que se han mencionado con anterioridad, las cuales se sitúan a la izquierda de la barra de herramientas. Si se observan las opciones que existen de herramientas desde la izquierda, se encontrará en primer lugar la herramienta "*Hand Tool*"; con ella se puede pulsar y arrastrar el ratón sobre la ventana para mover la cámara de la escena libremente. El acceso rápido de esta herramienta se hace con la tecla "Q".

Las tres herramientas que van a continuación son "*Translate*", "*Rotate*" y "*Scale*"; que se utilizan para trasladar, rotar y escalar los GameObject que se encuentren seleccionados. Cada uno se pueden operar con diversos accesos rápidos con las teclas: "W", "E" y "R" respectivamente. A la hora de realizar la operación aparecerá un Gizmo alrededor del GameObject seleccionado en el Scene View. Con el ratón, se podrá manipular el eje de cualquier Gizmo para cambiar el componente "*Transform*" del GameObject.

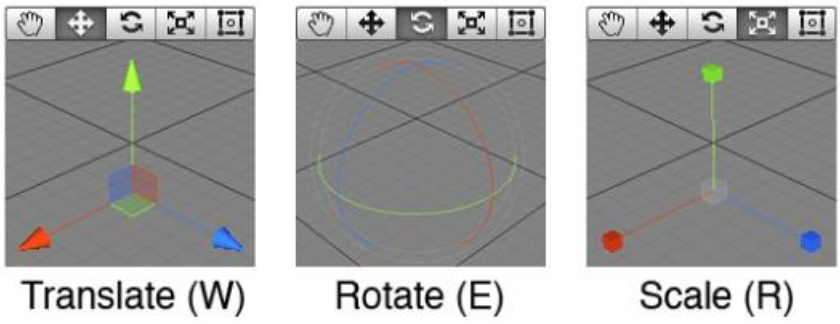

Figura 15. Gizmo herramientas "*Translate*", "*Rotate*" y "*Scale"*

En la Figura 15, se pueden observar los diferentes Gizmos que aparecen al elegir una de las tres herramientas mencionadas. En el caso de la herramienta trasladar, se puede observar tres cuadrados que pueden usarse para arrastrar el objeto hacia la dirección elegida. Esta operación se puede realizar con el ratón o manteniendo presionada la tecla shift mientras se pulsa y se arrastra el centro del Gizmo, el cual cambiará para ser un cuadrado plano. Esta acción permitirá mover el objeto alrededor del plano relativo a la manera en que la cámara de la Ventana de Escena (ver Figura 8) esté mirando.

Con "*Rotate*", que es la tercera herramienta en la Figura 14, se puede cambiar la rotación del objeto, pulsando y arrastrando los ejes del gizmo de la esfera metálica que aparece alrededor de él. Al igual que con la herramienta para trasladar, el último eje que fue modificado será marcado de amarillo y se permitirá ajustarse más haciendo click en el botón del medio del mouse y arrastrando. En el Gizmo se usarán los colores para poder relacionar los círculos con sus respectivos ejes, por lo que los círculos rojos, verde y azul, permitirán realizar una rotación alrededor de los ejes rojo, verde y azul respectivamente. Finalmente, el círculo más externo puede ser utilizado para girar el objeto alrededor del eje -Z de la Ventana de Escena; efectuando un giro alrededor del espacio de pantalla.

A continuación, en la Figura 14, se encuentra el botón "*Scale*". Esta herramienta de escala permite reescalar el objeto en cada uno de sus ejes pulsando y arrastrando el cubo dentro Gizmo del eje deseado. También existe la opción de aumentar el objeto en todos sus ejes simultáneamente, pulsando esta vez el cubo en el centro del Gizmo. Nuevamente, el último eje cambiado aparecerá en amarillo y puede ser ajustado arrastrándolo con el botón medio del ratón.

Dentro del modo 2D, el eje Z no puede ser cambiado en la escena usando los gizmos. No obstante, es útil para ciertas técnicas de scripting usar el eje Z para otros motivos; por lo que aunque sea un entorno bidireccional, se

puede establecer el eje Z en la Ventana Inspector (ver Figura 26).

La siguiente herramienta que se encuentra en la Figura 14 se trata de la herramienta "*Rect Transform*" que se trata de la contraparte del diseño 2D de la herramienta "*Transform*" (siguiente y última herramienta de la Figura 14). En concreto, "*Transform*" es una herramienta que representa un solo un punto al que se le puede modificar la posición, la rotación y la escala de manera simultánea. Por otro lado, con "*Rect Transform*" se representa un rectángulo donde un objeto puede ser colocado en su interior y modificarlo como tal. El acceso rápido para ambas herramientas es la tecla "T" para "*Rect Transform*" y la tecla "Y" para el caso de "*Transform*".

Una vez se han descrito las herramientas básicas centradas en el posicionamiento del objeto, se pasaran a explicar la palanca de visualización de Gizmo ("*Gizmo Display Toggles*") la cual se puede observar en la Figura 16.

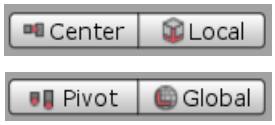

Figura 16. "*Gizmo Display Toggles*".

En estas opciones se definirá la ubicación de cualquier "*Transform Gizmo*". Como defecto, las opciones que saldrán serán las de "*Center*" y "*Local*". El interruptor "*Center*" está relacionado con la posición y ubicará el Gizmo en el centro de los límites dictados del objeto. Si se pulsase sobre él, aparecería "*Pivot*", asociada también a la posición, pero esta vez posicionará el Gizmo en el punto de pivote actual de un "*Mesh*". La opción "*Local*", está asociada a la rotación y mantendrá la rotación del Gizmo similar a la del objeto. En el caso que se pinchase sobre "*Local*", aparecerá la opción "*Global*", que sujetará el Gizmo a la orientación espacial del mundo virtual.

Figura 17. Botones de reproducción.

A continuación, en la barra de herramientas, se encuentran los botones de reproducción "*Play*", "*Stop*" y "*Step*" (ver Figura 17). Esto se usará en la Ventana del Juego, apartado 2.4.3, y se usará para reproducir, parar y pasar, respectivamente.

El siguiente botón en la barra de herramientas, la Figura 18, se trata de una herramienta que abre Unity Collaborate desde el menú desplegable de "*Collab*". Unity Collaborate es parte de Unity Teams, el cual permite colaborar, sincronizar y compartir proyectos a pequeños equipos en un entorno alojado en la nube. Gracias a esta opción se permite realizar un proyecto entre un grupo de personas independientemente de su ubicación.

Figura 18. Unity Collaborate.

Además, Collaborate mantiene un historial de las versiones del Proyecto que se publican, lo que permite restaurar archivos individuales o un proyecto completo.

Se pueden agregar miembros del equipo al proyecto, permitiendo trabajar síncronamente. Collaborate supervisa los cambios realizados por cada miembro del equipo y muestra una insignia en los archivos que han cambiado, pero que aún no se han publicado. Tras ver los cambios, se puede optar por revertir los cambios o publicarlos.

Para habilitar Collaborate se debe seleccionar la pestaña de "*Windows"*. Luego elegir "*General"* y finalmente "*Services"*. Se abrirá en la Ventana Inspector, explicada en el apartado 2.4.5, una pestaña de "*Services*", donde se tendrá que activar "*Collaborate service*".

Por último, para mantener activado Collaborate mientras se esté trabajando en el proyecto y esté notificando las modificaciones, se deberá de pulsar el botón "*Collab*", la Figura 18.

Figura 19. La nube.

La siguiente herramienta será el de la Nube (ver Figura 19). Esta herramienta permite acceder a otros servicios integrados que ofrece Unity.

Figura 20. Herramienta Account.

La penúltima herramienta en la barra se trata de la herramienta "*Account*" (ver Figura 20). Esta herramienta da el acceso al menú desplegable sobre el usuario.

Por último (ver Figura 21) está la herramienta de Layers y Layout. Con la primera se puede controlar que objetos aparecen en la Scene con un menú desplegable donde se pueden hacer invisibles o visibles las capas que se deseen. Por otro lado, con Layout se puede cambiar la disposición de las ventanas en el interfaz y luego guardar un nuevo diseño o cargar uno existente dentro del menú desplegable.

Figura 21. Botones Layers y Layout.

2.4.3 Ventana del Juego

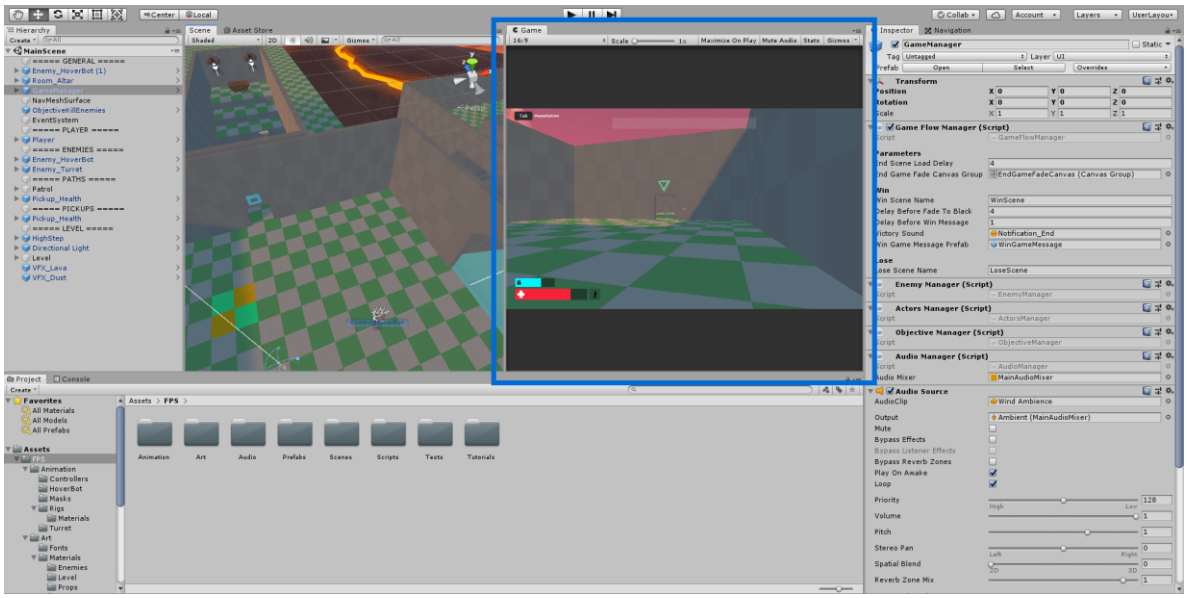

Figura 22. Ventana del Juego.

En esta ventana, señalada en la Figura 22, se visualizará la simulación tal y como lo verá el usuario, utilizando el GameObject "*Scene Camera*" para configurar el ángulo de visión de la simulación. Cuando se pulsa el botón de Reproducir en la barra de herramientas, da comienzo a la simulación. Cada vez que se entra en el "*Play mode*" o modo de reproducción, la interfaz del editor se vuelve oscura para recordarte que está activado dicho modo.

Figura 23. Barra de control de la ventana de Juego.

En la Figura 23 se pueden observar las diferentes opciones de configuración de la Ventana del Juego, que aparecen en una barra encima de la misma ventana. En primer lugar se encuentra el botón de "*Display*", este botón permite escoger entre una lista de cámaras, si existen varias cámaras. Por defecto, al crear un proyecto en Unity, se crea una cámara principal denominada "*Main Camera*" que se asocia al "*Display 1*". En el caso de que se quiera asignar a una cámara un "*Display*", se debe seleccionar la cámara y en la Ventana Inspector,

(ver Figura 26), asignarle el "*Display*" deseado en la opción "*Target Display*".

A continuación, en la barra de control se permite elegir el tamaño de la vista para que se ajuste mejor al monitor donde vaya a ser simulado. Esto se realizará con diferentes ratios o con la configuración que viene seleccionada por defecto al crear un proyecto, "*Free Aspect*".

"*Scale*" permite acercarse y examinar áreas de la pantalla del juego con más detalle; o al contrario, se puede alejar para ver la pantalla completa dónde la resolución de la pantalla sea mayor que el tamaño de la ventana del visto. También es posible realizar estas acciones con la rueda del ratón, si se ha posicionado previamente el cursor sobre la ventana de juego; pero esto solo será posible si el juego está parado o pausado.

La opción de "*Maximize on Play"* es una función que si se habilita, cuando se ejecute el modo de reproducción, aumentará, ocupando completamente la pantalla. Al igual que la anterior herramienta, "*Mute Audio*" puede ser habilitada o deshabilitada; pero en este caso en lo que se aplicará es en el audio cuando entre en el modo de reproducción.

Si se pulsase sobre "*Stats*", se abrirá una ventana superpuesta donde se mostrarán estadísticas del audio y gráficos permanentemente, aun estando dentro del modo de reproducción.

Por último, si se pulsa sobre Gizmos se dará visibilidad a los todos los Gizmos de los GameObjects. Además, permite diferentes opciones en un menú desplegable donde se puede elegir los Gizmos que se quieren ver u ocultar. Este menú es idéntico al de la Ventana Escena que ya fue explicado en el apartado 2.4.1.

2.4.4 **Ventana de Jerarquía**

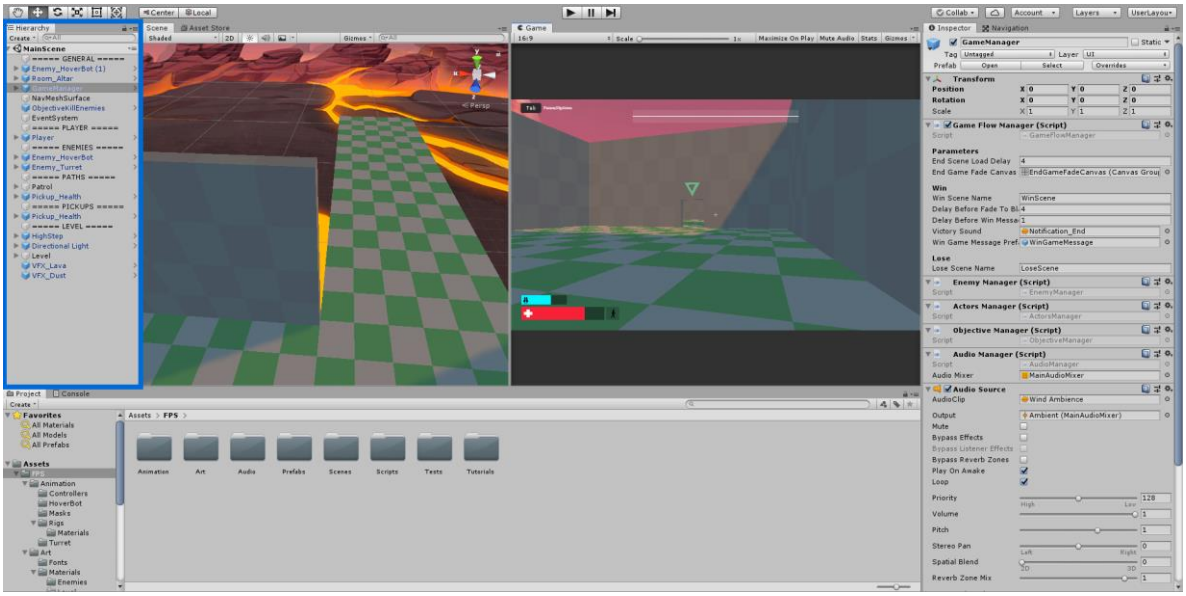

Figura 24. Ventana de Jerarquía.

En la ventana de Jerarquía, marcada en la Figura 24, se tienen todos los objetos de la escena actual. Cada elemento dentro de la Ventana Escena (ver Figura 8), tiene una entrada dentro de la lista de jerarquía, por lo que lo que ambas ventanas están vinculadas. Además, también se encuentra en gran parte vinculada a la Ventana Inspector (ver Figura 26) pues cuando se seleccione cualquier GameObject de la lista, aparecerá en la Ventana Inspector mostrando todas sus características.

La lista de jerarquía revela la estructura de cómo los GameObjects se unen entre sí. De forma predeterminada, la ventana ordena los objetos de manera que los que hayan sido creados más recientemente están en la parte inferior y los más antiguos, en la parte superior. Por otro lado, se puede reordenar los GameObjects arrastrándolos hacia arriba o abajo en la lista de jerarquía.

En Unity se usa el concepto denominado Parentesco o "*Parenting*", esto implica crear un GameObject al que se le denominará padre y crear otro objeto u objetos llamados hijos que estarán agrupados dentro del padre. Esto creará una estructura padre-hijo anidada. Para hacer que dos GameObject estén emparentados, se tendrá que arrastrar el que se desee que sea el hijo hasta el padre. Un hijo va a heredar todas las características y

componentes del padre, pudiendo ser modificado posteriormente.

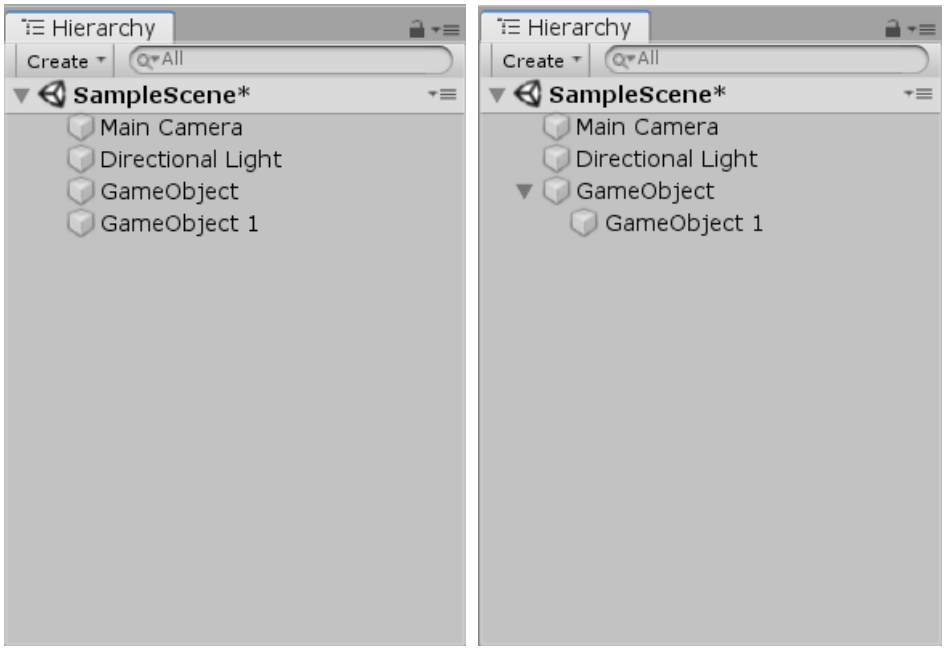

Figura 25. Objetos no emparentados y emparentados.

En la Figura 25 se puede observar dos objetos en los que en el caso de la izquierda no se encuentran emparentados; y por el contrario, en la derecha si se encuentran emparentados siendo GameObject el padre y GameObject 1 el hijo.

2.4.5 **Ventana Inspector**

La Ventana Inspector, marcada en la Figura 26, permite ver y editar todas las propiedades del GameObject seleccionado. Dependiendo del tipo de GameObject existirán diferentes tipos de propiedades, características de diseño y contenido; por lo que la ventana de Inspector cambiará cada vez que se seleccione un GameObject diferente.

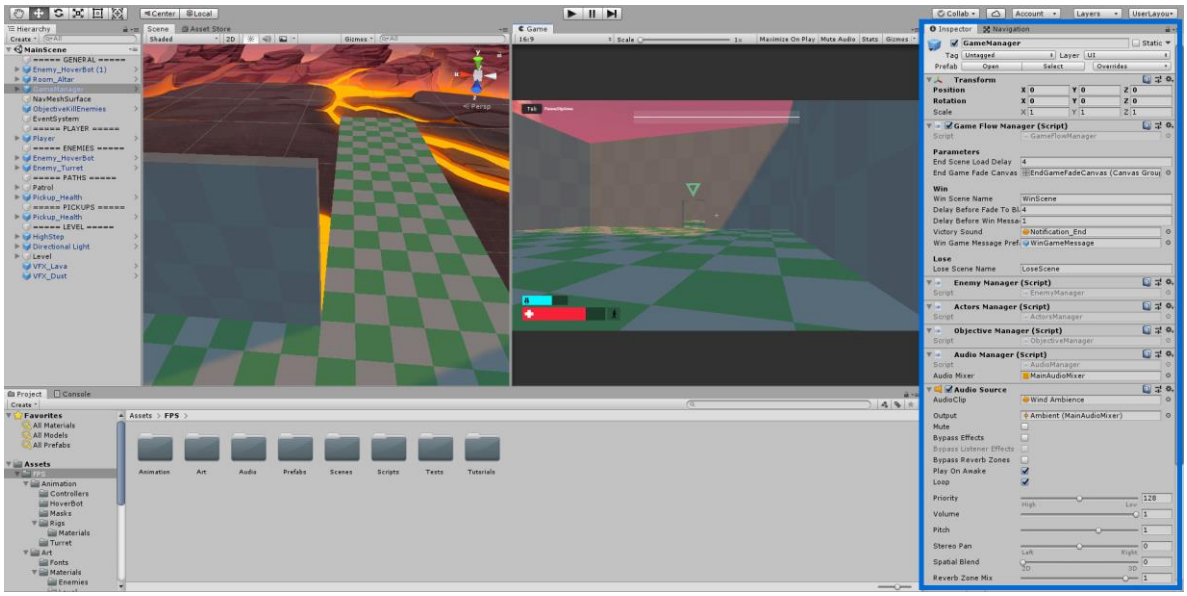

Figura 26. Ventana Inspector

Dentro de una Ventana Inspector de cualquier objeto, se encontrarán siempre la componente Transform. Esta componente detallará la posición, rotación y escala del GameObject. Las demás características o componentes

se irán añadiendo en función de la naturaleza del objeto o de las particularidades que se quisiese añadir. Por ejemplo, en el caso de la Figura 26, se pueden ver que existen varios script. Estos componentes suelen ser añadidos para generar comportamientos específicos mediante el uso de la programación y del que se hablará más adelante detenidamente en el apartado 3, y también estará "*Audio Source*" que aportará la reproducción de un audio al objeto.

En el caso de que se quiera añadir una componente al objeto, como puede ser un script o un material, bastaría con arrastrar lo que se quiera añadir desde su ubicación en la Ventana del Proyecto hasta la Ventana Inspector donde previamente se debería haber seleccionado el objeto. O seleccionar la opción "*Add component*" al final de la ventana y añadir la componente que se desee.

2.4.6 Ventana del Proyecto

La ventana del Proyecto (ver Figura 27) muestra todos los archivos del proyecto y permite navegar entre ellos. Cuando se crea un nuevo proyecto, esta ventana está abierta por defecto. En el caso de que no se encontrase, se puede encontrar en "*Window"*, seleccionando "*General"* y pulsando sobre "*Project"*. Otra opción es pulsar la tecla "Ctrl+9".

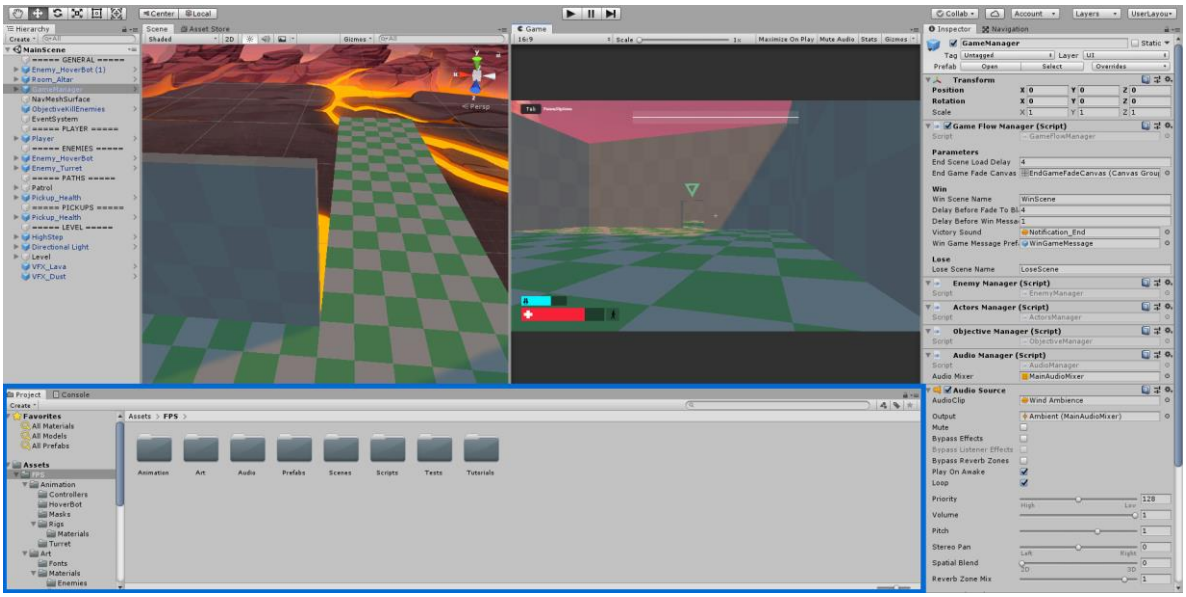

Figura 27. Ventana del Proyecto.

A la izquierda del panel se muestra la estructura de carpetas del Proyecto. Cuando se selecciona una carpeta de la lista, Unity muestra su contenido en el panel a la derecha. Para expandir o contraer el contenido de la carpeta, se puede pulsar sobre el triángulo a la izquierda del nombre de la propia carpeta, mostrando así los propios archivos o carpetas anidadas que tenga.

Los archivos individuales se muestran en el panel de la derecha como iconos independientes que muestran su tipología. Para cambiar el tamaño de los iconos, se puede usar el control deslizante en la parte inferior del panel; serán reemplazados por una vista de lista de archivos si el control se mueve hacia el extremo de la izquierda.

Con la Ventana del Proyecto viene asociada una barra de herramientas que podemos observar en la Figura 28.

Figura 28. Barra de herramientas de la Ventana del Proyecto.

En primer lugar se encuentra la opción de crear menú, el cual permite agregar nuevos assets y subcarpetas en la ubicación que se encuentre. A continuación, hay una barra que permite buscar un archivo en concreto dentro del proyecto. Si se escribe sobre la barra más de un término de búsqueda se estrechará la búsqueda. La siguiente opción en la barra de herramientas te permite concretar la búsqueda a un tipo concreto de archivo (ver Figura 29). Otra forma de concretar aún más la búsqueda es con el siguiente botón el cual permite realizar la búsqueda en función a su etiqueta. Dado que el número de etiquetas puede ser muy grande, el menú de

etiquetas tiene su propia caja de búsqueda (ver Figura 30). El navegador dispone de una función de búsqueda potente que es especialmente útil en el momento de ubicar archivos en grandes proyectos que conllevan una gran cantidad de activos.

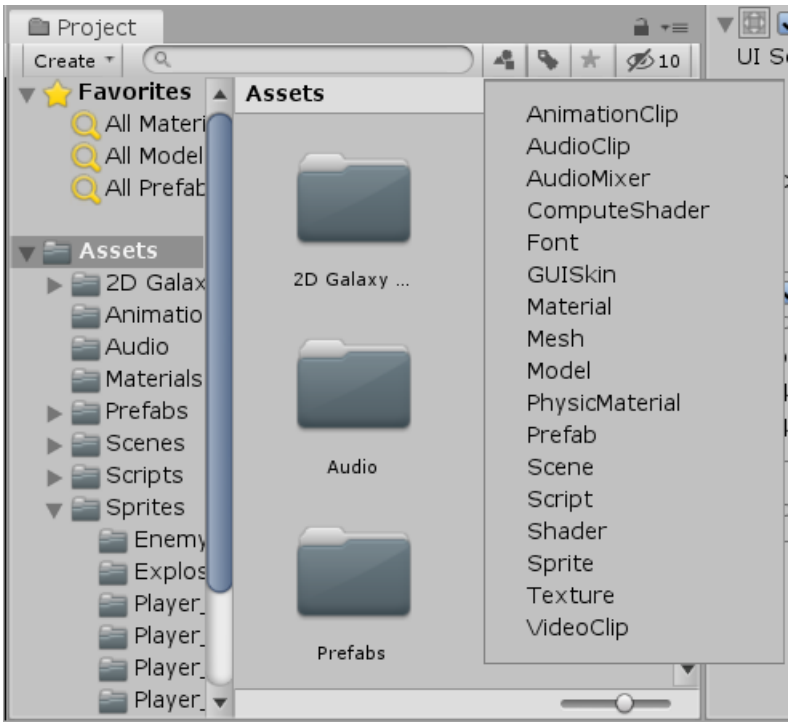

Figura 29. Menú desplegado de archivos.

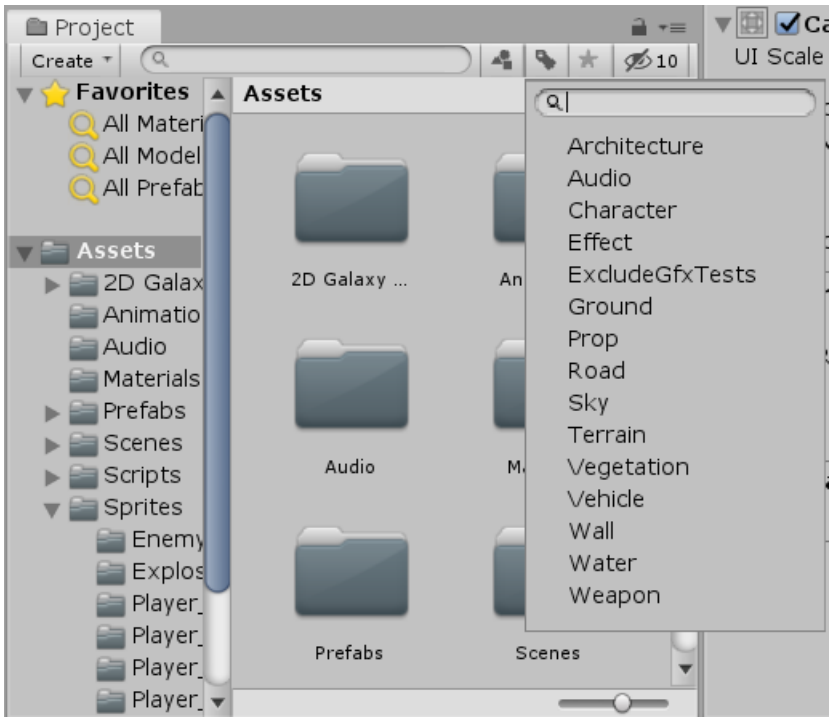

Figura 30. Menú desplegado etiquetas.

La opción de la estrella permite marcar una sección como favorita, esto será favorable para las secciones que más se usen ya que tendrá un fácil acceso dentro de la lista de carpetas a la izquierda de la Ventana del Proyecto. Por último, el ojo tachado actúa como interruptor para al alternar la visibilidad de los paquetes en la ventana.

Al buscar en el navegador del proyecto, se permite la opción de búsqueda en la "*Asset Store*" de Unity. Si se

busca algo, se puede seleccionar el botón de "*Asset Store*", tal como se puede ver en la Figura 31. Al pulsar sobre él se mostrarán todos los elementos gratuitos y de pago de la tienda que coincidan con su búsqueda. Buscando por tipo y etiqueta funciona de la misma manera que para un proyecto de Unity.

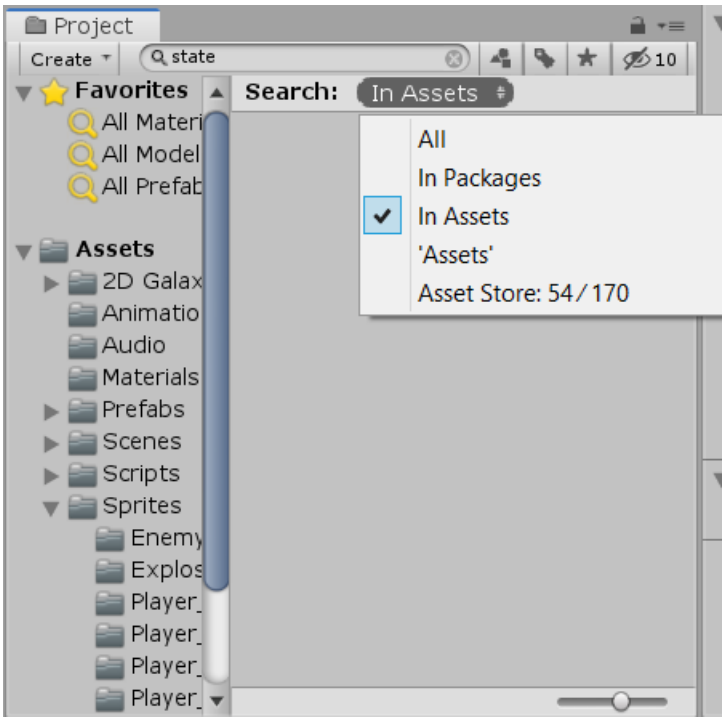

Figura 31. Opción de busqueda en la Asset Store.

2.4.7 **Consola**

En la consola (ver Figura 32) se muestran los errores, advertencias y otros mensajes generados. También se pueden mostrar mensajes generados por el propio usuario con el uso de scripts añadiendo al código "*Debug.Log*", "*Debug.LogWarning*" o "*Debug.LogError*" seguido del mensaje.

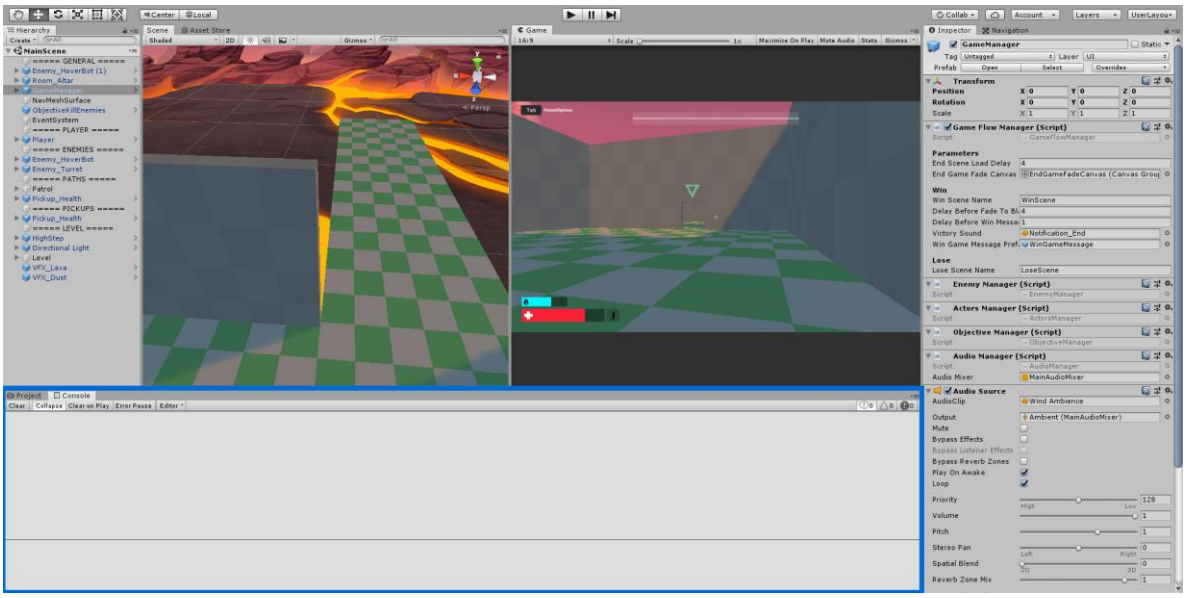

Figura 32. Consola.

La ventana permite opciones para controlar cuantos mensajes se desean que se visualicen, abrir archivos y opciones de seguimiento de la pila. También tiene una barra de control para gestionar los mensajes, la cual se puede visualizar mejor en la Figura 33.

Figura 33. Barra de control de la Consola.

En esta barra, la opción "*Clear*" permite eliminar los mensajes generados por el código pero mantiene los errores. "*Collapse*" muestra en caso de que haya un error repetido, el primero de ellos. La opción "*Clear on Play*" limpia la consola automáticamente antes de entrar en el modo de reproducción. Con "*Clear on Build*" se limpia la consola automáticamente cada vez que se compila el Proyecto. Con "*Error Pause*" se pausa la simulación cada vez que ocurre un Debug.LogError, esto es útil en el caso de que se quiera especificar el punto en el que deja de funcionar. "*Editor*" permite, en el caso que se esté conectado a un puerto remoto, mostrar el registro de Unity Player local en lugar del registro de compilación remota. La barra de búsqueda, permite buscar entre los mensajes con una palabra específica, como se especifica en la Figura 34.

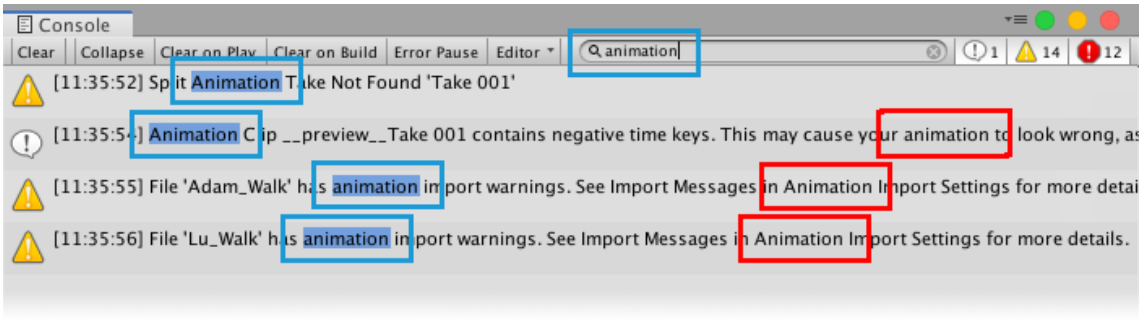

Figura 34. Búsqueda dentro de la Consola.

Por último los tres contadores del final muestran el número de mensajes, advertencias y errores detectados respectivamente. También, si se pulsa, funciona como filtro y aparecerán únicamente los seleccionados en la pantalla de consola.

De forma predeterminada, la consola muestra diez líneas, lo cual a su vez es lo máximo. Para cambiar esta configuración se debe pulsar sobre las tres rayas que hay en la zona superior derecha de la ventana de consola, como aparece en la Figura 35, y pulsar sobre "*Log Entry*" donde se podrá elegir el número de líneas que se prefiera.

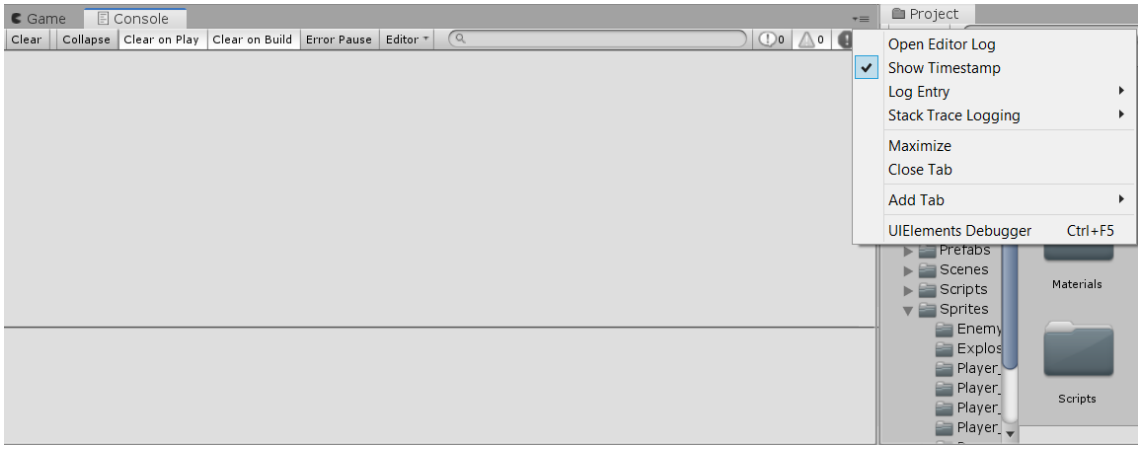

Figura 35. Menú desplegable para la configuración de la Consola.

Unity puede incluir información detallada sobre el seguimiento de la pila cuando imprime mensajes en la consola o en el archivo de registro. Esto es útil cuando un mensaje de error no es muy claro. La consola puede dar información sobre el seguimiento de pila para el código administrado y código no administrado. El código administrado son los scripts de C# que se ejecutan en Unity y los no administrados engloba al código de motor propio de Unity; es decir, código que se ejecuta directamente en la máquina y no se puede acceder a él a menos que tengo el código fuente original del binario propio.

El seguimiento de pila por tanto es más propio para código administrativo, ya que ayuda a depurar errores; sin embargo, también puede usarse en el administrativo para determinar si el error fue del código propio o del motor de Unity.

Para configurar con detalle el seguimiento de pila, al igual que cuando se cambió la longitud en la Figura 35, se seleccionará "*Stack Trace Logging*" y se incluirá lo que se desee a la pila.

2.4.8 Ventana de Animación

La ventana de animación (ver Figura 36) permite crear y modificar los clips de animación directamente dentro de Unity. Junto a las animaciones de movimiento, permite animar materiales u objetos con un clip y seleccionar dicha animación como un evento, las cuales se pueden especificar en qué momento debe suceder a lo largo de la línea del tiempo.

Para abrir la ventana de animación basta con pulsar dos veces con el botón derecho sobre una animación ya existente, o abrir una nueva ventana pulsando sobre "*Window*," y eligiendo "*Animation"*. El comando rápido de teclado para esta opción es "Ctrl+6".

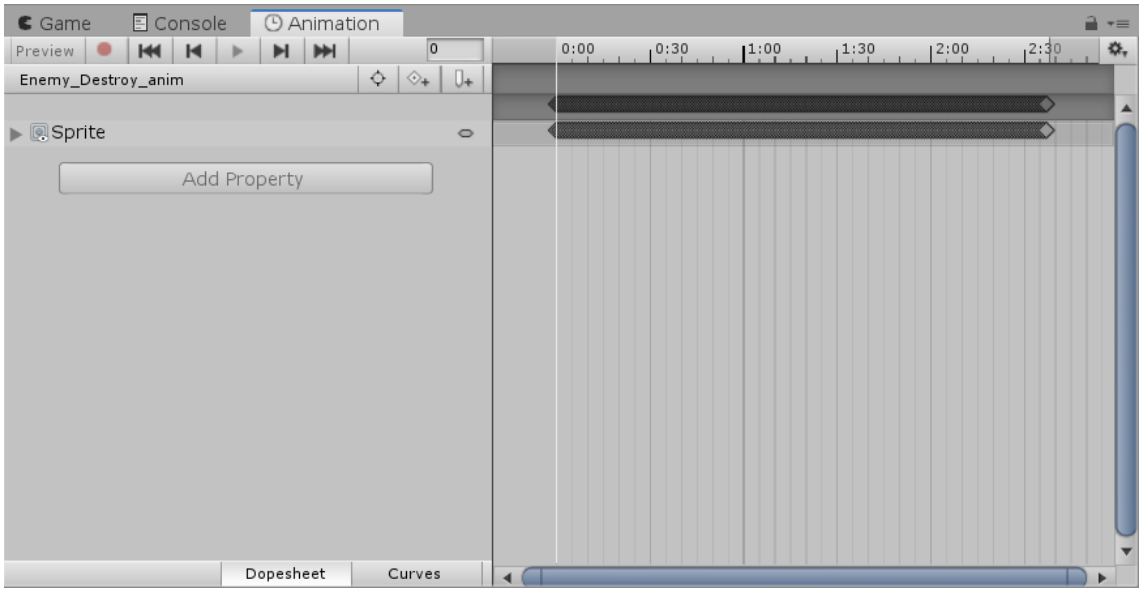

Figura 36. Ventana de Animación.

En la Figura 36, se puede observar que a la izquierda se tiene la animación usada actualmente por el GameObject. En el caso de que se crearan hijos a partir de este objeto al que se tiene asociado el clip, también mantendrían esta característica. En la parte superior del cuadro derecho se encuentra una lista de propiedades, y da la opción al final de añadir una nueva. Cada propiedad se puede plegar y desplegar para revelar los valores exactos registrados en cada fotograma clase. Los campos de valor muestran el valor interpolado si el cabezal de reproducción, la línea blanca sobre la ventana de la derecha, está entre fotogramas clave. A la derecha se encuentra un cronograma para el clip. Los fotogramas clave vendrán marcados en la línea del tiempo como rombos. El cronograma se puede visualizar de dos maneras distintas, que se puede seleccionar justo en la parte inferior de la ventana izquierda: "*Dopesheet*", que es la que se puede visualizar en la Figura 36, y "*Curves*" que permite visualizarlo como una curva.

En la parte superior izquierda también se puede ver una barra de controles de reproducción y navegación de cuadros. De la izquierda a la derecha los controles son el interruptor de "*Preview*", el interruptor de grabación el cual siempre que esté encendido lo estará también el de "*Preview*", el botón para volver al comienzo del clip, el botón para volver al fotograma clave anterior, interruptor de reproducir, el botón para volver al fotograma clave siguiente y el botón para ir al final del clip.

En el caso de que se esté creando una nueva animación, aparecerá un botón en el centro de la Ventana de animación que permitirá guardar el clip como un archivo dentro de las carpetas de archivos del Proyecto. Para poder asignar una animación a un GameObject será necesario que este tenga un componente "*Animator*" dentro de su Ventana Inspector (ver Figura 26) tal y como se muestra en la Figura 37. Para poder añadirle la animación se le tiene que agregar un "*Animator Controller*", en el caso que se está visualizando en la Figura 37 se trata de "*Enemy*", donde se le pueda asignar un Clip.

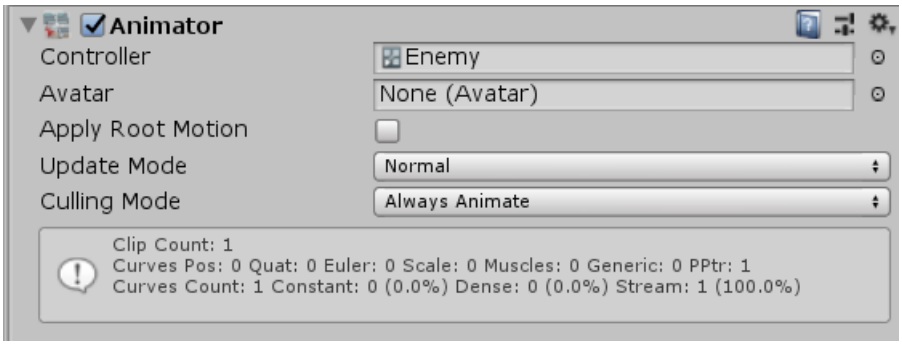

Figura 37. Animator.

Si se pulsa dos veces sobre "*Enemy*", se abrirá la Ventana de Animator (ver Figura 38). En esta ventana se muestra el contenido del "*Animator Controller*" o el control de la animación, y tiene dos secciones principales: el área principal de diseño en cuadrícula y el panel izquierdo de capas y parámetros.

La zona de la derecha se usa para crear, conectar y organizar los estados. Para crear un nuevo estado basta con pulsar con el botón derecho sobre el fondo y seleccionar "*Create state*". A consecuencia, se desplegará un menú donde se elegirá el tipo de estado nuevo a crear. Para reorganizar el diseño se debe pulsar y arrastrar los nodos, pudiendo reorganizar eficientemente el conjunto de estados, el cual forman la denominada máquina de estados.

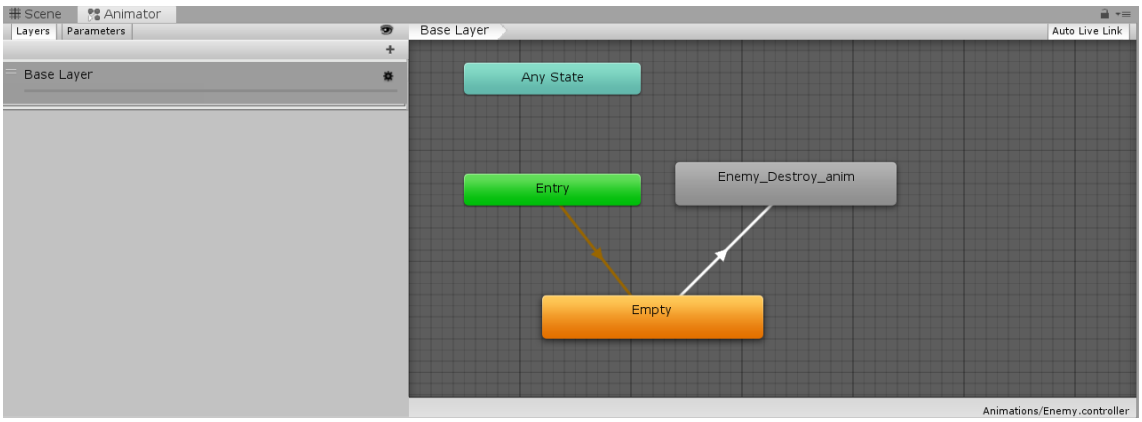

Figura 38. Ventana de Animator.

En el panel de la izquierda se puede modificar los parámetros para que actúen sobre el controlador. Estas variables son las que acaban definiendo como actúan todos los estados. Para agregar un parámetro se debe pulsar sobre el icono más y seleccionar el tipo de parámetro en el menú desplegable.

3 SCRIPTS

Un lenguaje scripting es un tipo de lenguaje diseñado para integrarse y comunicarse con otros lenguajes de programación. La principal diferencia entre este tipo de lenguaje respecto al resto es su modo de compilación; pues aunque lo necesite, estos lenguajes son capaces de interpretar sin compilarse[4]. Cuando se dice que interpreta conlleva que se necesita un intérprete que haga de intermediario para traducir el código para que la máquina lo entienda. Estos scripts serán necesarios en casi todas las aplicaciones que se quiera hacer en Unity desde crear efectos, controlar los movimientos o comportamientos.

El comportamiento de los GameObjects está controlado por los componentes que están vinculados a los mismos. Unity permite crear dentro del objeto un componente denominado script y a su vez, acceder a él por medio de un editor de texto, para programarlo según sea el objetivo deseado. Unity admite el lenguaje de programación C#, un lenguaje estándar de la industria similar a Java o C++.[5]

Existe una extensión de manuales [6] que, en función de la versión, explican cómo usar las diferentes opciones que hay en visual Studio e informan las novedades existentes en la versión actualizada respecto a la anterior. En esta recreación se ha trabajado con la versión de 2019.

3.1 Visual Studio

Visual Studio es un entorno de desarrollo integrado capaz de editar, depurar y compilar código y a su vez es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic.NET, F#, Java, Phyton, Ruby y PHP[7]. Por defecto, Unity usa Visual Studio y por ello cuando se hace doble click sobre un script este es el editor de texto que se ejecuta. Sin embargo, se podría elegir otro editor en la opción "*External Tools*" dentro de las preferencias de Unity.

Un script de Visual Studio hace sus conexiones con el funcionamiento interno de Unity al implementar una clase que deriva desde la clase integrada llamada "*MonoBehaviour*". Una clase es como un tipo de plano para crear una nueva componente que quede vinculada a un GameObject. Cada vez que se añada este script a un GameObject, este crea una instancia del objeto definido por el plano. El nombre de la clase se toma del nombre del propio script, y será importante que ambos coincidan para permitir ser adjuntando a una GameObject[5].

Figura 39. Visual Studio.

3.2 Creación de Scripts

Para crear un nuevo script basta con pulsar con el botón derecho sobre la carpeta de la Ventana del Proyecto (ver Figura 27) donde se quiera crear. De todas las opciones que aparecerán, se deberá pulsar sobre "*Create*", eligiendo en este caso "*C# Scripts"* en el menú desplegable. Esta acción generará un archivo C#, como aparece en la Figura 40, dentro del fichero previamente elegido. Para poner el nombre que se desee bastará con pulsar sobre el título una vez, en el caso de este ejemplo se trata de "*MainMenu*".

Figura 40. Icono de un script.

Otra forma de crear un script sería desde la Ventana Inspector (ver Figura 26). Como se ha comentado anteriormente, existe la posibilidad de crear y vincular un script para el GameObject desde el mismo objeto. Para conseguirlo, dentro de la Ventana Inspector y con el objeto en concreto seleccionado, se deberá pulsar sobre "*Add Component*" y buscar script. Este archivo estará automáticamente asociado al GameObject y también tendrá el mismo nombre.

Una vez se ha creado un script, si se hace doble click sobre él se abrirá el editor de texto correspondiente, siendo Visual Studio el que se ejecuta en Unity por defecto. En la Figura 41 se puede observar la estructura inicial de un script y los componentes que vienen añadidos por defecto. Lo primero que aparece en el recién creado script es el espacio destinado a las librerías. Inicialmente habrá dos librerías: "*UnityEngine*", "*System.Collections*" y "*System.Collections.Generic*", que cuentan con funciones básicas. También, desde el principio, el script cuenta con la clase "*MonoBehaviour*", necesaria para que el script se pueda comunicar con Unity, la cual tendrá por defecto el propio nombre del script. Como se ha explicado con anterioridad, para que el script pueda funcionar como componente de otro objeto debe tener el mismo nombre de clase que el objeto para que se puedan vincular.

Por defecto, el script se creará con dos funciones: la función "*Update*" y la función "*Start*". La función "*Update*" se ejecutará cada lapso de tiempo. Esta función es especialmente útil para poder implementar acciones respuesta a entradas. Por el contrario, la función "*Start*" solo se llamará una vez y será antes de que la simulación comience, por lo que abarcará todo lo relacionado con la inicialización.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Robot : MonoBehaviour
6   {
7       // Start is called before the first frame update
8       void Start()
9       {
10
11      }
12
13      // Update is called once per frame
14      void Update()
15      {
16
17      }
18  }
```

Figura 41. Contenido inicial de un script.

3.3 Vincular un Script a un GameObject

Para vincular un script a un GameObject será necesario arrastrar el mismo archivo C# a la Ventana Inspector (ver Figura 26) del objeto que previamente se deberá haber seleccionado. Una vez se haya arrastrado, deberá salir una pestaña como la Figura 42, donde se muestra que se ha vinculado con éxito al script. Esta pestaña puede tener más usos pues existen variables que pueden ser visualizadas y hasta modificadas desde este apartado durante la simulación.

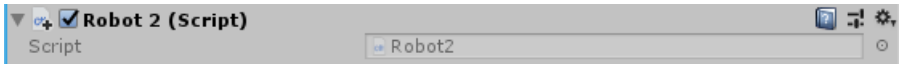

Figura 42. Componente Script dentro de la Ventana Inspector.

3.4 Variables en los Scripts

Dentro del propio script se utilizarán variables para configurar el comportamiento para nuestro GameObject. Habrá dos tipos de variables: públicas y privadas. Las variables privadas son las que pueden llegar a ser modificadas en la ventana Inspector, como se ha mencionado en el apartado 3.3, incluso mientras se esté ejecutando la simulación. Y las privadas solo podrán ser modificadas dentro del propio script.

3.5 Comunicación entre Scripts

Durante la simulación, se necesitará establecer comunicación entre diferentes scripts. Para ello, se utilizará la función "*GetComponent*", la cual permitirá acceder a la componente script de un objeto. Pero antes de poder acceder a dicha componente, será necesario localizar el propio GameObject. El medio que se utilizará para poder encontrarlo serán los "*Tags*" o etiquetas. Estas previamente habrán sido configuradas dentro de la Ventana Inspector (ver Figura 26) para que cada objeto tenga una etiqueta propia. Para entender mejor este concepto se puede ver en la Figura 43 como el objeto Robot6 tiene la etiqueta Robot6 en la sección "*Tag*". Por tanto, y volviendo al script que quería localizar otro, se utilizará "*FindWithTag*" para localizar el objeto con su etiqueta correspondiente, conseguirá acceso a la componente con "*GetComponent*" y podrá finalmente modificar o comunicarse con el script del objeto.

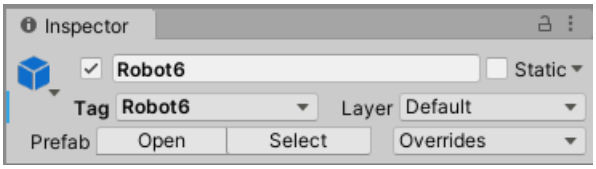

Figura 43. Ventana Inspector del Robot número 6.

Para conocer más aspectos relacionados con la programación scripting en Unity, se puede consultar de nuevo el manual mencionado en el apartado 2.1. [2]

4 COMUNICACIÓN

En la recreación, será necesario establecer una comunicación entre Matlab y Visual Studio. Por un lado, Matlab es un programa de cálculo que cuenta con herramientas para diseñar el movimiento de los vehículos no tripulados, por lo que desde este programa se podrían calcular las posiciones y exportarlas como una matriz de cinco filas, siendo estas respectivamente: coordenada x, coordenada y, coordenada z, tiempo y batería. Esta matriz, sería la información que se envía a Visual Studio para ejecutar la recreación.

4.1 Comunicación Matlab-Unity

La comunicación se realizará mediante protocolo TCP/IP, el cual permite el intercambio de datagramas a través de la red sin que se haya establecido previamente una conexión. Este protocolo permite que las aplicaciones puedan por tanto conectarse con garantías independientemente de las capas inferiores del modelo TCP/IP. Otra característica importante de este protocolo es que permite que la información viaje desde un origen hasta un destino y que los datos lleguen en orden, es decir, en el mismo orden que fueron emitidos; algo importante debido a la cantidad de datos matriciales que se deberá mandar en la simulación.[8]

Para realizar la comunicación, se utilizarán los denominados "*listener*" y "*socket*". Estos últimos en concreto permiten a los programas intercambiar datos aun teniendo lenguajes distintos. Este "*socket*" se tiene que abrir desde el programa donde se quiera realizar el envío de datos, una vez abierto permanecerá en ese estado hasta que se lo comunique el mismo programa u ocurra un error. Por el otro lado, el "*listener*" esperará desde el otro programa a escuchar la información. [9]

4.2 Fases de comunicación

La comunicación bidireccional entre ambos programas se realizará en dos fases:

1. Visual Studio como servidor y Matlab como cliente: en esta fase Matlab mandará la información a Visual Studio.

   Como se puede visualizar en la Figura 44, esta fase se refiere a los dos primeros pasos. La comunicación comenzará en Matlab, donde se procederá a enviar los datos a Visual Studio por la variable previamente creada "*tcpipClient*". En Visual Studio, la variable "*listener*" estará escuchando hasta que reciba los datos de Matlab.

2. Matlab como servidor y Visual Studio como cliente: en esta fase Visual Studio mandará un mensaje en el que confirma haber recibido la información.

   Esta fase comienza una vez recibido los datos, mandará por "*mySocket*" un mensaje a Matlab para confirmar que llegó el mensaje y procederá con la simulación correspondiente. Por otro lado, Matlab, tendrá la variable "*tcpipServer*" escuchando hasta recibir el mensaje. Una vez recibido, Matlab acabará.

El mensaje que intercambiarán los programas serán cadenas, por lo que si se quisiera mandar la matriz de cinco filas por el número de columnas que serían el número de posiciones totales; se necesitaría crear un string donde añadir toda la información unida. Para facilitar el tratamiento posterior, se añadirán comandos para distinguir entre matrices de los diferentes robots, entre las diferentes filas y también entre los propios componentes.

En la Figura 44 también se han añadido debajo de cada variable los comandos que han sido necesarios para declarar cada una en su respectivo entorno.

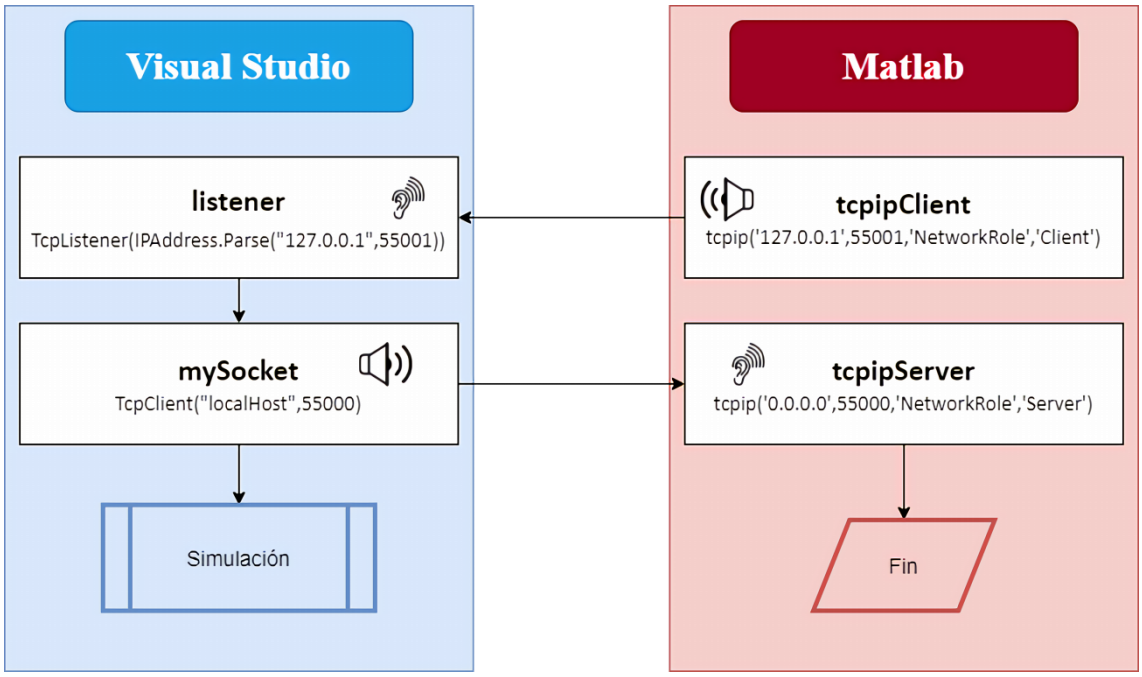

Figura 44. Esquema de la comunicación entre programas.

4.2.1 Visual Studio como servidor y Matlab como cliente

En primer lugar, en el entorno de Visual Studio, será necesario añadir nuevas librerías especializadas en la comunicación. Una vez se tienen las bibliotecas adecuadas, se podrá declarar la variable "*listener*" que será la encargada, como su nombre lo indica, de actuar como "*listener*" quedando a la espera de recibir un mensaje del cliente por la red TCP. A la hora de crear esta variable será necesario añadirle información tal como la dirección IP local y el puerto en el cual se escucharán los intentos de conexión entrantes. Una vez creada se deberá activar y esta quedará en escucha de información, estado que no terminará hasta que o reciba información o halla algún error. Debido a que Visual Studio es el que espera inicialmente; será necesario que sea ejecutado previamente a Matlab ya que podría suceder que este último tratara de comunicarse antes de que Visual Studio esté a la escucha.

En Matlab, también se necesitará crear una variable pero esta vez del tipo "*socket*", que corresponde a "*tcpipClient*" en el esquema, a la que se añadirá una dirección IP local, el puerto y también será importante señalar que en esta variable Matlab está trabajando como cliente, por ello se añade 'Client'. Una vez se tiene la variable declarada, se puede abrir el puerto, mandar la información y cerrarlo. Si Visual Studio está a la escucha, la información se mandará correctamente. El "*socket*" de Matlab se cerrará una vez mandado la información. En el caso de que haya un fallo de conexión, mostrará un error en la ventana de comandos en el que advierta de la incomunicación y cerrará el "*socket*". Una vez se ha mandado la información como string a Visual Studio, este lo recibirá y la guardará.

4.2.2 Matlab como servidor y Visual Studio como cliente

En este caso, Visual Studio será el que envíe un mensaje a Matlab. Para ello en primer lugar, Matlab deberá crear una variable, "*tcpipServer*", como se hizo en el apartado 4.2.1, solo que esta vez será de tipo "*listener*" ya que esta vez trabajará como un servidor. Esta variable, se deberá abrir y dejarla en espera a recibir información desde el puerto indicado en su declaración. Por otro lado, en Visual Studio, se deberá crear una variable tipo "*socket*", denominada en el esquema como "*mySocket*", que se conectará a un puerto especificado y que deberá ser el mismo que al que se conectará el "*listener*" de Matlab. Se enviará el mensaje correspondiente, que Matlab recibirá en forma a su vez de cadena.

Una vez se haya acabado la comunicación, Matlab finalizará y Visual Studio comenzará con el procesamiento del mensaje y posteriormente, con la recreación.

5 PLANTA TERMOSOLAR

Una Planta Termosolar es una tecnología de generación de energía mediante fuentes renovables, en concreto, a partir de la radiación solar. El funcionamiento de una central termosolar es similar al de una central térmica, pero en lugar de usar carbón o gas se utiliza la energía del sol. Los rayos solares se concentran mediante espejos en un receptor que alcanza temperaturas de hasta 1000 ºC. Este calor se usa para calentar un fluido, conocido como HTF ("*Heat Transfer Fluid*"), y generar vapor, que mueve una turbina y produce electricidad. Aunque las primeras centrales sólo podían operar durante las horas de irradiación solar, hoy en día es posible almacenar el calor para producir energía también por la noche.[10]

5.1 Tipos de Plantas Termosolares

Actualmente, existen cuatro tipos de plantas termosolares: de torre, de reflectores lineales Fresnel, de disco parabólico y de cilindro parabólico. La diferencia entre ellas radica en cómo se concentra la energía del sol.

5.1.1 Planta de torre

Estas plantas están formadas por un campo de helióstatos o espejos móviles que se orientan según la posición del sol. Los helióstatos tienen la función de captar la radiación solar y dirigirla hacia el receptor, el cual transfiere el calor recibido a un fluido de trabajo, que puede ser agua, sales fundidas, etc. La torre sirve de soporte al receptor.[11]

Figura 45. Planta Termosolar de torre.[5]

5.1.2 Planta de disco parabólico

Las plantas de disco parabólico o disco Stirling usan un concentrador solar de alta reflectividad, un receptor solar de cavidad y un motor Stirling que se acopla a un alternador. Su funcionamiento consiste en calentar un fluido localizado en el receptor hasta una temperatura entorno a los 750 ºC. Para que tenga un óptimo funcionamiento, se necesitará añadir mecanismos suplementarios para que pueda seguir la posición del sol en dos ejes. [11]

[5] Tomado de: https://www.diariorenovables.com/2016/06/la-planta-termosolar-mas-grande-y-barata-del-mundo.html

Figura 46. Planta de disco parabólico.[6]

5.1.3 Planta de reflectores lineales Fresnel

Estas plantas tienen espejos lineales orientables que concentran los rayos sobre tubos absorbedores situados en la parte superior. Los reflectores lineales se disponen en dirección Norte-Sur haciendo un seguimiento completo en un solo eje a lo largo del día. [11]

Figura 47. Planta de reflectores lineales Fresnel.[7]

5.1.4 Planta de cilindro parabólico

Estas plantas funcionan siguiendo el sol y concentrando los rayos solares en unos tubos receptores de alta eficacia térmica, que se localizan en la línea focal de los cilindros. Dentro de estos tubos, circula un fluido transmisor de calor que es calentado aproximadamente a 400 ºC y bombeado a través de una serie de intercambiadores de calor para producir vapor sobrecalentado. El calor presente en este vapor, pasará a ser energía mecánica en una turbina de vapor convencional y posteriormente en energía eléctrica por medio de un generador. [11]

[6] Tomado de: https://energiaunam.wordpress.com/2010/03/04/energia-solar-termica-disco-stirling/

[7] Tomado de: https://www.evwind.com/2013/02/18/murcia-ya-cuenta-con-900-megavatios-de-energias-renovables-eolica-termosolar-y-energia-solar-fotovoltaica/

Figura 48. Planta cilindro parabólica.[8]

5.2 Elementos de la planta cilindro parabólica

Existen principalmente tres configuraciones para una planta cilindro parabólico. La configuración más simple produce electricidad solamente durante las horas del sol. Las otras dos configuraciones: ciclo combinado-solar y tecnología de sales fundidas añaden funciones que permiten mejorar la capacidad para mejorar el suministro de la electricidad de forma que pueda ser producida como la red eléctrica lo necesite.[12]

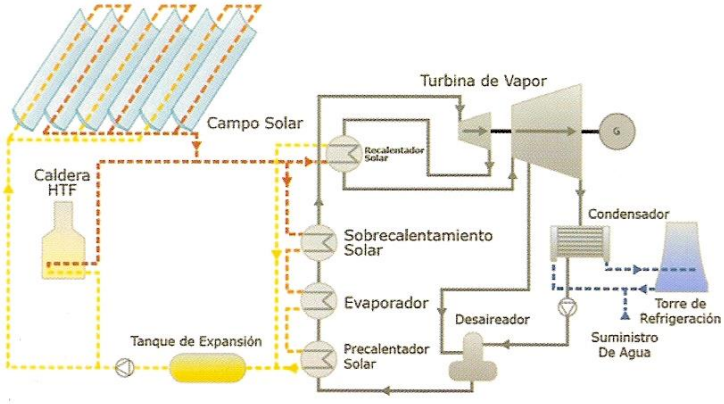

Figura 49. Esquema de una planta cilindro parabólica con la configuración más simple.[9]

La planta con tecnología de sales fundidas es la que se visualizará en la recreación de Unity, por lo que vamos a estudiarla con más detenimiento. El sistema está dividido por zonas según su funcionalidad: el campo solar, el sistema HTF, el ciclo de potencia, los sistemas auxiliares y los sistemas eléctricos.

5.2.1 Campo solar

El campo solar está formado por los colectores, el sistema de HTF y diversos subsistemas.

Los colectores forman un sistema modular conformado por lazos conectados en paralelo. El conjunto se conecta por un sistema de tuberías aisladas por las que circula HTF. Este fluido es bombeado por el sistema de potencia hacia el campo solar, donde se distribuye por los lazos de colectores calentándose, para después volver al generador de vapor del ciclo de potencia.[13]

El colector cilindro parabólico está compuesto por:

- Cimentación. Es la parte que los soporta y fija al suelo de forma que el conjunto estructural soporte las

[8] Tomado de: https://www.energias-renovables.com/termosolar/acero-de-alta-resistencia-para-plantas-termosolares-20151222
[9] Tomado de: https://desenchufados.net/tecnologia-termica-solar-cilindro-parabolica/

cargas para las que está diseñado, suelen ser de hormigón armado. Se realizan en función de las dimensiones de los colectores y de las características de la estructura.

- Estructura. Aporta rigidez al conjunto de elementos que lo componen, suelen ser metálicas. Se busca que sea de buena calidad ya que cualquier deformación de esta a lo largo de su vida afectará a la concentración de la luz.
- Sistema de seguimiento solar. Es el mecanismo de seguimiento solar que se encarga de cambiar la posición del colector conforme el Sol se va moviendo.
- Reflector cilindro-parabólico. Es el que se encarga de reflejar la radiación solar que incide sobre él y proyectarla de forma concentrada sobre el tubo absorbente. Están formado por espejos hechos de plata o aluminio aplicados sobre chapa, plástico o cristal.
- Tubo absorbente. Es el encargado de convertir la luz solar concentrada en energía térmica en el HTF. Principalmente se trata de dos tubos, uno interior de metal, y otro tubo transparente de vidrio de alta transmitancia en el intervalo solar. [11]

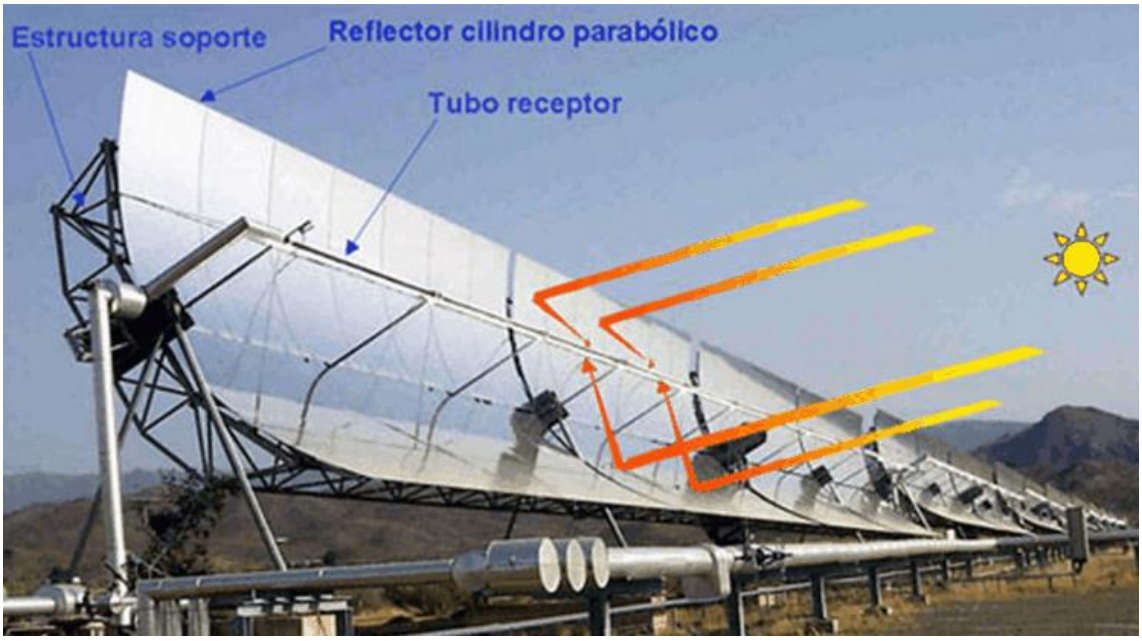

Figura 50. Colector cilindro-parabólico.[10]

5.2.2 Sistema de fluido térmico

El sistema de fluido térmico o sistema HTF, se trata de un circuito cerrado de tuberías, depósito de expansión y válvulas, por donde circula el fluido térmico. El objetivo del sistema es el de transmitir la energía térmica captada en el campo solar al bloque de potencia y al sistema de almacenamiento.

El circuito consta de una gran cantidad de tubos absorbentes a través del que circula el fluido térmico, que a medida que recorre los lazos de colectores solares se va calentando como se puede observar en la Figura 51. La tubería comienza el lazo con el color amarillo lo que implica que el fluido que está dentro de la tubería está frío, pero al acabar de realizar el lazo la tubería acaba siendo roja, lo que implica que el fluido dentro está rojo.

[10] Tomado de: https://themorningstarg2.wordpress.com/2012/03/16/tecnologia-cilindro-parabolico/

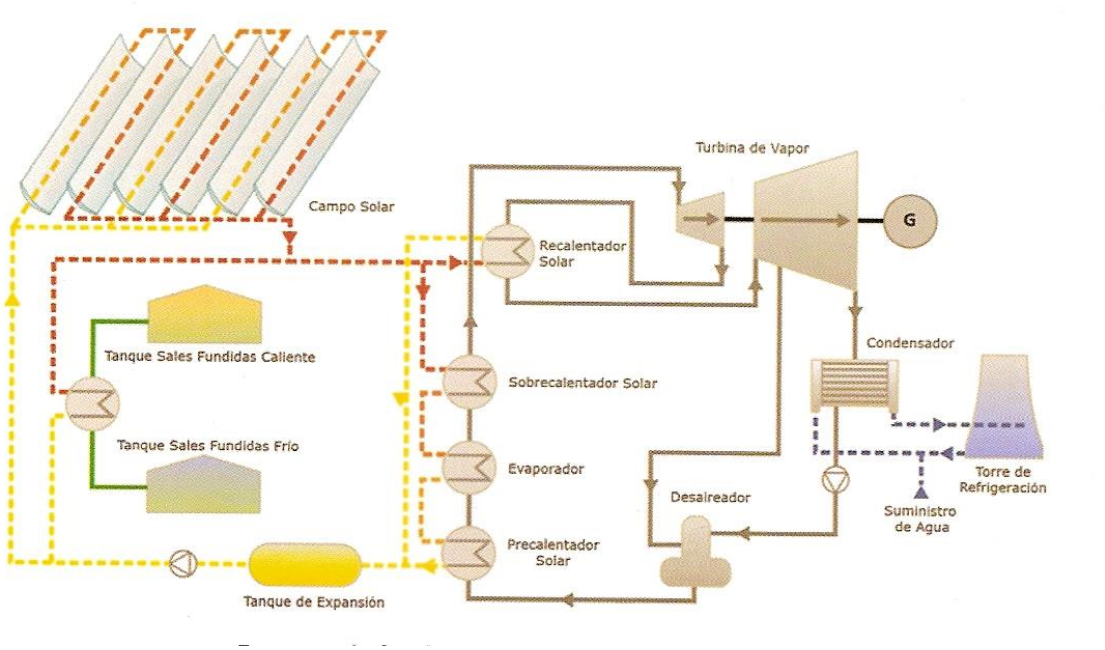

Figura 51. Planta con tecnología de sales fundidas.

Por el campo solar circularán dos tuberías en paralelo, una con fluido frío y otra con fluido caliente. Estas dos tuberías estarán conectadas a los lazos y cada uno contará con una conexión con la tubería de frío, que será la entrada del lazo, y con la caliente, que será la salida.

El tipo de HTF determinará el rango de temperaturas en la que opera, afectando a su vez al rendimiento máximo que se puede obtener en el ciclo de potencia. El fluido que normalmente se emplea es una mezcla eutéctica de dos hidrocarburos aromáticos: el bifenilo y el óxido de difenilo que tiene el inconveniente de tener un punto de congelación de 12ºC. A consecuencia de esta condición, será necesario hacer circular de forma continua el fluido para evitar congelaciones.[14]

El sistema de HTF está compuesto por diferentes subsistemas y equipos que se describen a continuación.

5.2.2.1 Sistema de bombeo

Será necesario un sistema de bombeo que haga circular el aceite térmico por toda la planta. Cuanto más grande sea el circuito de tuberías, más cantidad de bombas será necesario instalar. La principal función por tanto es bombear el fluido térmico desde el tanque de expansión pasando por el campo solar y el sistema de generación de vapor o sistema de almacenamiento, como se puede observar en la Figura 51. En dicha figura, las bombas están representadas como un triángulo dentro de un círculo. [11]

Otra función de las bombas es adaptar el caudal del fluido térmico a las necesidades puntuales de la instalación. Para ello, cada bomba poseerá un variador de frecuencia para poder modificar el caudal bombeado. Habrá que tener especial cuidado en la presión para que no se produzca el fenómeno de cavitación.

5.2.2.2 Sistema de nitrógeno

El sistema de nitrógeno está formado por diferentes tanques y se encarga de evitar la degradación del aceite por oxidación. Los tanques que son inertizados con nitrógeno son el tanque de expansión y el de rebose.

El tanque de expansión trata de un tanque a presión que se sitúa en el punto más elevado de la instalación y que se encarga de presurizar el sistema a una presión mediante la introducción de nitrógeno y de absorber las variaciones de volumen producidas por las variaciones de temperatura del fluido térmico.

Y por el otro lado, el de rebose recogen el fluido térmico que rebosa del tanque de expansión y el fluido térmico limpio procedente del sistema de regeneración. Este tanque forma parte del sistema colector principal, el cual es capaz de contener la totalidad del fluido térmico existente en la instalación.[15]

5.2.2.3 Sistema de regeneración del fluido

Está por un lado formado por tanques, llamado sistema de recuperación, que trata de eliminar del fluido térmico los vapores de alto punto de ebullición que se producidos por la degradación del aceite, antes de que se supere la solubilidad máxima de los mismos y empiecen a precipitar en el sistema. Y por otro lado están otros tanques que forman el sistema de merma, que recogen los vapores de bajo punto de ebullición del aceite con objeto de controlar su pureza.

5.2.2.4 Sistema colector principal

Es el sistema que es capaz de contener la totalidad del fluido térmico existente en la instalación. Está formado por el tanque de rebose previamente mencionado y por el tanque de almacenamiento auxiliar

5.2.3 Ciclo de potencia

El ciclo de potencia que se implementa es el tipo Rankine regenerativo con recalentamiento. Su principal función es generar electricidad. Siguiendo la Figura 52, se genera vapor a unas condiciones idóneas de presión y temperatura para poder mover la turbina que a su vez moverá un generador donde se producirá la electricidad.

Se utiliza vapor porque es un fluido barato y se puede ajustar su temperatura con precisión, por la relación entre presión y temperatura. Es capaz de transportar grandes cantidades de energía con poca masa. Su principal problema se da al tener que trabajar con altas presiones. [11]

Se necesitarán intercambiadores de calor para los procesos de absorción y cesión de calor.

Con el recalentamiento el vapor parcialmente expansionado en la turbina se vuelve a calentar, generalmente hasta la misma temperatura inicial, para posteriormente expandirlo de nuevo en la turbina. Su objetivo prioritario es reducir la humedad en los últimos escalonamientos y a consecuencia, aumentar el rendimiento interno de la turbina.[16]

Por otro lado, en el ciclo regenerativo se precalienta el agua con vapor extraído en puntos intermedios de la turbina. Como ventaja, el rendimiento térmico mejora, la cantidad de calor cedida en el condensador disminuye ocasionando que pueda tener menor tamaño, el gasto másico de vapor en los últimos escalonamientos de la turbina es menor, lo que permite reducir la velocidad de salida y, con ello, las pérdidas de escape; y el trabajo específico de la turbina disminuye. [11]

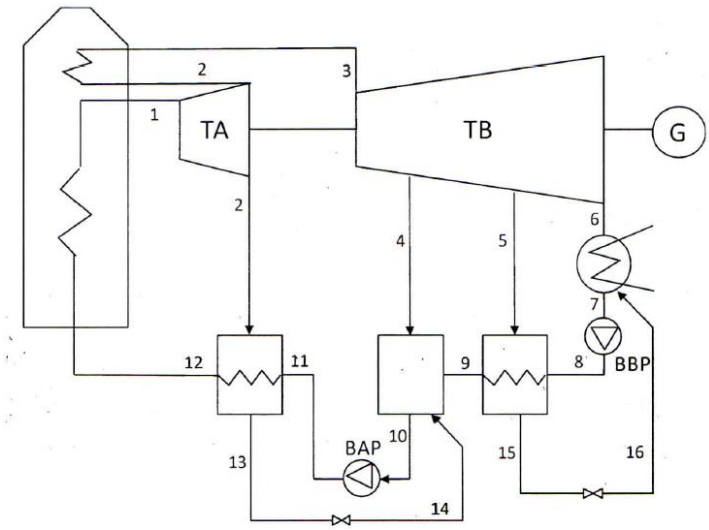

Figura 52. Ejemplo de turbina de vapor con Ciclo Rankine regenerativo con recalentamiento.

[11] Renovetec, «Ciclo Termodinámico de la turbina de gas».

5.2.3.1 Caldera

La caldera, situada en la Figura 52 entre los puntos 12-1-2, se encarga de producir vapor a elevada presión y temperatura. Entre sus componentes están el economizador, el evaporador y el sobrecalentador. El economizador, o precalentadores, son intercambiadores de vapor procedentes de la turbina que condensan el vapor y cede esa energía al agua del ciclo antes de este se inyecte en la caldera. Por otro lado, los evaporadores son intercambiadores que aprovechan el calor de los gases de escape de temperatura intermedia para evaporar el agua a la presión del circuito correspondiente. Y los sobrecalentadores son intercambiadores que se encuentran en la parte más cercana a la entrada de los gases procedentes de la combustión en la turbina de gas y por tanto, tratan de conseguir que el vapor que entre esté libre de gotas de agua.

5.2.3.2 Turbina de vapor

La turbina de vapor es el elemento principal, pues es la encargada de transformar la energía del vapor en movimiento rotativo. Esto se transmitirá al alternador por medio del rotor.

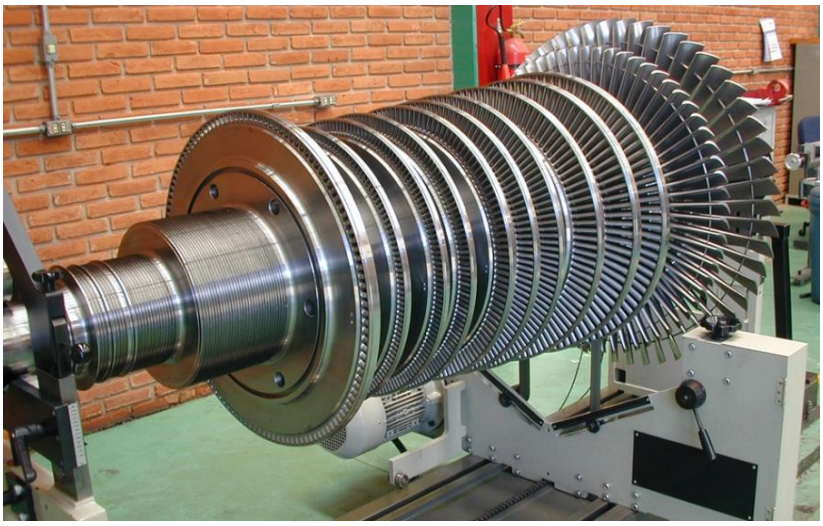

Figura 53. Turbina de vapor.

Los elementos principales de una turbina son estator más un rotor. El estator es un conducto fijo donde la energía almacenada en el fluido se transforma, pero no hay intercambio con el exterior de la máquina. El rotor son conductos móviles en los que se da lugar el intercambio de energía.

5.2.3.3 Condensador

Sirve para condensar el vapor de escape de la turbina a muy baja presión, por ello está situado a la salida de la turbina. A consecuencia de tener una muy baja presión, se consigue una temperatura menor que la de saturación.

5.2.3.4 Bombas

Sirven para elevar la presión del agua condensada hasta la presión de admisión de la turbina. También puede ser necesaria bombas en el caso del ciclo con recalentamiento para el fluido que haya entre ambas turbinas, como se puede ver en la Figura 52 se trataría de la bomba BAP[12].

5.2.4 Sistema auxiliar

5.2.4.1 Sistema de almacenamiento de sales

El sistema de almacenamiento de sales es el encargado de almacenar la energía térmica para ser usada en las

[12] Bomba de Alta Presión.

horas de baja o nula radiación solar. La utilización de sales se debe a que para almacenar la misma energía con aceite térmico serían necesarios unos tanques mucho más grandes.

Serán necesarios intercambiadores para realizar el intercambio térmico entre el aceite y las sales fundidas. El número de intercambiadores será en función de la temperatura de aproximación entre aceite térmico y sales.

En el proceso de intercambio, el aceite circula por los tubos, mientras que las sales circulan por la carcasa.

5.2.4.2 Caldera auxiliar

Su principal función es mantener el aceite en los valores apropiados. Esto puede ser especialmente útil para situaciones en las que los captadores no suministran suficiente energía. La cantidad de combustible fósil vendrá marcada por la normativa vigente.

5.2.5 Sistemas eléctricos

Para el sistema eléctrico se necesitará un generador conectado a la turbina. Esta máquina se encarga de transformar la energía mecánica en eléctrica mediante la acción de un campo magnético sobre los conductores eléctricos dispuestos sobre un estator.

También será necesaria una subestación eléctrica para la transformación de la tensión de red o del generador a una tensión adecuada a las necesidades. La subestación que se va a implementar en nuestra planta de la simulación se trata de una subestación a la intemperie, cuyo aislante es el aire. Para su control se empleará diferente aparamenta: interruptores, seccionadores, aparatos de protección y transformadores de intensidad y tensión.

Los seccionadores se encargan de cortar las líneas cuando no circula corriente a través de ellas, mientras que los seccionadores la cortan cuando circula corriente.

El transformador es el principal elemento de la subestación, es el encargado de convertir el valor de la tensión del generador en el valor de la tensión de la red donde se vuelca la energía producida. También se líneas de distribución aéreas o subterráneas que se encargan de la distribución de la energía eléctrica. Otro elemento necesario será el Grupo electrógeno, son generadores diésel empleados en caso de averías o accidentes que dejen la instalación sin suministro de electricidad de la red. Se suelen usar diésel ya que tienen un tiempo de reacción muy corto. [15]

5.3 Control de la Planta Termosolar mediante el uso de vehículos no tripulados

Las Plantas Termosolares suelen estar ubicadas en lugares con alta radiación solar, siendo esta su fuente principal de funcionamiento. Sin embargo, este no es un parámetro que sea constante en plantas que ocupan una gran extensión de terreno y que pueden llegar a resultar más afectadas por factores como las nubes.

El control principal de temperatura se lleva a cabo variando el caudal de HTF que circula por el campo solar. Su objetivo se basa en producir energía alrededor de un punto de operación nominal en el que la temperatura de salida del campo solar es cercana al límite térmico. Uno de los problemas que pueden aparecer en este tipo de control surge al localizar una nube, a lo que consecuentemente decrementa el caudal del sistema de HTF. Esta acción beneficia a las zonas afectadas por las nubes; sin embargo, afecta negativamente a otros colectores no alterados por las nubes y que, a consecuencia del cambio de flujo, sufren un aumento de subida de temperatura en el HTF. Este es uno de los motivos principales por los que por tema de seguridad, los colectores, se desfocalizan a pesar de la pérdida de energía que implica. Para entender mejor el control de una Planta Termosolar se puede consultar el libro [18]

Diversos estudios, como [19] y [20], han demostrado que aplicar un modelo predictivo presenta mayores ventajas debido que se homogeniza la temperatura y se consigue reaccionar con mayor eficacia a eventos como nubes mediante el uso de válvulas.

Para la implementación de una estrategia de MPC distribuido es necesario tener una estimación espacialmente distribuida de la radiación solar, y para realizar las mediciones necesarias es necesario el uso de una "*Wireless Sensor Network*" [21]. Sin embargo, como los sensores de irradiancia directa (pirheliómetros) son muy caros para montar una red de ellos a lo largo de la planta, se ha propuesto montar los sensores sobre vehículos

terrestres y aéreos no tripulados, creando asi una "*Robotic Sensor Network*" [22].

6 SIMULACIÓN

En este punto se estudiará en primer lugar, la Planta Termosolar creada en Unity y también se comentarán los aspectos más importantes del código implementado en Visual Studio.

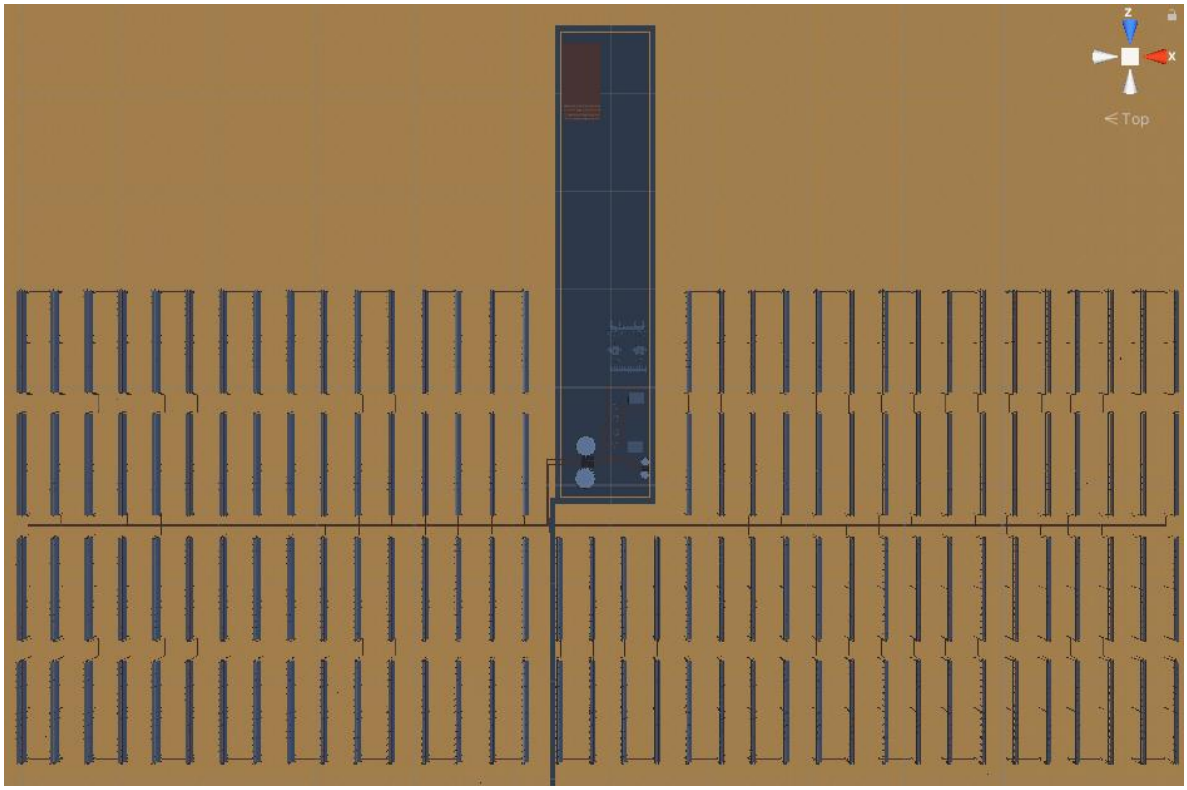

Figura 54. Vista de la planta en Unity.

6.1 Recreación dentro de Unity.

En Unity como se ha explicado anteriormente en el apartado 2, se van creando objetos a los que se les van añadiendo funciones. Para poder ver los objetos que se han utilizado para la simulación, habrá que acceder a la Ventana de Jerarquía, donde aparecerá un listado con todos los objetos que se han utilizado para hacer la planta.

En la Figura 55 se pueden observar todos los objetos que han sido necesarios para realizar la recreación. Entre ellos se encuentra la "*MainCamera*" y la "*Direction Light*" que eran objetos que vienen de serie cada vez que se crea un proyecto. La "*MainCamera*" se usará entre otras cosas para crear secuencias donde poder visualizar el movimiento de los drones; sin embargo, deberá ser eliminada cuando se pase a las Oculus Quest.

Con la idea de crear un medio, se añadirán nubes y un terreno. El terreno en un principio no tendrá textura, por lo que habrá que añadirle una capa con el material elegido, en este caso un material arenoso como se puede ver en la propia vista de la Figura 54. Para añadir el material del terreno, se debe acceder a la Ventana Inspector del terreno, pulsar sobre el pincel en pestaña "*Terrain*" y seleccionar "*Edit Terrain Layers*". El material elegido en este caso se ha elegido de una librería descargada de la "*Asset Store*" denominada "*Ground materials*". Una vez seleccionado el material, se añadirá automáticamente a todo el terreno. En la misma pestaña de "*Terrain*" se permiten muchas más opciones como añadir árboles, flores, etc…

Por otro lado, para añadir Nubes será necesario descargarse la librería "*Standard Assets*", ya que incluye un apartado denominado "*ParticleSystem*" que cuenta con una variedad de PreFabs como nubes o humo. En concreto, usaremos el "*DustStorm*" y lo configuraremos, dentro de la Ventana Inspector, las características para adecuarlas a lo que necesitamos. Por ello, se cambiará el color de las nubes para que no parezca una

tormenta, la velocidad con la que van apareciendo y desapareciendo y también la cantidad de concentración de nubes. Por otro lado, dentro de la Ventana Scene será necesario modificar su tamaño para que ocupen todo el terreno. Finalmente, el terreno y las nubes quedarán como se puede ver en la Figura 56, un terreno arenoso y unas nubes suaves que van evolucionando en diferentes concentraciones.

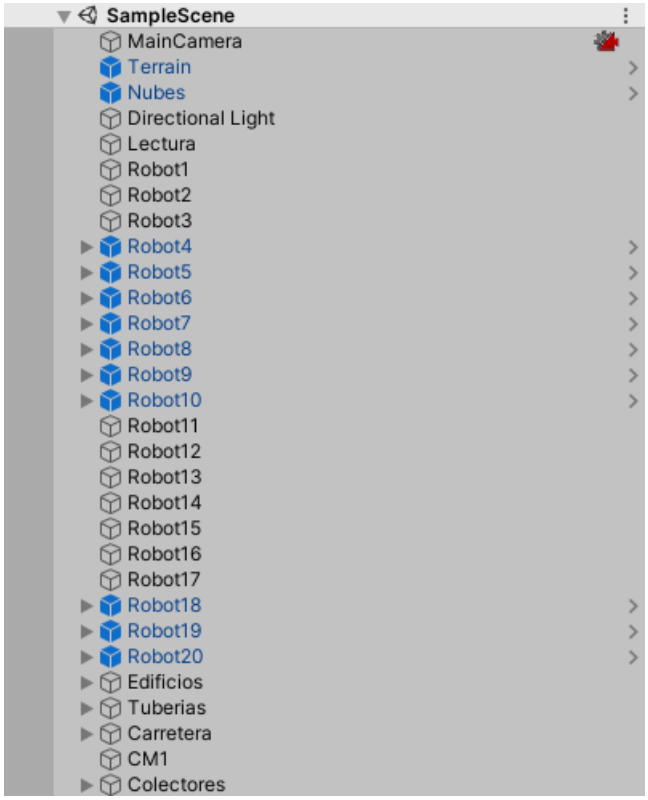

Figura 55. Objetos de la Ventana de Jerarquía.

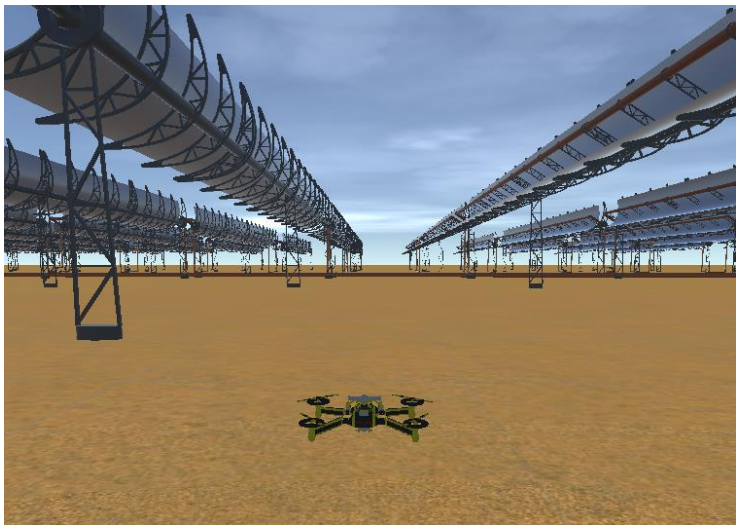

Figura 56. Nubes en la escena.

También existe un objeto vacío denominado Colectores, donde se agruparán todos los lazos de colectores en el campo solar (ver Figura 57). A su vez, cada colector lleva añadido un material metálico para darle un mejor aspecto. La disposición de los colectores se hará en función de la configuración de las trayectorias de los drones con el objetivo de que los robots aéreos al desplazarse por el eje z lo hagan por los espacios entre colectores que forman pasillos y nunca por encima de los colectores solares. Los terrestres, por el contrario, circularán por la debajo de los colectores y evitará las zonas con tuberías.

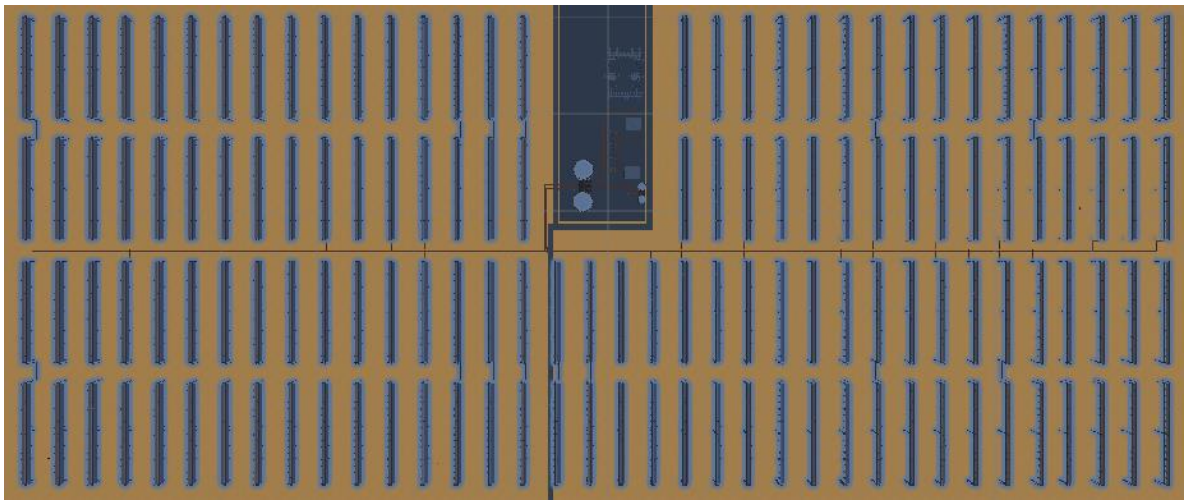

Figura 57. Vista de la planta con todos los lazos seleccionados.

Con un método similar, se crearán los objetos vacíos Edificios, tuberías y carreteras (ver Figura 55).Estos al igual que con los colectores, servirán para recoger diferentes objetos similares. Por un lado, el objeto de tuberías incluirá todas las tuberías existentes en el campo solar a excepción de las que conforman el lazo de colectores (ver Figura 58), ya que estas irán incluidas dentro de los mismos colectores.

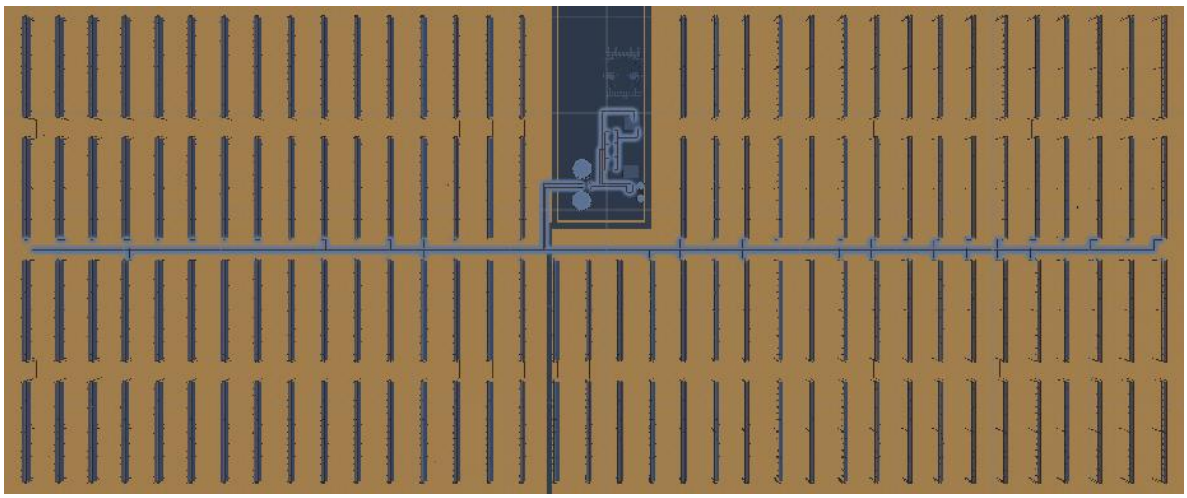

Figura 58. Vista de la planta con las tuberías seleccionadas.

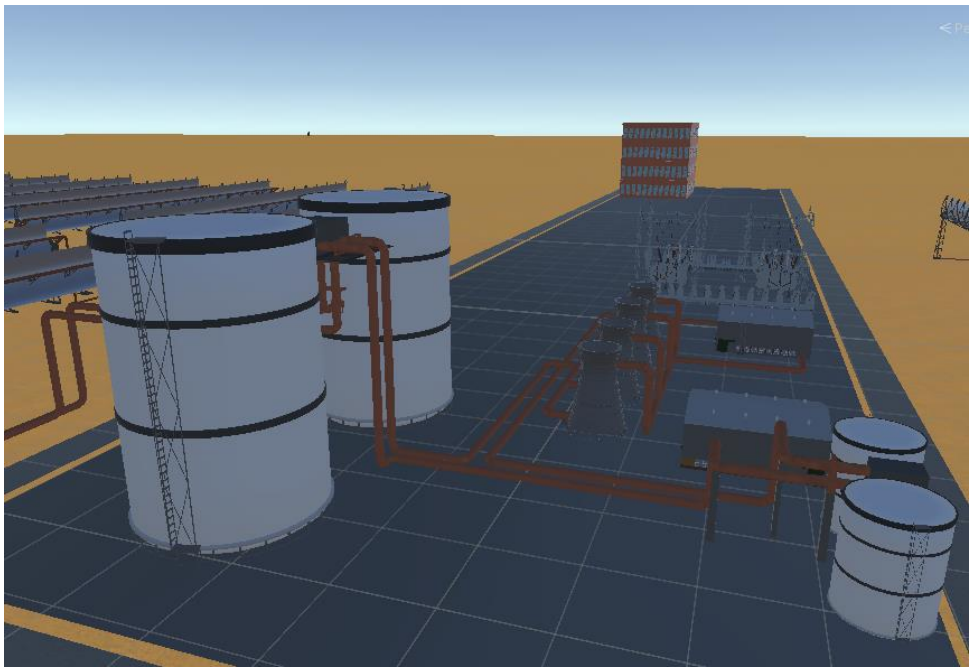

Figura 59. Edificios de la Planta Termosolar.

Y también estará la zona de edificios (ver Figura 59) donde se concentrarán todos los edificios de la Planta Termosolar. Se ha sobredimensionado ante la posibilidad de una expansión futura, para poder así disponer del suficiente espacio para la nueva maquinaria. Dentro de la zona de edificios, y siguiendo el ciclo de potencia, las tuberías que circulan por los lazos entran y salen a la zona de los edificios por los tanques de sales fundidas. Estos están representados, junto a sus respectivos intercambiadores, en la Figura 60.

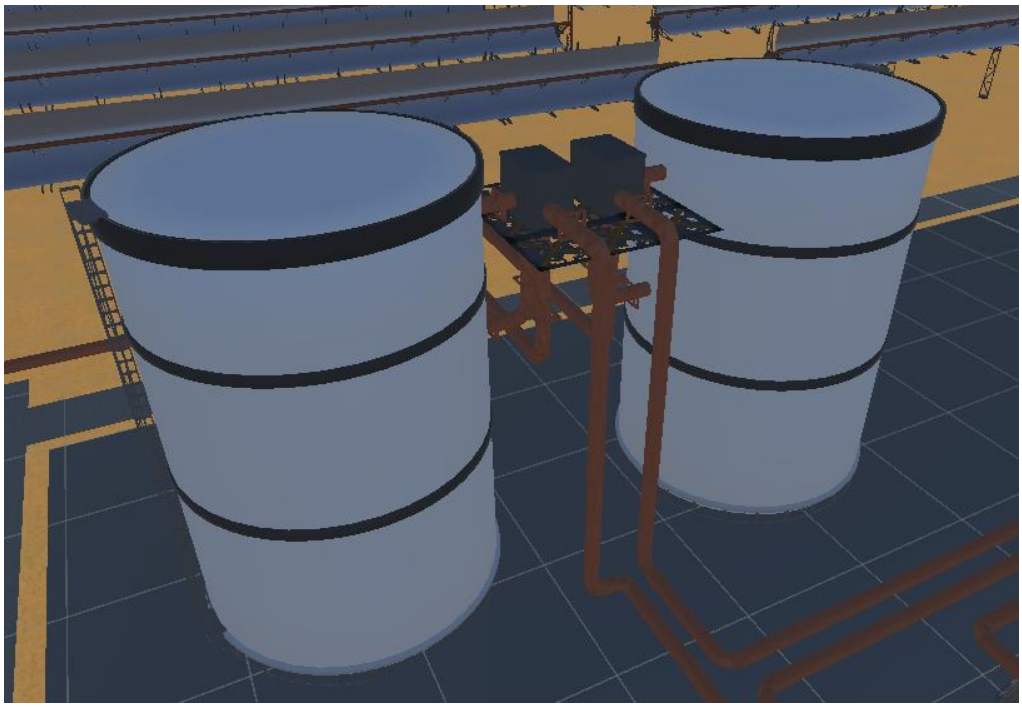

Figura 60. Tanques de sales fundidas e intercambiadores.

A la derecha de estos tanques, se encontrará el sistema de HTF (ver Figura 61). Siguiendo las tuberías, se encontrará el edificio destinado a los generadores (ver Figura 62), con cuatro torres de refrigeración.

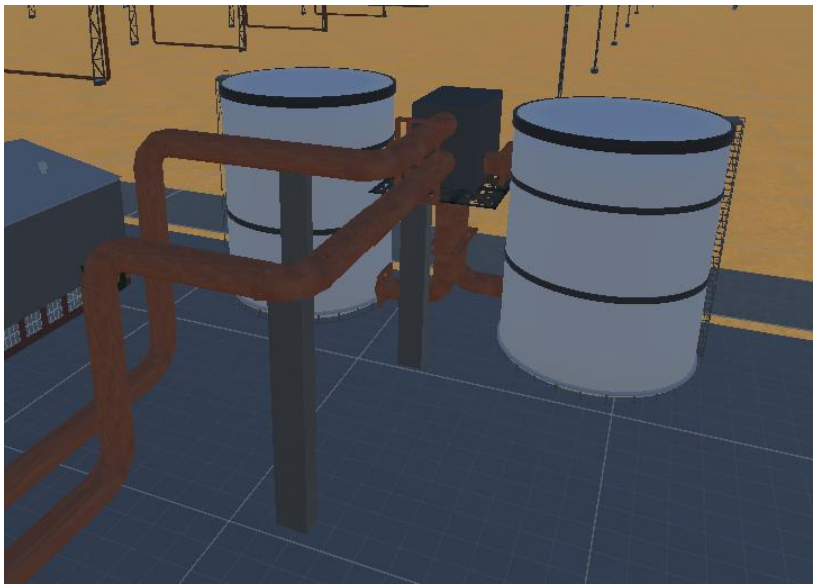

Figura 61. Sistema de HTF.

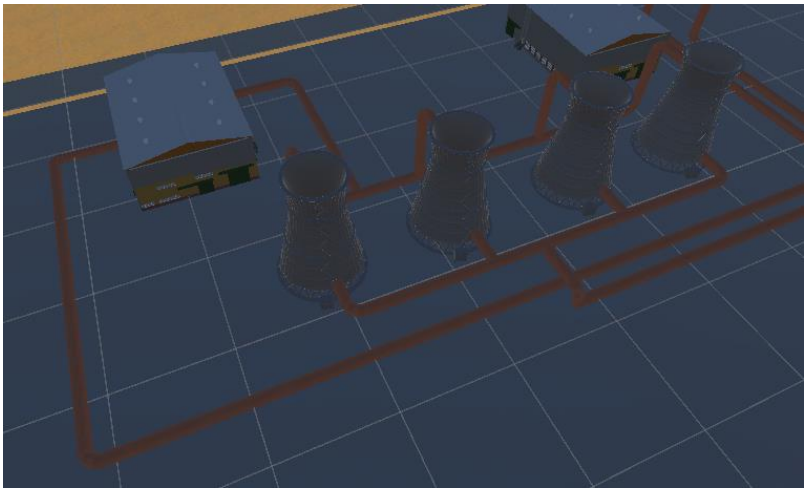

Figura 62. Generadores y Torres de refrigeración.

Como se puede visualizar, se ha tenido preferencia a que las tuberías circulen a la altura del suelo, esto se debe a diversos motivos, pero principalmente al económico. Las tuberías cambiarán su altura cuando sea imposible llevarlas por el suelo.

Se añadirán también carreteras que den acceso a la planta (ver Figura 63). Estas carreteras rodearán la planta y se incluirá una salida por la parte inferior. Para poder pasar por las tuberías de calor y frío que cruzan la planta, se agregará un pequeño puente que cruce las tuberías.

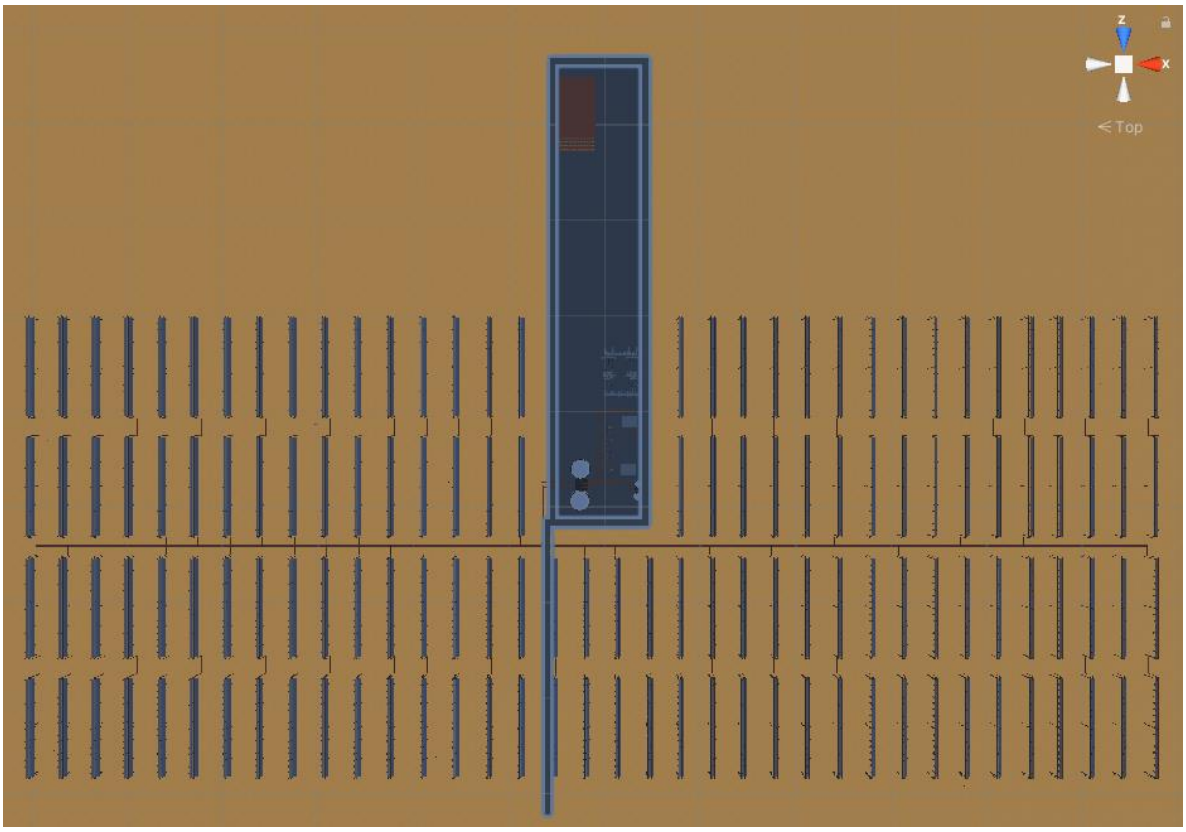

Figura 63. Vista de la planta con todas las carreteras seleccionadas.

Por último, se añadirán a la escena los diferentes vehículos no tripulados. Los modelos seleccionados se pueden ver en la Figura 64. En el caso de que el vehículo sea terrestre, se usará el modelo de vehículo de cuatro ruedas de la izquierda. Y en el caso de que sea un vehículo aéreo, se utilizará el modelo de la derecha. Como la limitación del código es que se pueden mandar un máximo de diez trayectorias, se deberán añadir diez robots terrestres y otros diez robots aéreos. Además, se adjunta una tabla aclaratoria para saber, respecto a la Figura 55. Objetos de la Ventana de Jerarquía.qué robots son terrestres y cuales aéreos.

Figura 64. Vehículos no tripulados terrestres y aéreos.

Robot 1	Vehículo no tripulado terrestre
Robot 2	Vehículo no tripulado terrestre
Robot 3	Vehículo no tripulado terrestre
Robot 4	Vehículo no tripulado aéreo
Robot 5	Vehículo no tripulado aéreo

Robot 6	Vehículo no tripulado aéreo
Robot 7	Vehículo no tripulado aéreo
Robot 8	Vehículo no tripulado aéreo
Robot 9	Vehículo no tripulado aéreo
Robot 10	Vehículo no tripulado aéreo
Robot 11	Vehículo no tripulado terrestre
Robot 12	Vehículo no tripulado terrestre
Robot 13	Vehículo no tripulado terrestre
Robot 14	Vehículo no tripulado terrestre
Robot 15	Vehículo no tripulado terrestre
Robot 16	Vehículo no tripulado terrestre
Robot 17	Vehículo no tripulado terrestre
Robot 18	Vehículo no tripulado aéreo
Robot 19	Vehículo no tripulado aéreo
Robot 20	Vehículo no tripulado aéreo

Tabla 1. Tipos de robots en la Escena.

6.2 Scripts necesarios para la simulación.

Se necesitarán scripts para poder recrear los movimientos de los vehículos no tripulados en función de las coordenadas provistas desde Matlab. En total se necesitará un script por robot, en el que se gestionará su propio movimiento, y también un script de lectura que se encargará de la comunicación con Matlab y del tratamiento del mensaje.

Como se comentó en el apartado 3.5, se necesitará crear comunicaciones entre scripts, en concreto será entre el script lectura con todos los robots, para mandarles las correspondientes coordenadas. Por ello, a cada vehículo no tripulado será necesario crearle su propia etiqueta o "*Tag*".

Para que el script Lectura se ejecute nada más iniciar la simulación, será necesario que esté presente dentro del plano. Por tanto, se deberá crear un objeto vacío, al que se le ha llamado Lectura en la Figura 55, y al que se le asociará el script con el que tiene comunicación con Matlab. Para más información sobre el desarrollo del programa, se adjuntan en los Anexos: Anexo II: Código Visual Studio y Anexo I: Código Matlab

7 GAFAS DE REALIDAD VIRTUAL

Para la visualización en realidad virtual se va a utilizar el modelo de gafas de realidad virtual Oculus Quest. Estas gafas se lanzaron al mercado con la novedad de no ser necesario cables para incorporar los gráficos desde el ordenador. Y que además, cuenta con todos los sensores integrados. A pesar de que no llega a tener la misma calidad de imagen que unas gafas de realidad virtual controladas por ordenador, consigue llegar a la altura en la mayoría de juegos ya que no es habitual que estos requieran un nivel de procesamiento realmente alto. Por estas opciones, se ha concluido que estas gafas eran las idóneas para la simulación.

Figura 65. Oculus Quest.

7.1 Características técnicas

Estas gafas cuentan con unas lentes ajustables OLED de 1600 x 1400 píxeles. Cuenta con 4 sensores de movimiento que permiten trasladar tus movimientos a la realidad virtual y también un seguimiento a escala de la sala. Su sistema operativo está basado en Android, lo que permitirá cargar la simulación generándola como una apk. El sonido está integrado en el visor, por lo que no es necesario integrar los auriculares, aunque cuentan con la posibilidad de utilizar auriculares. Lleva controles propios denominados Oculus Touch. El procesador es un Qualcomm Snapdragon 835, con una frecuencia de actualización de 72 Hz. Tiene 4GB de RAM y batería de iones de litio.

7.2 Entorno de trabajo

Las gafas tienen incorporado una conexión USB para poder conectarse al ordenador. Sin embargo, estas gafas solo interactuarán con el ordenador como dispositivo de carga a no ser que se acepte desde el propio mundo virtual que el ordenador pueda acceder a los datos de las gafas. Con este permiso se puede manipular los diferentes archivos desde el ordenador: sin embargo, aún no será posible cargar desde Unity la aplicación. Para ello, habrá que cumplir unos pasos previos.

7.2.1 Aplicación móvil Oculus Quest

Para administrar las gafas de realidad virtual, se utilizará una aplicación de móvil denominada Oculus[13]. Esta aplicación se puede adquirir dentro de las plataformas de descarga de aplicaciones del propio dispositivo. Una vez descargada, se deberá emparejar la aplicación con el visor. Esta aplicación permite entre otras cosas

[13] Oculus es una aplicación de Facebook Technologies.

gestionar remotamente la conexión a WiFi o visualizar la batería restante de los controles o del propio visor. Y en opciones avanzadas, permite activar el Modo Desarrollador. Este modo permitirá cargar aplicaciones creadas por el propio usuario dentro de la cámara, por lo que será esencial para subir un proyecto hecho desde Unity.

Para poder activar el Modo Desarrollador, se deberá haber dado previamente de alta el perfil dentro de la web de "*Oculus Developer*". Otra opción es añadirse a un grupo de "*devepolers*" o desarrolladores ya existente mediante el uso de una invitación.

7.2.2 Android

El proyecto se va a exportar a la cámara como una aplicación Android, y para ello será necesario preparar Unity y el propio ordenador donde se trabaje, para trabajar con esta extensión.

Añadir las funciones Android pueden abarcarse de dos maneras diferentes en función de la versión en la que se esté o se vaya a trabajar de Unity. En primer lugar se va a comentar el caso en el que ya se tenga instalada la versión en la que se quiere trabajar. Para ello, será necesario que, dentro de la versión en la que se esté trabajando de Unity, se descarguen los módulos de Android. Para ello, en Unity Hub, se deberá seleccionar "*Add Modules*", tal y como se ve en la Figura 66.

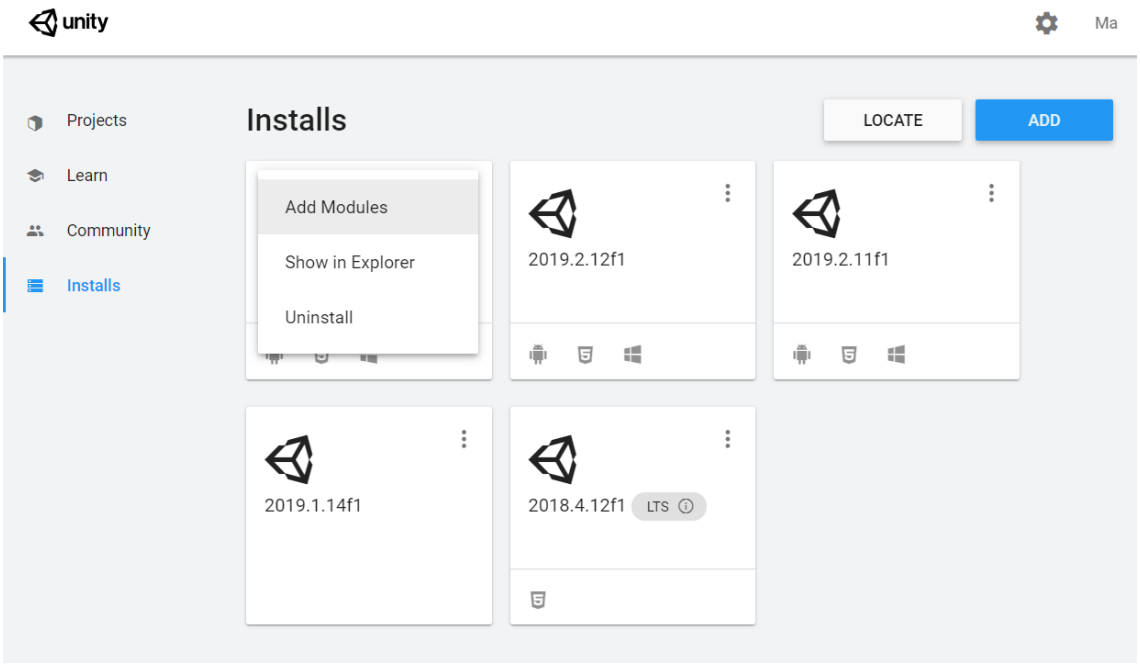

Figura 66. Instalación de módulos en una versión previamente instalada.

A continuación, se deberá seleccionar la opción "*Android Build Support*", justo como se puede ver en la Figura 67. Incluyendo todas las subpestañas que conlleve esa elección.

Figura 67. Opción a descargar para poder tener el módulo Android.

Una vez aceptado todos los términos y condiciones, se iniciará el proceso de instalación. Una vez finalizado aparecerá un icono debajo de la ventana de la versión de Unity que indica que está disponible esta opción de trabajo en esta versión de Unity.

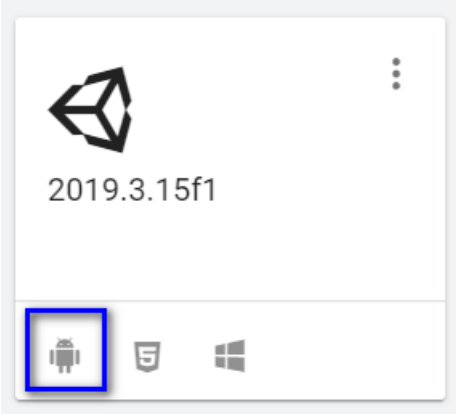

Figura 68. Versión de Unity con Android.

En el caso de que se quisiese trabajar en Android para una nueva versión aún no instalada de Unity. Dentro de la misma ventana de "*Install*", se deberá pulsar sobre el botón superior a la izquierda azul llamada "*Add*", que se puede ver en la Figura 66. Una vez pulsado se desplegará una lista de opciones para descargar las versiones disponibles de Unity y a continuación, una lista de módulos disponibles de Unity. Entre ellos, estará la opción de "*Android Build Support*", como en la Figura 67, que tendrá que ser seleccionado para que sea añadido dentro de la versión nueva de Unity. Una vez instalado la versión y el módulo, deberá aparecer una nueva ventana para esta opción, también con el símbolo de Android, como se puede observar en la Figura 68.

Una vez se tiene descargada la plataforma será necesario, dentro de la simulación que se quiera exportar a las gafas, comprobar que todo está correcto. Una forma es entrando en "*Preferences*", desde del desplegable de "*Edit*", y pulsando sobre "*External Tool*". Aquí se encontrará un apartado Android que deberá tener todo sin ninguna advertencia, como en la Figura 69. En el caso de que apareciese un símbolo acompañado de un mensaje de advertencia sucedería que la instalación no fue exitosa y convendría repetir el proceso de instalación.

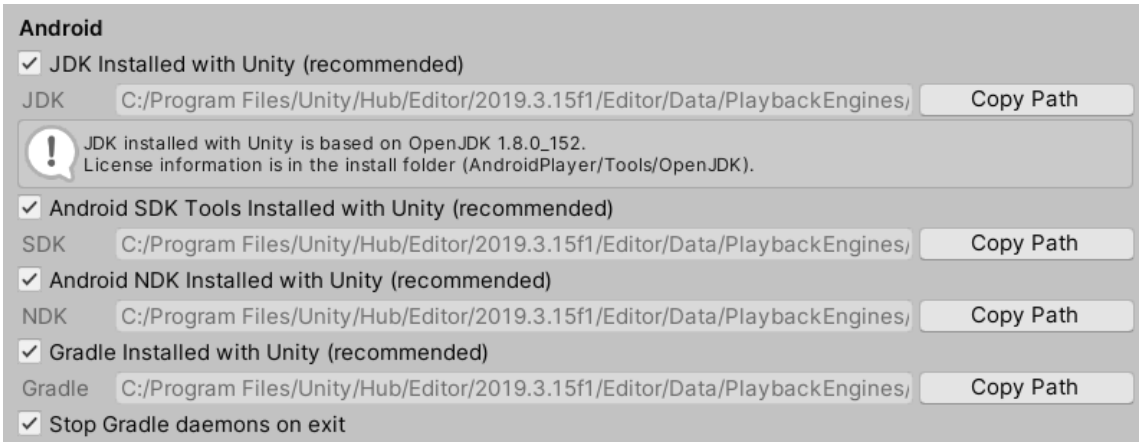

Figura 69. Información Android dentro de Unity.

En el caso de que esté todo correcto, se puede proceder a pulsar sobre "*Build Settings*" en la pestaña desplegable de "*File*", y pulsar sobre "*Switch Platform*" en la ventana de Android. Una vez se haya completado la configuración, deberá quedar como la Figura 70, listo para construir la apk.

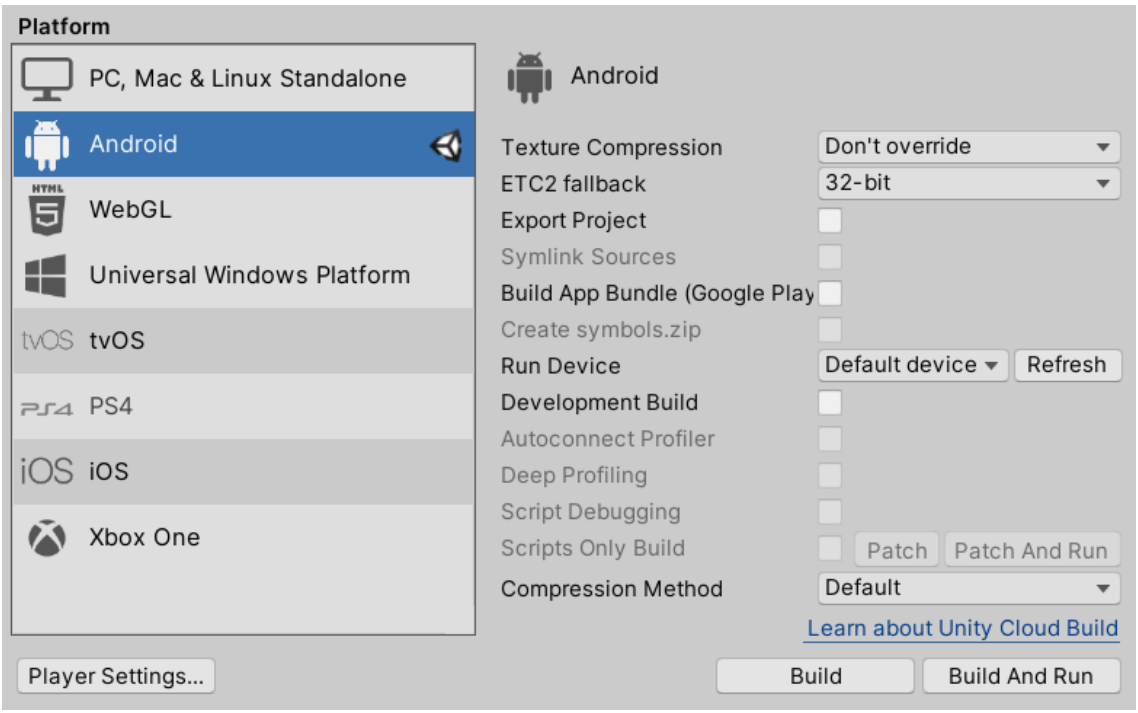

Figura 70. Opciones de configuración de plataforma de construcción en Unity

7.2.3 Configuración de los drivers

A pesar de tener preparado Unity para la construcción de la apk, se tendrá que preparar Unity y el ordenador para trabajar con Oculus. En el primer caso, puede ser un poco más complicado para los ordenadores con sistemas operativos Windows, por razones que se explicará más adelante.

Como se ha comentado en el apartado 7.2.1 sobre la aplicación Oculus, para poder conectar las gafas de realidad virtual al ordenador y poder cargar aplicaciones desde Unity, será necesario activar el Modo Desarrollador dentro de la aplicación de móvil. También será necesario dar permiso desde las propias gafas.

Una vez se tenga activado el Modo Desarrollador, se podrá conectar al ordenador mediante el cable USB. En el caso de que el ordenador no sea Windows, identificará el visor sin problema; sin embargo, si el ordenador es Windows, lo identificará como un Dispositivo desconocido.

En este caso, será necesario descargar los "*Oculus ADB Drivers*". Estos son accesibles dentro de la página web "*OculusDevelopers*", se descarga como un archivo zip que hay que descomprimir y pulsar con el botón derecho sobre el archivo inf[14] para seleccionar "*Instalar*". Para que comience a funcionar, será necesario reiniciar el ordenador.

Una vez reiniciado, cuando se conecten las gafas aparecerán como "*ADB*" pero aún no lo reconocerá como una Oculus, por lo que será necesario continuar configurando el dispositivo. Para ello, dentro del "*Panel de Control*" del equipo, se deberá entrar como Administrador en el "*Administrador de dispositivos*". Dentro de "*Otros dispositivos*" deberá aparecer un dispositivo "*ADB Interface*", como en la Figura 71.

[14] Archivo de texto sin formato utilizado para la instalación de software y controladores.

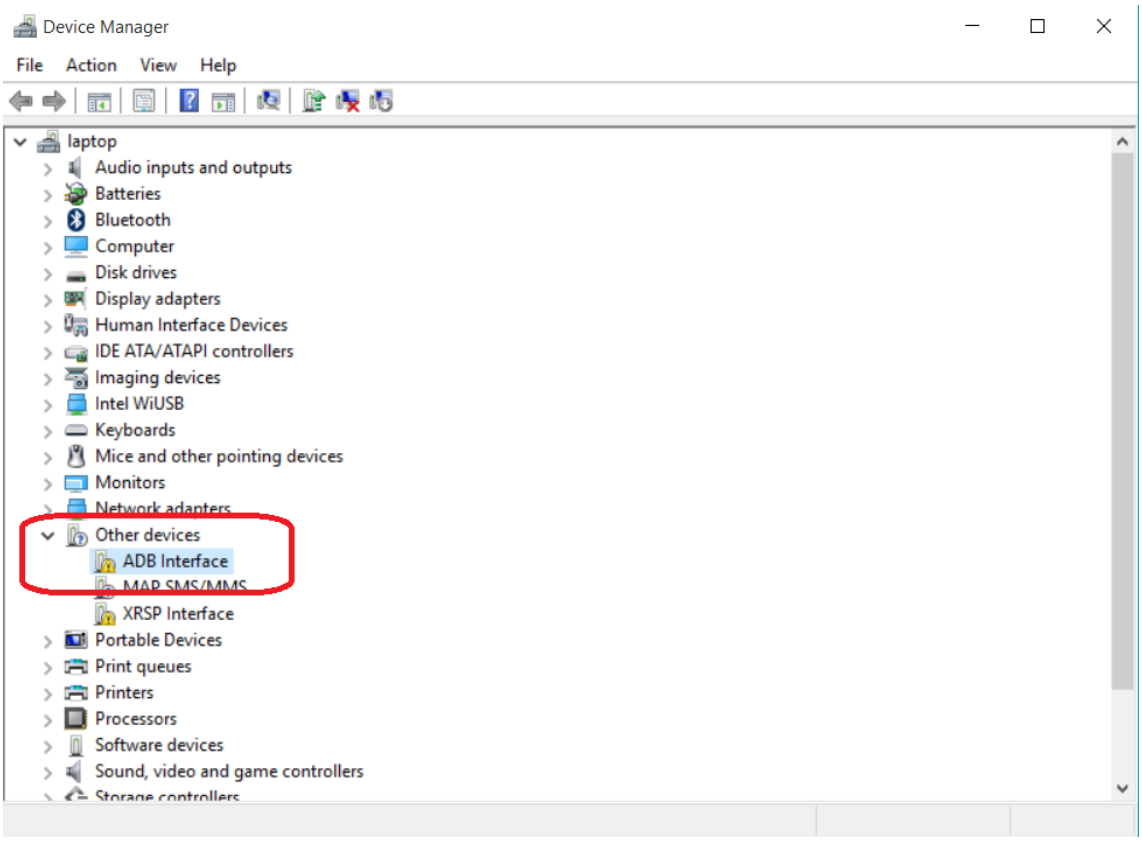

Figura 71. Panel de control con gafas de realidad virtual reconocidas como ADB Interface.

Sobre este dispositivo, se deberá pulsar con el botón derecho y elegir "*Actualizar software del controlador*". A continuación, se darán dos opciones de la que se tiene que elegir la segunda, "*Buscar un software de controlador en el equipo*". Esto llevará de nuevo a dos opciones, una en la que se puede examinar del propio equipo el software del controlador o elegir en una lista de controladores de dispositivo en el equipo. Se elegirá la segunda opción, y aparecerá una lista de tipos de hardware comunes de las que se tendrá que elegir "*Oculus Device*" (ver Figura 72).

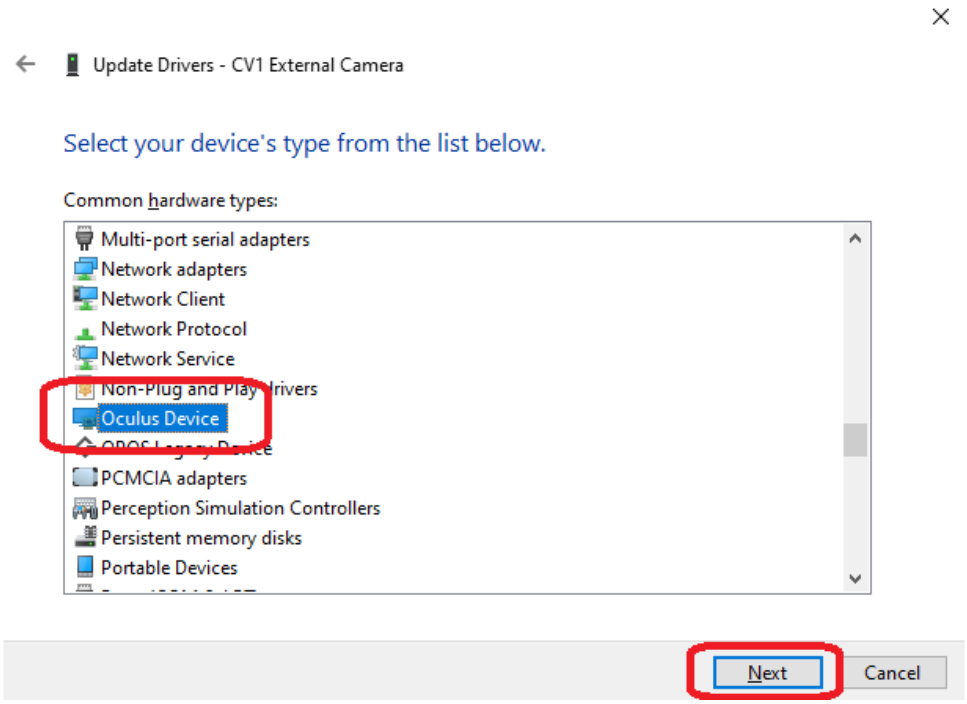

Figura 72. Lista de tipos de hardware.

Una vez se halla pulsado sobre aceptar, aparecerá una nueva ventana donde se debe elegir el controlador del dispositivo para ese hardware. En este caso, en "*Manufacturer*" se elegirá "*Oculus VR, LLC*". Y para el "*Model*" se elegirá "*Oculus Composite ADB Interface*".

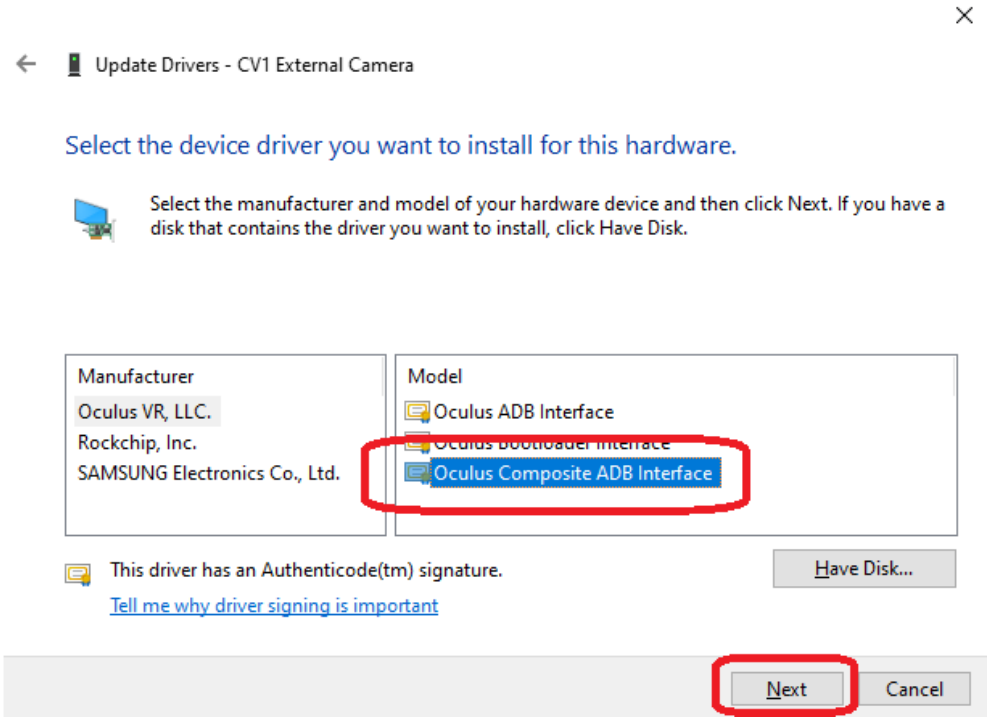

Figura 73. Elección de controlador del dispositivo.

Se deberá aceptar el cuadro de advertencia en el que explica que instalar ese controlador de dispositivo no es recomendable debido a que Windows no puede verificar que sea compatible. Una vez instalado, el dispositivo pasará a formar parte de los demás dispositivos como Oculus. Previamente, preguntará en el propio visor si se debe conectar con el ordenador y se deberá aceptar.

Una vez se tiene configurado los drivers para que establezcan conexión, únicamente faltará adaptar Unity para que trabaje con Oculus.

7.2.4 Configuración de Unity

Para poder conectar las Oculus a Unity, y así cuando se cree la apk automáticamente se cargue en las gafas, será necesario aplicar varias configuraciones. En primer lugar, se deberá acceder a la Asset Store y buscar "*Oculus Integration*". Esta librería es necesaria pues es la que permite interactuar con las gafas de realidad virtual.

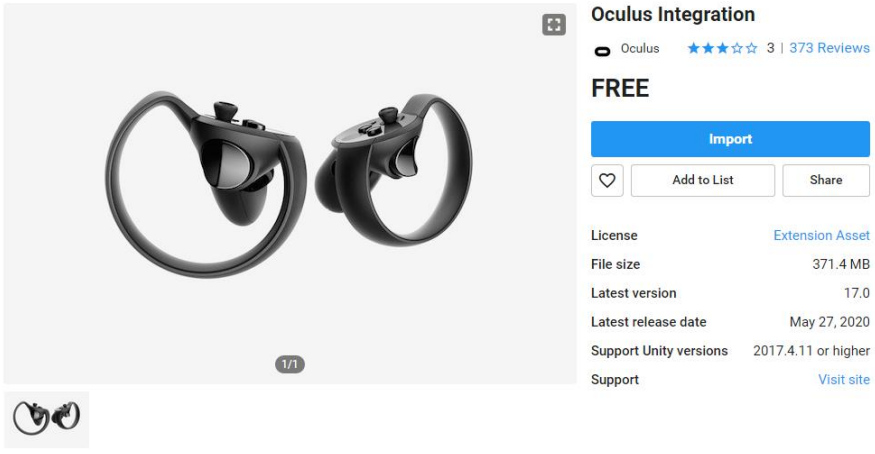

Figura 74. Librería "*Oculus Integration*".

En cuanto se haya descargado e importado la librería, se deberá eliminar la "*MainCamera*" del programa y añadir a la Ventana de Jerarquía (ver Figura 24) el objeto "*OVRPlayerController*". Este será la visión desde las gafas, y por ello se deberá situar desde el punto que se desee que empiece la visión de la simulación. También, se le deberá añadir el "*Local Avatar*" dentro del "*OVRPlayerController*" para poder tener un avatar dentro de la escena que permita cuando se esté en el mundo virtual, moverse por este usando los mandos.

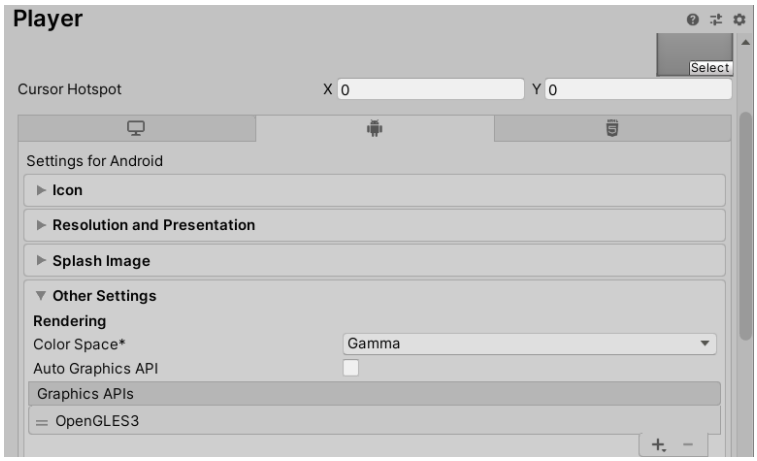

Figura 75. Opciones de configuración del proyecto para "*Player*" en Android

A continuación, se deberá acceder desde "*Editor*" a los "*Project Settings*". Dentro de la ventana que se abrirá, se tendrá que atender a la pestaña de "*Player*" pues será aquí donde se tendrá que añadir la configuración para las Oculus. En la Figura 75 se puede observar como en la ventana "*Player*" existe la opción de configuración Android, que es la que se debe seleccionar pues es la plataforma que se va a utilizar. Lo primero que se debe hacer es en "*Other Settings*" eliminar "*Vulkan*" de los "*Graphics APIs*" ya que los gráficos que interesan son "*OpenGLES3*".

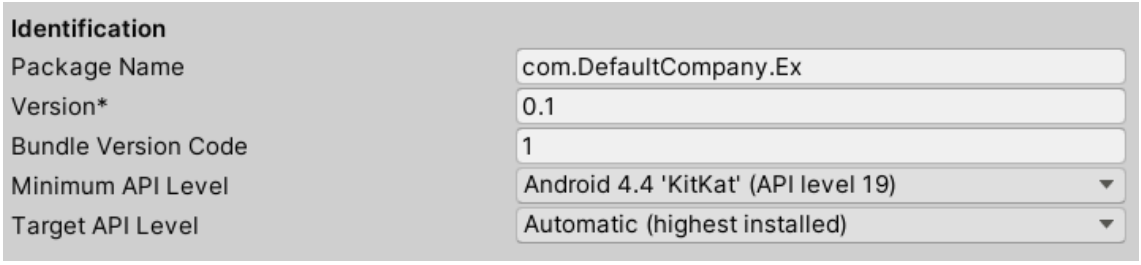

Figura 76. Configuración de la Identificación.

Además, en el apartado de "*Identification*" (ver Figura 76) se elegirá la versión mínima de Android que será capaz de arrancar la simulación. En este caso, se ha optado por la 4.4. Y por último, en "*XR Settings*" habrá que añadir las Oculus como "*Virtual Reality SDKs*" (ver Figura 77).

Una vez se hayan completado estos pasos, bastará con volver a la ventana donde se cambió a la plataforma Android y esta vez pulsar "*Build and Run*" (ver Figura 70). Tardará un poco en crearse, pero se creará automáticamente dentro del menú de la Oculus.

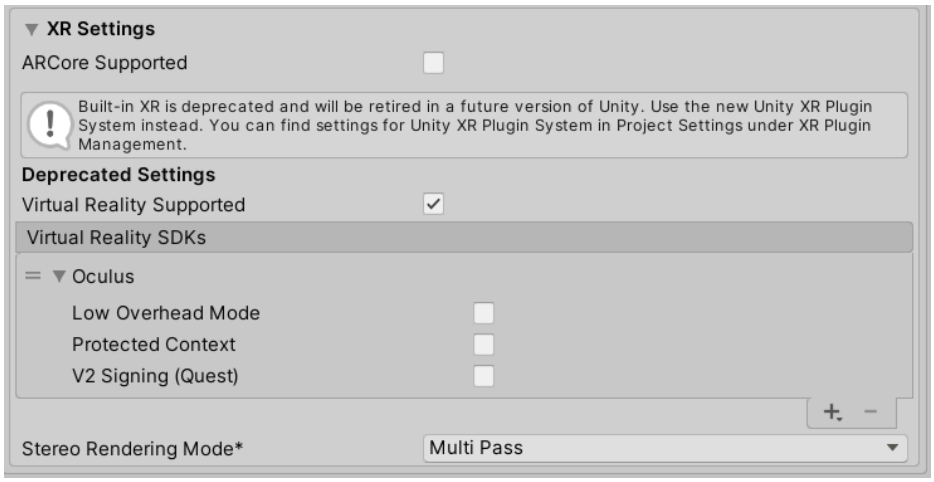

Figura 77. Añadir las gafas de realidad virtual Oculus como plataforma.

8 CONCLUSIONES Y LÍNEAS FUTURAS

En este proyecto se trata de simular las trayectorias de diferentes robots en Unity, pero mandándole los puntos desde Matlab. Esto se ha hecho en vista de que Matlab resulta ser una herramienta más potente y frecuente para el cálculo de trayectorias; sin embargo, la comunicación entre ambos no es sencilla. El mayor inconveniente, respecto a la comunicación, ha surgido al poder mandar únicamente un mensaje string con toda la información. Para resolver este problema se tuvo que realizar un string de gran tamaño y utilizar caracteres para diferenciar las partes. Esto supuso a su vez tener que procesar el mensaje una vez fuese recibido por Visual Studio.

Por otro lado, para realizar el movimiento de los robots, hubo diversos problemas. Entre ellos estaba el hecho de que cada objeto de Unity lleva un eje de coordenadas intrínseco que es el que se tiene que manejar para el movimiento y que no tiene por qué coincidir con el Gizmo de la Ventana de Escena. Por ello, había que coordinar ambos ejes y hacer los cambios oportunos para que la trayectoria que entrase fuese la que acabase visualizándose, aun con ejes cambiados. Otro problema parecido apareció en los terrestres, a la hora de girarlos. El eje anteriormente comentado, también giraba con el objeto, por lo que cada vez que el robot giraba era necesario actualizar el vector dirección.

El sistema tiempo real en el ambiente de Unity resultó un inconveniente. El objetivo era visualizar la trayectoria, pero el robot saltaba al final directamente. Por lo que se tuvo que poner un sistema de traslados a tiempo real que fuesen recorriendo la matriz de puntos a la vez que se iban moviendo hacia ellos. Esto también produjo otro inconveniente y era el paso de los robots, tenía que ser un paso lo suficientemente pequeño que no le hiciese dar movimientos bruscos y que evitase que se pasase de las coordenadas fijadas (algo que puede hacer que los robots se chocasen con otros objetos, que tuviesen que volver a las coordenadas fijadas dando marcha atrás o que el movimiento tuviese ruido). Sin embargo, tampoco podía ser muy pequeño, pues el tiempo de la trayectoria se vería afectado.

Para plantear la ampliación de la recreación se han tenido en cuenta diversos factores como la visualización y la funcionalidad para plantear diferentes puntos como líneas futuras.

Una posible mejora de visualización sería añadir una animación a las tuberías, con el objetivo de poder visualizar la temperatura del fluido según va recorriendo el circuito de tuberías. Además, una alternativa para añadir realismo a la recreación sería el de agregar sombras a las nubes. Respecto a la Plata Termosolar se podría continuar con su desarrollo, añadiendo más equipos como puede ser la sala de operadores donde se tendría la posibilidad de ver el SCADA[15].

Respecto a la funcionalidad del se podría dotar al programa de una funcionalidad más completa, añadiendo modelos de robots y haciendo su cantidad variable. Además se podría añadir algoritmos para calcular rutas desde el propio programa. Otra opción de mejora es crear una interfaz con diferentes menús donde se dé la posibilidad de variar el número de lazos o enseñar un mapa de la planta.

A su vez, otra posibilidad sería desarrollar la aplicación sobre las gafas de realidad virtual más extensivamente, aportando modalidades que permitiesen tener diferentes puntos de visión. Un ejemplo sería poder tener el punto de vista desde dentro de un vehículo no tripulado. Para que la movilidad dentro de la simulación fuese más cómoda, se podrían añadir menús que permitiesen una navegación flexible entre diferentes puntos de visión o entre distintos robots.

[15] "*Supervisory Control and Data Acquisition*", software para ordenadores que permite controlar y supervisar procesos industriales a distancia.

REFERENCIAS

[1] D. Erosa García, «Qué es Unity.», *OpenWebinars*, 2019. https://openwebinars.net/blog/que-es-unity/.

[2] «Unity Manual». [En línea]. Disponible en: https://docs.unity3d.com/Manual/UnityManual.html.

[3] Unity, «Unity's Interface». https://docs.unity3d.com/Manual/LearningtheInterface.html.

[4] EducacionIT, «Diferencia entre lenguajes de scripting, lenguajes de marcado y lenguajes de programación», *EducacionIT*. https://blog.educacionit.com/2018/12/26/diferencia-entre-lenguajes-de-scripting-lenguajes-de-marcado-y-lenguajes-de-programacion/.

[5] Unity, «Creating and Using Scripts», *Unity*. https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html.

[6] D. Strauss, *Getting Started with Visual Studio 2019*. Apess.

[7] Microsoft, «Le damos la bienvenida a Visual Studio.», *Microsoft*. https://docs.microsoft.com/es-es/visualstudio/get-started/visual-studio-ide?view=vs-2019.

[8] Á. Robledano, «Qué es TCP/IP», *OpenWebinars*, jun. 18, 2019. https://openwebinars.net/blog/que-es-tcpip/.

[9] F. Cesarini y S. Thompson, *Erlang Programming: A Concurreent Approach to Software Development*, Primera. O'Reilly.

[10] A. Flores, «Energía solar termoeléctrica o termosolar». https://www.solarweb.net/termosolar.php.

[11] J. M. Tapia, «Planta Termosolar CCP», Proyecto de Fin de Grado, Escuela Técnica Superior de Ingeniería, Sevilla, 2013.

[12] A. Martínez, «Tecnología térmica solar: cilindro parabólica.» https://desenchufados.net/tecnologia-termica-solar-cilindro-parabolica/.

[13] Marquesado Solar, «COLECTORES: Espejos parabólicos y absorbedores», *Marquesado Solar*. https://marquesadosolar.com/es/campo-solar/.

[14] M. del C. Benítez Román, «Análisis de pérdidas térmicas en el sistema HTF y almacenamiento de centrales termosolares CCP.», Escuela Técnica Superior de Ingeniería.

[15] OPEXenergy, «El Sistema HTF», *OPEXenergy*. http://opex-energy.com/termosolares/sistema_htf_termosolar.html.

[16] N. Connor, «¿Qué es el sobrecalentamiento y el recalentamiento?» https://www.thermal-engineering.org/es/que-es-el-sobrecalentamiento-y-el-recalentamiento-ciclo-de-rankine-definicion/.

[17] Renovetec, «Ciclo Termodinámico de la turbina de gas», *Ciclos Combinados*. http://www.cicloscombinados.com/index.php/el-ciclo-brayton.

[18] M. Berenguel y F. Rodríguez Rubio, *Advanced Control of Solar*. Springer-Verlag London.

[19] A. Sánchez, A. Gallego, J. M. Escaño, y E. Fernandez, «Adaptive incremental state space MPC for collector defocusing of a parabolic trough plant».

[20] A. Sánchez, A. Gallego, J. M. Escaño, y E. Fernandez, «Temperature homogenization of a solar trough field for performance improvement».

[21] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, y E. Cayirci, «Wireless Sensor Networks: A Survey.»

[22] P. Ghosh, A. Gasparri, J. Jin, y B. Krishnamachari, *Robotic Wireless Sensor Networks*. Springer.

GLOSARIO

2D: Espacio bidimensional, 20
3D: Espacio tridimensional, 20
apk: es una extensión que indica que es un paquete para el sistema operativo Android, 64
BAP: Bomba de Alta Presión, 52
GameObjects: Son objetos que se adhieren al proyecto y lo conforman., 23
HTF: Sistema de fluido térmico., 47
Mesh: Se trata de una malla con vértices, aristas y caras que se utilizan en una representación poligonal para definir una forma tridimensional., 28
PreFabs: tipo de activo que permite almacenar un objeto completamente con componentes y propiedades., 53
script: programa en lenguaje de scripting, 40
USB: Bus de comunicaciones estándar, 65

ANEXO I: CÓDIGO MATLAB

En el código de Matlab se cargarán los datos desde una matriz y se mandarán, por comunicación TCP/IP, a Visual Studio. El programa cuenta con unas limitaciones, como son el número máximo de trayectorias de robot a mandar. En el caso de que el número de robots sea mayor que 10, imprimirá un error y no mandará dato alguno. A su vez, identificará que trayectorias pertenecen a robots terrestres y cuales a los aéreos. Esta información le será útil a Visual Studio para decidir a qué robot mandar las coordenadas.

Otro aspecto importante a tener en cuenta es que la matriz DataRobots.mat, donde se deberán cargar todos los datos, deberá tener un orden específico. Para las filas, de menor a mayor, deberá aportar el número de robot al que pertenece el dato, la coordenada x, la coordenada y, la coordenada z, la batería y el tiempo. En las columnas se deberán añadir todas las coordenadas de punto, teniendo que ser la misma cantidad para todos los robots.

Para una mejor compresión, se va a adjuntar un Diagrama de flujo (ver Figura 78).

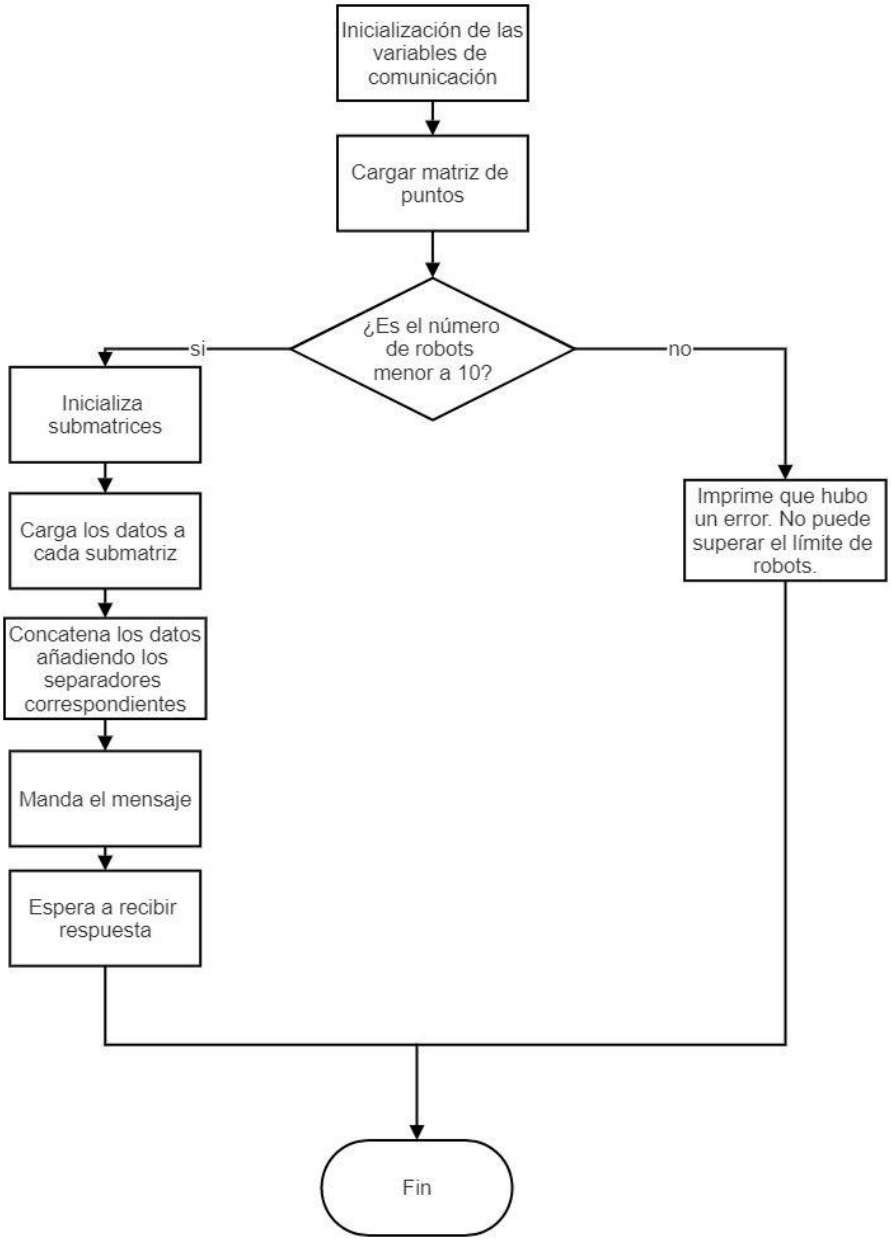

Figura 78. Diagrama de flujo del programa hecho en Matlab.

```
clc
clear all

tcpipClient = tcpip('127.0.0.1',55001,'NetworkRole','Client');
tcpipServer = tcpip('0.0.0.0',55000,'NetworkRole','Server');

tcpipClient.OutputBufferSize = 2000000^(10);

load('DatosRobots.mat');


[m,n]=size(DataRobots);
nrobot=DataRobots(i,j,n);

if nrobot<=10

    total=n/nrobot;

    for i=1:nrobot
        robot(i,:,:)=zeros(5,total);
    end

    i=1;
    j=1;

    while(i<(nrobot+1))
            t=1;
        while(j<n && DataRobots(1,j)==i)
            for z=1:5
               robot(i,z,t)=DataRobots(z+1,j);
            end
            t=t+1;
            j=j+1;
        end
        i=i+1;
    end

    set(tcpipClient,'Timeout',90);
    set(tcpipServer,'Timeout',90);
    data=membrane(1);

    for t=1:total
            if robot(1,3,total)>0
                band=1;
            end
    end

    msg=[sprintf('%d',band)];

    for i=1:nrobot

        if(i>1)
            for t=1:total
                if robot(i,3,total)>0
                    band=1;
                end
            end
```

```
            msg=[msg sprintf('/') sprintf('%d',band)];
        end

        msgx=[sprintf('%.3f;',robot(i,1,1))];

        for j=1:total
            msgx=[msgx sprintf('%.3f;',robot(i,1,j))];
        end

        msgy=[sprintf('%.3f;',robot(i,2,1))];

        for j=1:total
            msgy=[msgy sprintf('%.3f;',robot(i,2,j))];
        end

        msgz=[sprintf('%.3f;',robot(i,3,1))];

        for j=1:total
            msgz=[msgz sprintf('%.3f;',robot(i,3,j))];
        end

        msgb=[sprintf('%.3f;',robot(i,4,1))];

        for j=1:total
            msgb=[msgb sprintf('%.3f;',robot(i,4,j))];
        end

        msgt=[sprintf('%.3f;',robot(i,5,1))];

        for j=1:total
            msgt=[msgt sprintf('%.3f;',robot(i,5,j))];
        end

        msg=[msg sprintf('*') msgx];
        msg=[msg sprintf('_') msgy];
        msg=[msg sprintf('_') msgz];
        msg=[msg sprintf('_') msgb];
        msg=[msg sprintf('_') msgt];

    end

    fopen(tcpipClient);
    fwrite(tcpipClient,msg);
    fclose(tcpipClient);


    fopen(tcpipServer);
    rawData=fread(tcpipServer,14,'char');
    for i=1:8
     rawwData(i)=char(rawData(i));
    end
    fclose(tcpipServer);
    else
        print("El maximo es 10");
    end
```

ANEXO II: CÓDIGO VISUAL STUDIO

A. Script Lectura

El script de Lectura tendrá que en primer lugar localizar todos los objetos antes de que se produzca la comunicación, y lo hará mediante la comunicación explicada en el apartado 3.5. También deberá establecer la comunicación con Matlab, explicada en el apartado 4. Una vez que se haya recibido el mensaje resultante de la interacción con Matlab, se procederá con el procesamiento del mensaje. En primer lugar, se deberá descomponer de manera de que queden separadas las cadenas de cada robot. Para ello, se realizará el corte con el con el uso de una barra transversal "/", previamente introducido en Matlab como se puede ver en el

. Una vez se tenga hecha esta división se comenzará a tratar cada cadena de datos de cada robot por separado. En primer lugar, se deberá dividir la cadena resultante por el carácter "*" para poder así separar el dato que informe si se trata de un vehículo aéreo o terrestre con el resto de parámetros que proporcionan los puntos de la trayectoria. Con el resto de información se tiene que seguir trabajando para descomponerla; dentro de la información de cada robot, se dividirán las coordenadas por "_", por lo que podrán diferenciarse las coordenadas x, las coordenadas y, las coordenadas z, el tiempo y la batería. Para acabar de separar, cada cadena de coordenada tendrá cada uno de sus valores divido por ";" consiguiendo finalmente formar un vector con todos los valores separados. Tras acabar el tratamiento de datos, se mandará a un vehículo terrestre o aéreo, en función del dato que se haya mandado de Matlab y que previamente se adquiriera con el procesamiento.

Para un mejor entendimiento se agregará el código y un diagrama de flujo (ver Figura 79).

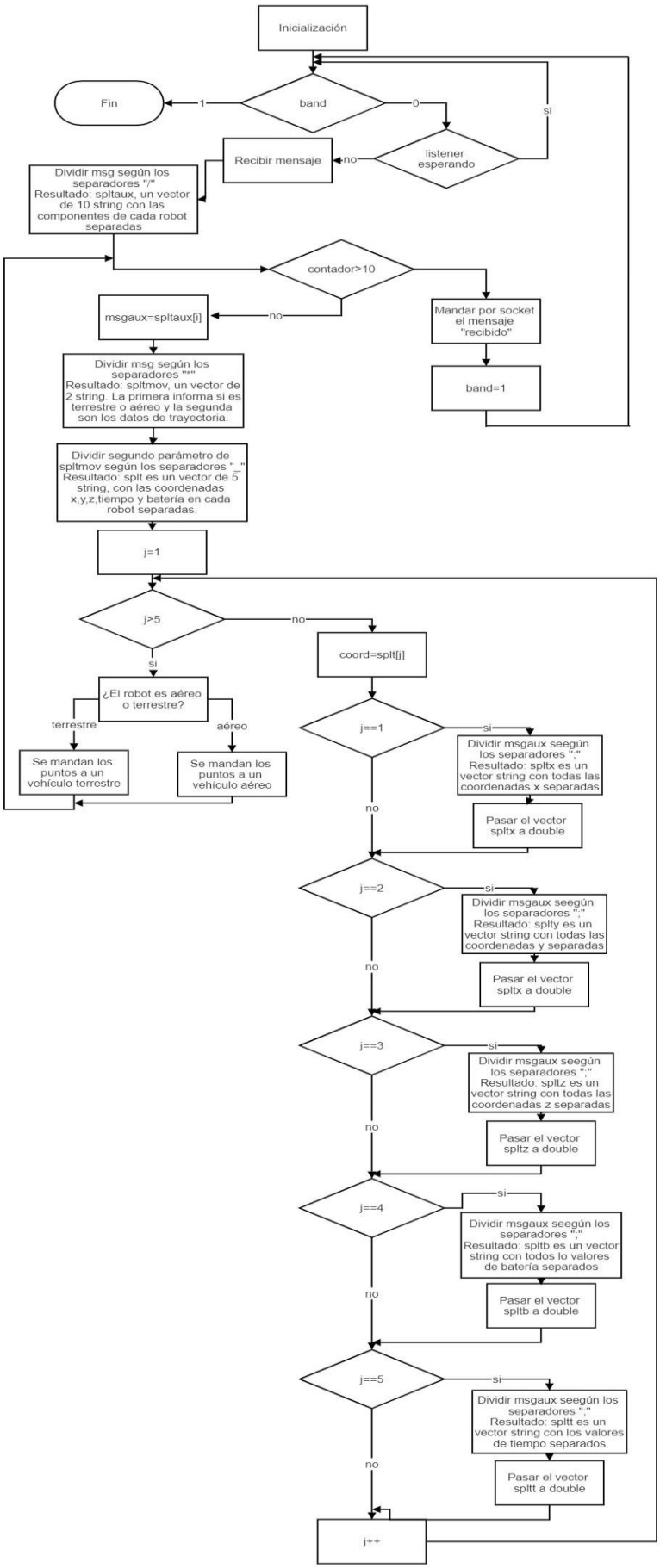

Figura 79. Diagrama de flujo del script Lecturas.

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Lectura : MonoBehaviour
{
    TcpListener listener;
    private String msg;

    internal Boolean socketReady = false;
    TcpClient mySocket;
    NetworkStream theStream;
    StreamWriter theWriter;
    StreamReader theReader;
    String Host = "localhost";
    Int32 Port = 55000;

    private string[] splt;
    private string[] spltx;
    private string[] splty;
    private string[] spltz;
    private string[] spltb;
    private string[] spltt;
    private string[] spltaux;
    private int length;
    private int longi;
    public double[] coordx;
    public double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;
    private double number;
    private double[][] auxx;

    int i;
    int j = 1;
    int contador=1;
    int band=0;

    private Robot2 _robot2;
    private Robot1 _robot1;
    private Robot3 _robot3;
    private Robot4 _robot4;
    private Robot5 _robot5;
    private Robot6 _robot6;
    private Robot7 _robot7;
    private Robot8 _robot8;
    private Robot9 _robot9;
```

```
private Robot10 _robot10;
private Robot11 _robot11;
private Robot12 _robot12;
private Robot13 _robot13;
private Robot14 _robot14;
private Robot15 _robot15;
private Robot16 _robot16;
private Robot17 _robot17;
private Robot18 _robot18;
private Robot19 _robot19;
private Robot20 _robot20;

void Start()
{
    _robot1 = GameObject.FindWithTag("Robot1").GetComponent<Robot1>();
    _robot2 = GameObject.FindWithTag("Robot2").GetComponent<Robot2>();
    _robot3 = GameObject.FindWithTag("Robot3").GetComponent<Robot3>();
    _robot4 = GameObject.FindWithTag("Robot4").GetComponent<Robot4>();
    _robot5 = GameObject.FindWithTag("Robot5").GetComponent<Robot5>();
    _robot6 = GameObject.FindWithTag("Robot6").GetComponent<Robot6>();
    _robot7 = GameObject.FindWithTag("Robot7").GetComponent<Robot7>();
    _robot8 = GameObject.FindWithTag("Robot8").GetComponent<Robot8>();
    _robot9 = GameObject.FindWithTag("Robot9").GetComponent<Robot9>();
    _robot10 = GameObject.FindWithTag("Robot10").GetComponent<Robot10>();
    _robot11 = GameObject.FindWithTag("Robot11").GetComponent<Robot11>();
    _robot12 = GameObject.FindWithTag("Robot12").GetComponent<Robot12>();
    _robot13 = GameObject.FindWithTag("Robot13").GetComponent<Robot13>();
    _robot14 = GameObject.FindWithTag("Robot14").GetComponent<Robot14>();
    _robot15 = GameObject.FindWithTag("Robot15").GetComponent<Robot15>();
    _robot16 = GameObject.FindWithTag("Robot16").GetComponent<Robot16>();
    _robot17 = GameObject.FindWithTag("Robot17").GetComponent<Robot17>();
    _robot18 = GameObject.FindWithTag("Robot18").GetComponent<Robot18>();
    _robot19 = GameObject.FindWithTag("Robot19").GetComponent<Robot19>();
    _robot20 = GameObject.FindWithTag("Robot20").GetComponent<Robot20>();

    listener = new TcpListener(IPAddress.Parse("127.0.0.1"), 55001);
    listener.Start();
}
public void Update()
{
    if(band==0)
    {
        llamada();
    }
}

public void llamada()
{
    if (!listener.Pending())
    {
    }
    else
    {
        while (listener.Pending())
        {
```

```
TcpClient client = listener.AcceptTcpClient();
NetworkStream ns = client.GetStream();
StreamReader reader = new StreamReader(ns);
msg = reader.ReadToEnd();
setupSocket();
spltaux = msg.Split(char.Parse("/"));
foreach (string msgaux in spltaux)
{
    spltcond = msgaux.Split(char.Parse("*"));
    foreach(string spltmov in spltcond)
    {
        if(param==1)
        {
            aereo = Convert.ToInt32(spltmov);
            param = param + 1;
        }
        else
        {
            splt = spltmov.Split(char.Parse("_"));
            length = splt.Length;
            j = 1;
            foreach (string coord in splt)
            {

                length = coord.Length;
                if (j == 1)
                {
                    coordx = new double[length];
                    spltx = coord.Split(char.Parse(";"));
                    longi = spltx.Length;
                    i = 1;
                    foreach (string num in spltx)
                    {
                        if (i < longi)
                        {
                            number = Convert.ToDouble(num);
                            number = number / 1000;
                            coordx[i] = number;
                            i++;
                        }
                    }
                }
                if (j == 2)
                {
                    coordy = new double[length];
                    splty = coord.Split(char.Parse(";"));
                    i = 1;
                    longi = splty.Length;
                    foreach (string num in splty)
                    {
                        if (i < longi)
                        {
                            number = Convert.ToDouble(num);
                            number = number / 1000;
```

```
                coordy[i] = number;
                i++;
            }
        }
    }
    if (j == 3)
    {
        coordz = new double[length];
        spltz = coord.Split(char.Parse(";"));
        longi = spltz.Length;
        i = 1;
        foreach (string num in spltz)
        {
            if (i < longi)
            {
                number = Convert.ToDouble(num);
                number = number / 1000;
                coordz[i] = number;
                i++;
            }
        }
    }
    if (j == 4)
    {
        coordb = new double[length];
        spltb = coord.Split(char.Parse(";"));
        longi = spltb.Length;
        i = 1;
        foreach (string num in spltb)
        {
            if (i < longi)
            {
                number = Convert.ToDouble(num);
                number = number / 1000;
                coordb[i] = number;
                i++;
            }
        }
    }
    if (j == 5)
    {
        coordt = new double[length];
        spltt = coord.Split(char.Parse(";"));
        longi = spltt.Length;
        i = 1;
        foreach (string num in spltt)
        {
            if (i < longi)
            {
                number = Convert.ToDouble(num);
                number = number / 1000;
                coordt[i] = number;
                i++;
            }
        }
```

```
                                }
                                j++;
                            }
                            if (aereo==0)
                            {
                                if (contadorter == 1)
                                {
                                    _robot1.movimiento(coordx, coordy,
coordz, coordb, coordt);
                                }
                                if (contadorter == 2)
                                {
                                    _robot2.movimiento(coordx, coordy,
coordz, coordb, coordt);
                                }
                                if (contadorter == 3)
                                {
                                    _robot3.movimiento(coordx, coordy,
coordz, coordb, coordt);
                                }
                                if (contadorter == 4)
                                {
                                    _robot11.movimiento(coordx, coordy,
coordz, coordb, coordt);
                                }
                                if (contadorter == 5)
                                {
                                    _robot12.movimiento(coordx, coordy,
coordz, coordb, coordt);
                                }
                                if (contadorter == 6)
                                {
                                    _robot13.movimiento(coordx, coordy,
coordz, coordb, coordt);
                                }
                                if (contadorter == 7)
                                {
                                    _robot14.movimiento(coordx, coordy,
coordz, coordb, coordt);
                                }
                                if (contadorter == 8)
                                {
                                    _robot15.movimiento(coordx, coordy,
coordz, coordb, coordt);
                                }
                                if (contadorter == 9)
                                {
                                    _robot16.movimiento(coordx, coordy,
coordz, coordb, coordt);
                                }
                                if (contadorter == 10)
                                {
                                    _robot17.movimiento(coordx, coordy,
coordz, coordb, coordt);
```

```
                    }
                    contadorter = contadorter + 1;
                }
                else
                {
                    if (contadorar == 1)
                    {
                        _robot4.movimiento(coordx, coordy,
coordz, coordb, coordt);
                    }
                    if (contadorar == 2)
                    {
                        _robot5.movimiento(coordx, coordy,
coordz, coordb, coordt);
                    }
                    if (contadorar == 3)
                    {
                        _robot6.movimiento(coordx, coordy,
coordz, coordb, coordt);
                    }
                    if (contadorar == 4)
                    {
                        _robot7.movimiento(coordx, coordy,
coordz, coordb, coordt);
                    }
                    if (contadorar == 5)
                    {
                        _robot8.movimiento(coordx, coordy,
coordz, coordb, coordt);
                    }
                    if (contadorar == 6)
                    {
                        _robot9.movimiento(coordx, coordy,
coordz, coordb, coordt);
                    }
                    if (contadorar == 7)
                    {
                        _robot10.movimiento(coordx, coordy,
coordz, coordb, coordt);
                    }
                    if (contadorar == 8)
                    {
                        _robot18.movimiento(coordx, coordy,
coordz, coordb, coordt);
                    }
                    if (contadorar == 9)
                    {
                        _robot19.movimiento(coordx, coordy,
coordz, coordb, coordt);
                    }
                    if (contadorar == 10)
                    {
                        _robot20.movimiento(coordx, coordy,
coordz, coordb, coordt);
                    }
```

```
                                contadorar = contadorar + 1;
                            }
                            param = 1;
                        }
                    }
                }
                band = 1;
            }
        }
    }
    public void setupSocket()
    {
        try
        {
            mySocket = new TcpClient(Host, Port);
            theStream = mySocket.GetStream();
            theWriter = new StreamWriter(theStream);
            theReader = new StreamReader(theStream);
            socketReady = true;
            Byte[] sendBytes = Encoding.UTF8.GetBytes("recibido");
            mySocket.GetStream().Write(sendBytes, 0, sendBytes.Length);
        }
        catch (Exception e)
        {
            Debug.Log("Socket error: " + e);
        }
    }
}
```

B. Script Vehículo no tripulado aéreo

Los scripts de robots estarán divididos en aéreos y terrestres. A pesar de que su comportamiento es prácticamente el mismo, los terrestres necesitarán tener el suplemento de que al ser un vehículo de cuatro ruedas, necesitarán girar a la derecha o a la izquierda en el caso de que su movimiento sea por el eje x y no únicamente por el eje z. Por ello, vamos a empezar explicando el funcionamiento del script aéreo y se acabará explicando las diferencias existentes con el terrestre.

En primer lugar, se necesitará una función, la que se ha llamado movimiento, que será la que esté en comunicación con el script Lectura. En esta función se esperará a recibir las distintas coordenadas, el tiempo y la batería; y una vez recibidas, se moverá el robot a la posición. Una vez recibida, activará la variable cambio, que afectará a la función comparación. Esta función será llamada cada vez que se ejecute Update; sin embargo, solamente realizará su función en el caso de que la variable cambio esté activada. Con esta función se encargará de recorrer el vector de coordenadas y averiguar si hay un nuevo cambio. Una vez encuentre un nuevo cambio, se desactivará la variable cambio para que la función comparación no se vuelva a ejecutar; además, se activará la variable mov.

Esta variable mov, activará otra parte del script, donde se buscará realizar el movimiento del robot. Las funciones que dan movimiento a los objetos, como transform.Translate utilizan un vector dirección, una velocidad y el tiempo para moverse. Es por eso que para que se mueva durante el tiempo que se especifique, se deberá meter en primer lugar dentro de una función que esté siendo constantemente llamada. El comportamiento se otorgará con el uso de condiciones. Por ello, se pedirá que se mueva hasta que la variable mov sea cero, lo que indicará que ha llegado a la coordenada que se buscaba.

Para tener la correcta dirección, la primera vez que se entre dentro de mov, se deberá considerar que la dirección es correcta en función de las coordenadas nuevas a las que hay que dirigirse. Además, esto se tendrá que hacer la primera vez únicamente. Es por eso que apoyándose en la variable actualizar, exclusivamente se entrará la primera vez en el código que crea la variable direction. Una vez se haya calculado, la variable

actualizar pasará a valer cero no volviendo a ser uno hasta que se haya conseguido llegar a las coordenadas idóneas y se acabe el bucle mov, por lo que deberá reiniciarse. La velocidad se calculará dividiendo los metros a recorrer por la diferencia del tiempo entre la posición buscada y la anterior.

Una vez se haya conseguido la dirección, se procede a mover el objeto con transform.Translate. A continuación, se deberá comprobar si el robot ha llegado a las coordenadas correctas. En el caso de que se haya llegado, en la coordenada correspondiente de dirección se hará cero para que en el próximo movimiento este eje no se vea afectado en el movimiento. En el caso de que todos los elementos del vector direction sea cero, el robot habrá llegado y se reiniciarán todo los valores, volviéndose a activar la variable cambio para continuar con la siguiente coordenada.

Para una mejor comprensión se va a adjuntar el código de los drones y también un diagrama de flujo, (ver Figura 80).

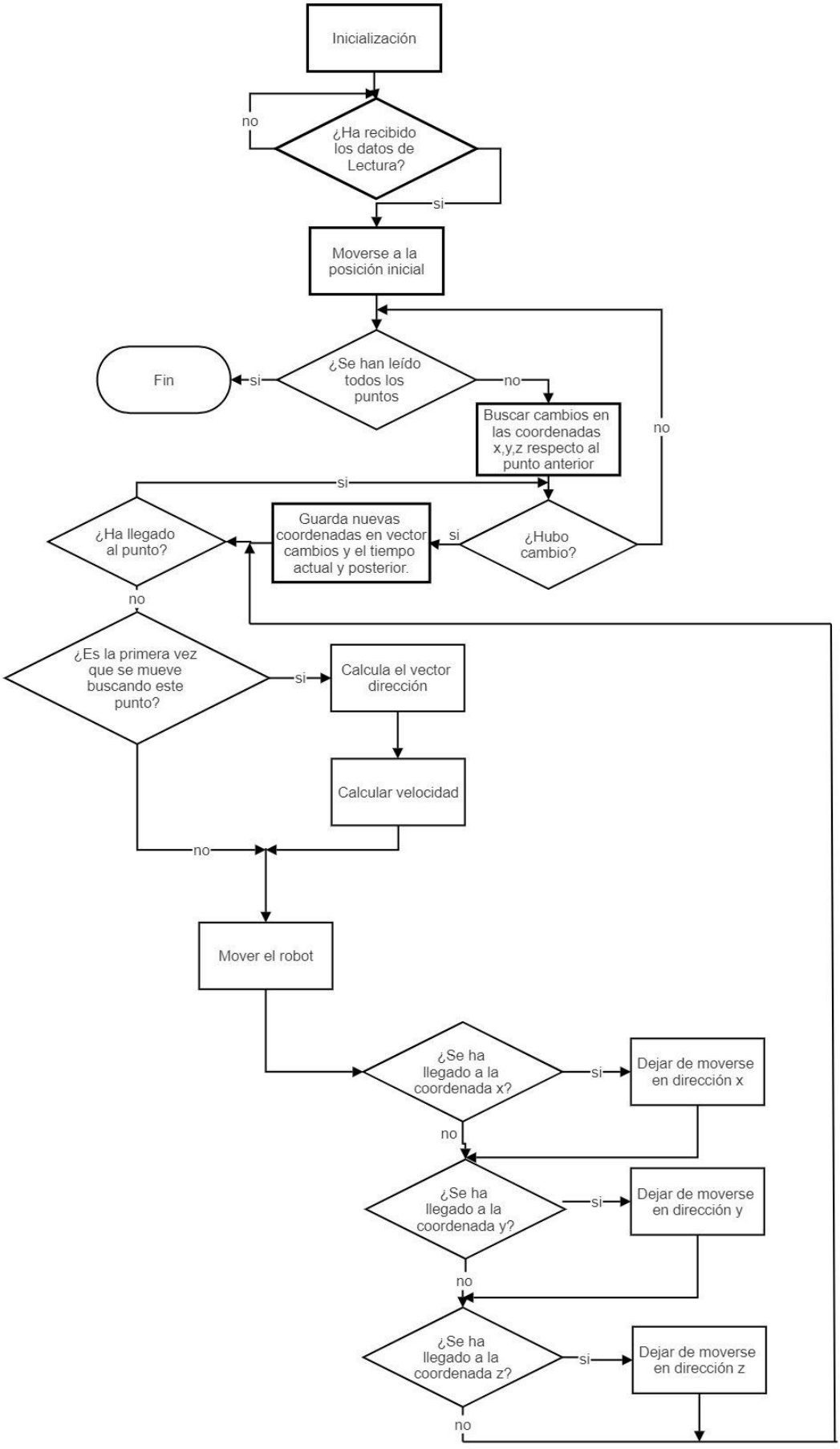

Figura 80. Diagrama de flujo Vehículo aéreo no tripulado.

Script Robot 4

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot4 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cambio = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    public void Update()
    {
        comparacion();
        actualizacion();
    }

    public void actualizacion()
    {
        if (mov == 1)
        {
            if (actualizar == 1)
            {
                if (Math.Round(cambios.x - transform.position.x, 1) > 0.1f)
                {
                    direction.z = 0.2f;
```

```
                    dist = Math.Abs(cambios.x - transform.position.x) + dist;
                }
                else
                {
                    direction.z = 0f;
                }
                if (Math.Round(cambios.x - transform.position.x, 1) < 0.1f)
                {
                    direction.z = -0.2f;
                    dist = Math.Abs(cambios.x - transform.position.x) + dist;
                }
                if (Math.Round(cambios.y - transform.position.y, 1) > 0.1f)
                {
                    direction.y = 0.1f;
                    dist = Math.Abs(cambios.y - transform.position.y) + dist;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.y - transform.position.y, 1) < 0.1f)
                {
                    direction.y = -0.1f;
                    dist = Math.Abs(cambios.y - transform.position.y) + dist;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.1f)
                {
                    direction.x = -0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                else
                {
                    direction.x = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.1f)
                {
                    direction.x = 0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                actualizar = 0;
                speed = dist/ (tnuevo - t);
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (direction.z == 0.2f)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f)
            {
```

```
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.1f)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.1f)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cambio = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }

    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
```

```
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cambio = 1;
        mov = 0;

    }

    public void comparacion()
    {
        while (cambio == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cambio = 0; ;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
```

```
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if(Math.Abs(transform.position.z-coordy[i])>0.2f)
                        {
                            aux = Convert.ToSingle(coordy[i]); ;
                            cambios.z = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.z = transform.position.z;
                        }
                    }
                    if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.y - coordz[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordz[i]); ;
                            cambios.y = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cambio == 0)
                    {
                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    else
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
```

```
}
```

Script Robot 5

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot5 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cambio = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    public void Update()
    {
        comparacion();
        actualizacion();
    }

    public void actualizacion()
    {
        if (mov == 1)
        {
            if (actualizar == 1)
            {
                if (Math.Round(cambios.x - transform.position.x, 1) > 0.1f)
```

```
                {
                    direction.z = 0.2f;
                    dist = Math.Abs(cambios.x - transform.position.x) + dist;
                }
                else
                {
                    direction.z = 0f;
                }
                if (Math.Round(cambios.x - transform.position.x, 1) < 0.1f)
                {
                    direction.z = -0.2f;
                    dist = Math.Abs(cambios.x - transform.position.x) + dist;
                }
                if (Math.Round(cambios.y - transform.position.y, 1) > 0.1f)
                {
                    direction.y = 0.1f;
                    dist = Math.Abs(cambios.y - transform.position.y) + dist;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.y - transform.position.y, 1) < 0.1f)
                {
                    direction.y = -0.1f;
                    dist = Math.Abs(cambios.y - transform.position.y) + dist;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.1f)
                {
                    direction.x = -0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                else
                {
                    direction.x = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.1f)
                {
                    direction.x = 0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                actualizar = 0;
                speed = dist / (tnuevo - t);
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (direction.z == 0.2f)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f)
            {
```

```
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.1f)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.1f)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cambio = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }

    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
```

```
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cambio = 1;
        mov = 0;

    }

    public void comparacion()
    {
        while (cambio == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cambio = 0; ;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
```

```
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.z - coordy[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordy[i]); ;
                            cambios.z = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.z = transform.position.z;
                        }
                    }
                    if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.y - coordz[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordz[i]); ;
                            cambios.y = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cambio == 0)
                    {
                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    else
                    {
                        j++;
                    }
                }
                i++;
            }
```

```
        }
    }
}
```

Script Robot 6

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot6 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cambio = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    public void Update()
    {
        comparacion();
        actualizacion();
    }

    public void actualizacion()
    {
        if (mov == 1)
        {
            if (actualizar == 1)
```

```
            {
                if (Math.Round(cambios.x - transform.position.x, 1) > 0.1f)
                {
                    direction.z = 0.2f;
                    dist = Math.Abs(cambios.x - transform.position.x) + dist;
                }
                else
                {
                    direction.z = 0f;
                }
                if (Math.Round(cambios.x - transform.position.x, 1) < 0.1f)
                {
                    direction.z = -0.2f;
                    dist = Math.Abs(cambios.x - transform.position.x) + dist;
                }
                if (Math.Round(cambios.y - transform.position.y, 1) > 0.1f)
                {
                    direction.y = 0.1f;
                    dist = Math.Abs(cambios.y - transform.position.y) + dist;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.y - transform.position.y, 1) < 0.1f)
                {
                    direction.y = -0.1f;
                    dist = Math.Abs(cambios.y - transform.position.y) + dist;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.1f)
                {
                    direction.x = -0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                else
                {
                    direction.x = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.1f)
                {
                    direction.x = 0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                actualizar = 0;
                speed = dist / (tnuevo - t);
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (direction.z == 0.2f)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
```

```
            }
            if (direction.z == -0.2f)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.1f)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.1f)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cambio = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
```

```
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cambio = 1;
        mov = 0;

    }

    public void comparacion()
    {
        while (cambio == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cambio = 0; ;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cambio = 0;
                        }
                        else
                        {
```

```
                cambios.x = transform.position.x;
            }
        }
        if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
        {
            aux = Convert.ToSingle(coordy[i]); ;
            cambios.z = aux;
            cambio = 0;
        }
        else
        {
            if (Math.Abs(transform.position.z - coordy[i]) >
0.2f)
            {
                aux = Convert.ToSingle(coordy[i]); ;
                cambios.z = aux;
                cambio = 0;
            }
            else
            {
                cambios.z = transform.position.z;
            }
        }
        if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
        {
            aux = Convert.ToSingle(coordz[i]); ;
            cambios.y = aux;
            cambio = 0;
        }
        else
        {
            if (Math.Abs(transform.position.y - coordz[i]) >
0.2f)
            {
                aux = Convert.ToSingle(coordz[i]); ;
                cambios.y = aux;
                cambio = 0;
            }
            else
            {
                cambios.y = transform.position.y;
            }
        }
        if (cambio == 0)
        {
            mov = 1;
            actualizar = 1;
            tnuevo = Convert.ToSingle(coordt[i]);
            t = Convert.ToSingle(coordt[i - j]);
            j = 1;
        }
        else
        {
            j++;
        }
```

```
                }
                i++;
            }
        }
    }
}
```

Script Robot 7

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot7 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cambio = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    public void Update()
    {
        comparacion();
        actualizacion();
    }
    public void actualizacion()
    {
        if (mov == 1)
```

```
        {
            if (actualizar == 1)
            {
                if (Math.Round(cambios.x - transform.position.x, 1) > 0.1f)
                {
                    direction.z = 0.2f;
                    dist = Math.Abs(cambios.x - transform.position.x) + dist;
                }
                else
                {
                    direction.z = 0f;
                }
                if (Math.Round(cambios.x - transform.position.x, 1) < 0.1f)
                {
                    direction.z = -0.2f;
                    dist = Math.Abs(cambios.x - transform.position.x) + dist;
                }
                if (Math.Round(cambios.y - transform.position.y, 1) > 0.1f)
                {
                    direction.y = 0.1f;
                    dist = Math.Abs(cambios.y - transform.position.y) + dist;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.y - transform.position.y, 1) < 0.1f)
                {
                    direction.y = -0.1f;
                    dist = Math.Abs(cambios.y - transform.position.y) + dist;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.1f)
                {
                    direction.x = -0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                else
                {
                    direction.x = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.1f)
                {
                    direction.x = 0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                actualizar = 0;

                speed = dist / (tnuevo - t);
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (direction.z == 0.2f)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
```

```
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.1f)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.1f)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cambio = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
```

```
    }

    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cambio = 1;
        mov = 0;

    }

    public void comparacion()
    {
        while (cambio == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cambio = 0; ;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cambio = 0;
                        }
                        else
```

```
                    {
                        cambios.x = transform.position.x;
                    }
                }
                if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordy[i]); ;
                    cambios.z = aux;
                    cambio = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.z - coordy[i]) >
0.2f)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cambio = 0;
                    }
                    else
                    {
                        cambios.z = transform.position.z;
                    }
                }
                if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordz[i]); ;
                    cambios.y = aux;
                    cambio = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.y - coordz[i]) >
0.2f)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cambio = 0;
                    }
                    else
                    {
                        cambios.y = transform.position.y;
                    }
                }
                if (cambio == 0)
                {
                    mov = 1;
                    actualizar = 1;
                    tnuevo = Convert.ToSingle(coordt[i]);
                    t = Convert.ToSingle(coordt[i - j]);
                    j = 1;
                }
                else
                {
```

```
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 8

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot8 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cambio = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    public void Update()
    {
        comparacion();
        actualizacion();
    }
    public void actualizacion()
```

```
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.1f)
            {
                direction.z = 0.2f;
                dist = Math.Abs(cambios.x - transform.position.x) + dist;
            }
            else
            {
                direction.z = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.1f)
            {
                direction.z = -0.2f;
                dist = Math.Abs(cambios.x - transform.position.x) + dist;
            }
            if (Math.Round(cambios.y - transform.position.y, 1) > 0.1f)
            {
                direction.y = 0.1f;
                dist = Math.Abs(cambios.y - transform.position.y) + dist;
            }
            else
            {
                direction.y = 0f;
            }
            if (Math.Round(cambios.y - transform.position.y, 1) < 0.1f)
            {
                direction.y = -0.1f;
                dist = Math.Abs(cambios.y - transform.position.y) + dist;
            }
            if (Math.Round(cambios.z - transform.position.z, 1) > 0.1f)
            {
                direction.x = -0.2f;
                dist = Math.Abs(cambios.z - transform.position.z) + dist;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.z - transform.position.z, 1) < 0.1f)
            {
                direction.x = 0.2f;
                dist = Math.Abs(cambios.z - transform.position.z) + dist;
            }
            actualizar = 0;

            speed = dist / (tnuevo - t);
        }
        transform.Translate(direction * speed * Time.deltaTime);
        if (direction.z == 0.2f)
        {
```

```
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.1f)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.1f)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cambio = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
```

```
            }
        }
    }

    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cambio = 1;
        mov = 0;

    }

    public void comparacion()
    {
        while (cambio == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cambio = 0; ;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
```

```
                cambio = 0;
            }
            else
            {
                cambios.x = transform.position.x;
            }
        }
        if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
        {
            aux = Convert.ToSingle(coordy[i]); ;
            cambios.z = aux;
            cambio = 0;
        }
        else
        {
            if (Math.Abs(transform.position.z - coordy[i]) >
0.2f)
            {
                aux = Convert.ToSingle(coordy[i]); ;
                cambios.z = aux;
                cambio = 0;
            }
            else
            {
                cambios.z = transform.position.z;
            }
        }
        if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
        {
            aux = Convert.ToSingle(coordz[i]); ;
            cambios.y = aux;
            cambio = 0;
        }
        else
        {
            if (Math.Abs(transform.position.y - coordz[i]) >
0.2f)
            {
                aux = Convert.ToSingle(coordz[i]); ;
                cambios.y = aux;
                cambio = 0;
            }
            else
            {
                cambios.y = transform.position.y;
            }
        }
        if (cambio == 0)
        {
            mov = 1;
            actualizar = 1;
            tnuevo = Convert.ToSingle(coordt[i]);
            t = Convert.ToSingle(coordt[i - j]);
            j = 1;
        }
```

```
                    else
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 9

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot9 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cambio = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    public void Update()
    {
        comparacion();
        actualizacion();
```

```
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.1f)
            {
                direction.z = 0.2f;
                dist = Math.Abs(cambios.x - transform.position.x) + dist;
            }
            else
            {
                direction.z = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.1f)
            {
                direction.z = -0.2f;
                dist = Math.Abs(cambios.x - transform.position.x) + dist;
            }
            if (Math.Round(cambios.y - transform.position.y, 1) > 0.1f)
            {
                direction.y = 0.1f;
                dist = Math.Abs(cambios.y - transform.position.y) + dist;
            }
            else
            {
                direction.y = 0f;
            }
            if (Math.Round(cambios.y - transform.position.y, 1) < 0.1f)
            {
                direction.y = -0.1f;
                dist = Math.Abs(cambios.y - transform.position.y) + dist;
            }
            if (Math.Round(cambios.z - transform.position.z, 1) > 0.1f)
            {
                direction.x = -0.2f;
                dist = Math.Abs(cambios.z - transform.position.z) + dist;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.z - transform.position.z, 1) < 0.1f)
            {
                direction.x = 0.2f;
                dist = Math.Abs(cambios.z - transform.position.z) + dist;
            }
            actualizar = 0;

            speed = dist / (tnuevo - t);
        }
        transform.Translate(direction * speed * Time.deltaTime);
```

```
            if (direction.z == 0.2f)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.1f)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.1f)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cambio = 1;
                cambios.y = 0;
                cambios.x = 0;
```

```
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cambio = 1;
        mov = 0;

    }

    public void comparacion()
    {
        while (cambio == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cambio = 0; ;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
```

```
                        cambios.x = aux;
                        cambio = 0;
                    }
                    else
                    {
                        cambios.x = transform.position.x;
                    }
                }
                if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordy[i]); ;
                    cambios.z = aux;
                    cambio = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.z - coordy[i]) >
0.2f)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cambio = 0;
                    }
                    else
                    {
                        cambios.z = transform.position.z;
                    }
                }
                if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordz[i]); ;
                    cambios.y = aux;
                    cambio = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.y - coordz[i]) >
0.2f)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cambio = 0;
                    }
                    else
                    {
                        cambios.y = transform.position.y;
                    }
                }
                if (cambio == 0)
                {
                    mov = 1;
                    actualizar = 1;
                    tnuevo = Convert.ToSingle(coordt[i]);
                    t = Convert.ToSingle(coordt[i - j]);
```

```
                    j = 1;
                }
                else
                {
                    j++;
                }
            }
            i++;
        }
    }
}
}
```

Script Robot 10

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot10 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cambio = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    public void Update()
    {
```

```
        comparacion();
        actualizacion();
    }

    public void actualizacion()
    {
        if (mov == 1)
        {
            if (actualizar == 1)
            {
                if (Math.Round(cambios.x - transform.position.x, 1) > 0.1f)
                {
                    direction.z = 0.2f;
                    dist = Math.Abs(cambios.x - transform.position.x) + dist;
                }
                else
                {
                    direction.z = 0f;
                }
                if (Math.Round(cambios.x - transform.position.x, 1) < 0.1f)
                {
                    direction.z = -0.2f;
                    dist = Math.Abs(cambios.x - transform.position.x) + dist;
                }
                if (Math.Round(cambios.y - transform.position.y, 1) > 0.1f)
                {
                    direction.y = 0.1f;
                    dist = Math.Abs(cambios.y - transform.position.y) + dist;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.y - transform.position.y, 1) < 0.1f)
                {
                    direction.y = -0.1f;
                    dist = Math.Abs(cambios.y - transform.position.y) + dist;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.1f)
                {
                    direction.x = -0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                else
                {
                    direction.x = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.1f)
                {
                    direction.x = 0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                actualizar = 0;
```

```
                speed = dist / (tnuevo - t);
            }

            transform.Translate(direction * speed * Time.deltaTime);

            if (direction.z == 0.2f)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.1f)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.1f)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
```

```
                mov = 0;
                cambio = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }

    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cambio = 1;
        mov = 0;

    }

    public void comparacion()
    {
        while (cambio == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cambio = 0; ;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cambio = 0;
                    }
```

```
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.z - coordy[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordy[i]); ;
                            cambios.z = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.z = transform.position.z;
                        }
                    }
                    if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.y - coordz[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordz[i]); ;
                            cambios.y = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cambio == 0)
```

```
                    {
                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    else
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 18

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot18 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cambio = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
```

```
private float dist = 0;
private int j = 1;

public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.1f)
            {
                direction.z = 0.2f;
                dist = Math.Abs(cambios.x - transform.position.x) + dist;
            }
            else
            {
                direction.z = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.1f)
            {
                direction.z = -0.2f;
                dist = Math.Abs(cambios.x - transform.position.x) + dist;
            }
            if (Math.Round(cambios.y - transform.position.y, 1) > 0.1f)
            {
                direction.y = 0.1f;
                dist = Math.Abs(cambios.y - transform.position.y) + dist;
            }
            else
            {
                direction.y = 0f;
            }
            if (Math.Round(cambios.y - transform.position.y, 1) < 0.1f)
            {
                direction.y = -0.1f;
                dist = Math.Abs(cambios.y - transform.position.y) + dist;
            }
            if (Math.Round(cambios.z - transform.position.z, 1) > 0.1f)
            {
                direction.x = -0.2f;
                dist = Math.Abs(cambios.z - transform.position.z) + dist;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.z - transform.position.z, 1) < 0.1f)
            {
                direction.x = 0.2f;
```

```
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                actualizar = 0;

                speed = dist / (tnuevo - t);
            }

            transform.Translate(direction * speed * Time.deltaTime);

            if (direction.z == 0.2f)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.1f)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.1f)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
```

```
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cambio = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }

    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cambio = 1;
        mov = 0;

    }

    public void comparacion()
    {
        while (cambio == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cambio = 0; ;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
```

```
                    aux = Convert.ToSingle(coordx[i]); ;
                    cambios.x = aux;
                    cambio = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.x - coordx[i]) >
0.2f)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cambio = 0;
                    }
                    else
                    {
                        cambios.x = transform.position.x;
                    }
                }
                if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordy[i]); ;
                    cambios.z = aux;
                    cambio = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.z - coordy[i]) >
0.2f)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cambio = 0;
                    }
                    else
                    {
                        cambios.z = transform.position.z;
                    }
                }
                if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordz[i]); ;
                    cambios.y = aux;
                    cambio = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.y - coordz[i]) >
0.2f)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cambio = 0;
                    }
                    else
```

```
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cambio == 0)
                    {
                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    else
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 19

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot19 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cambio = 0;
```

```
private int mov = 0;
private int length = 0;
private float aux;
private int actualizar = 0;
private float tnuevo = 0;
private float dist = 0;
private int j = 1;

public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.1f)
            {
                direction.z = 0.2f;
                dist = Math.Abs(cambios.x - transform.position.x) + dist;
            }
            else
            {
                direction.z = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.1f)
            {
                direction.z = -0.2f;
                dist = Math.Abs(cambios.x - transform.position.x) + dist;
            }
            if (Math.Round(cambios.y - transform.position.y, 1) > 0.1f)
            {
                direction.y = 0.1f;
                dist = Math.Abs(cambios.y - transform.position.y) + dist;
            }
            else
            {
                direction.y = 0f;
            }
            if (Math.Round(cambios.y - transform.position.y, 1) < 0.1f)
            {
                direction.y = -0.1f;
                dist = Math.Abs(cambios.y - transform.position.y) + dist;
            }
            if (Math.Round(cambios.z - transform.position.z, 1) > 0.1f)
            {
                direction.x = -0.2f;
                dist = Math.Abs(cambios.z - transform.position.z) + dist;
            }
            else
```

```
                {
                    direction.x = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.1f)
                {
                    direction.x = 0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                actualizar = 0;

                speed = dist / (tnuevo - t);
            }

            transform.Translate(direction * speed * Time.deltaTime);

            if (direction.z == 0.2f)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.1f)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.1f)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
```

```
            if (direction.x == -0.2f)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cambio = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }

    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cambio = 1;
        mov = 0;

    }

    public void comparacion()
    {
        while (cambio == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cambio = 0; ;
                    direction.x = 0;
                    direction.y = 0;
```

```
                direction.z = 0;
            }
            else
            {
                if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordx[i]); ;
                    cambios.x = aux;
                    cambio = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.x - coordx[i]) >
0.2f)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cambio = 0;
                    }
                    else
                    {
                        cambios.x = transform.position.x;
                    }
                }
                if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordy[i]); ;
                    cambios.z = aux;
                    cambio = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.z - coordy[i]) >
0.2f)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cambio = 0;
                    }
                    else
                    {
                        cambios.z = transform.position.z;
                    }
                }
                if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordz[i]); ;
                    cambios.y = aux;
                    cambio = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.y - coordz[i]) >
0.2f)
                    {
```

```
                            aux = Convert.ToSingle(coordz[i]); ;
                            cambios.y = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cambio == 0)
                    {
                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    else
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 20

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot20 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;
```

```
Vector3 direction = new Vector3(0, 0, 0);
Vector3 cambios = new Vector3(0, 0, 0);

private int i = 2;
private int cambio = 0;
private int mov = 0;
private int length = 0;
private float aux;
private int actualizar = 0;
private float tnuevo = 0;
private float dist = 0;
private int j = 1;

public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.1f)
            {
                direction.z = 0.2f;
                dist = Math.Abs(cambios.x - transform.position.x) + dist;
            }
            else
            {
                direction.z = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.1f)
            {
                direction.z = -0.2f;
                dist = Math.Abs(cambios.x - transform.position.x) + dist;
            }
            if (Math.Round(cambios.y - transform.position.y, 1) > 0.1f)
            {
                direction.y = 0.1f;
                dist = Math.Abs(cambios.y - transform.position.y) + dist;
            }
            else
            {
                direction.y = 0f;
            }
            if (Math.Round(cambios.y - transform.position.y, 1) < 0.1f)
            {
                direction.y = -0.1f;
                dist = Math.Abs(cambios.y - transform.position.y) + dist;
            }
            if (Math.Round(cambios.z - transform.position.z, 1) > 0.1f)
            {
```

```
                    direction.x = -0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                else
                {
                    direction.x = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.1f)
                {
                    direction.x = 0.2f;
                    dist = Math.Abs(cambios.z - transform.position.z) + dist;
                }
                actualizar = 0;

                speed = dist / (tnuevo - t);
            }

            transform.Translate(direction * speed * Time.deltaTime);

            if (direction.z == 0.2f)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.1f)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.1f)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f)
            {
```

```
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cambio = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }

    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cambio = 1;
        mov = 0;

    }

    public void comparacion()
    {
        while (cambio == 1)
        {
            if (i < length)
            {
```

```
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cambio = 0; ;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.z - coordy[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordy[i]); ;
                            cambios.z = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.z = transform.position.z;
                        }
                    }
                    if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
```

```
                        cambio = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.y - coordz[i]) >
0.2f)
                        {
                            aux = Convert.ToSingle(coordz[i]); ;
                            cambios.y = aux;
                            cambio = 0;
                        }
                        else
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cambio == 0)
                    {
                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    else
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

C. Script Vehículo no tripulado terrestre

El procedimiento del vehículo terrestre será prácticamente igual que el aéreo. La mayor diferencia estará dentro del bucle mov, donde se deberán de añadir dos variables: bandderecha y bandizquierda. Estas dos variables serán las que se activen cuando el robot terrestre gire a la izquierda o a la derecha. Estas variables se desactivarán a la hora de actualizar el siguiente movimiento ya que es cuando se debe preguntar si debe continuar girado o volver a ponerse en su dirección original, consiguiendo un movimiento más limpio de constantes giros.

Para una mejor comprensión se va a adjuntar el código de los drones y también un diagrama de flujo (ver Figura 81).

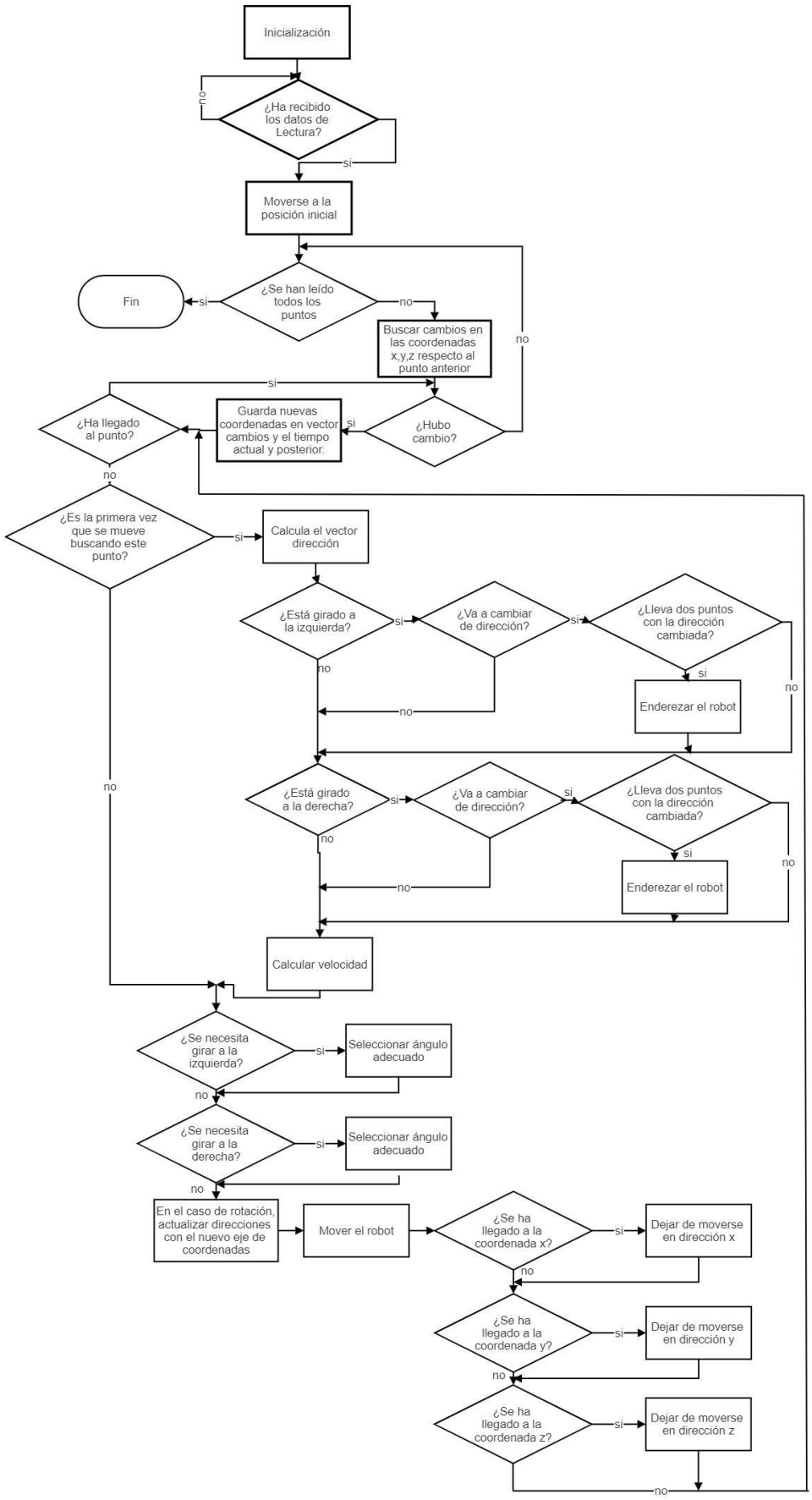

Figura 81. Diagrama de flujo vehículo terrestre no tripulado.

Script Robot 1

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot1 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cam = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    private float bandderecha;
    private float bandizquierda;
    private int contador;
    private int aviso;

    void Start()
    {
        bandderecha = 0;
        bandizquierda = 0;
        contador = 0;
        aviso = 0;
    }

    public void Update()
    {
```

```
        comparacion();
        actualizacion();
    }

    public void actualizacion()
    {
        if (mov == 1)
        {
            if (actualizar == 1)
            {
                if (Math.Round(cambios.x - transform.position.x, 1) > 0.05f)
                {
                    direction.x = -0.2f;
                }
                else
                {
                    direction.x = 0f;
                }
                if (Math.Round(cambios.x - transform.position.x, 1) < 0.05f)
                {
                    direction.x = 0.2f;
                }
                if (Math.Abs(cambios.y - transform.position.y) > 0.5f)
                {
                    if (cambios.y > transform.position.y)
                    {
                        direction.z = 0.2f;
                    }
                    else
                    {
                        direction.z = 0f;
                    }
                    if (cambios.y < transform.position.y)
                    {
                        direction.z = -0.2f;
                    }
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.05f)
                {
                    direction.y = 0.2f;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.05f)
                {
                    direction.y = -0.2f;
                }
                actualizar = 0;
                if (bandizquierda == 1)
                {
                    if (direction.x == 0 || direction.x == -0.2f)
                    {
```

```
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandizquierda = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                if (bandderecha == 1)
                {
                    if (direction.x == 0 || direction.x == 0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandderecha = 0;
                            contador = 0;
                        }
```

```
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                dist = Math.Abs(cambios.z + cambios.y + cambios.x -
transform.position.x - transform.position.y - transform.position.z);
                speed = dist / (tnuevo - t);
                if (speed > 10 && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 0.1f;
                }
                if (speed < 0.1f && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 10;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && direction.z == 0
&& bandderecha == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 100.0f,
0.0f);
```

```
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0.0f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 90.0f,
0.0f);
                }
                bandizquierda = 1;
                aviso = 1;
            }
            if (direction.x == -0.2f && bandderecha == 0 && direction.z == 0
&& bandizquierda == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -90.0f,
0.0f);
                }
                bandderecha = 1;
                aviso = 1;
            }
            if (bandizquierda == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = -0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = 0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
```

```
            }
            if (bandderecha == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = 0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = -0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (bandizquierda == 1)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandizquierda == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
```

```
                }
            }
            if (bandderecha == 1)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (direction.z == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
```

```
                }
            }
            if (direction.y == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cam = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
```

```
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cam = 1;
        mov = 0;
    }
    public void comparacion()
    {
        while (cam == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cam = 0;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
```

```
                    cam = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.z - coordy[i]) >
0.1f)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cam = 0;
                    }
                    else
                    {
                        cambios.z = transform.position.z;
                    }
                }
                if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordz[i]); ;
                    cambios.y = aux;
                    cam = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.y - coordz[i]) >
0.5f)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cam = 0;
                    }
                    else
                    {
                        cambios.y = transform.position.y;
                    }
                }
                if (cam == 0)
                {

                    mov = 1;
                    actualizar = 1;
                    tnuevo = Convert.ToSingle(coordt[i]);
                    cam = 0;
                    dist = 0;
                    t = Convert.ToSingle(coordt[i - j]);
                    j = 1;
                }
                if (cam == 1)
                {
                    j++;
                }
            }
            i++;
        }
```

```
        }
    }
}
```

Script Robot 2

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot2 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cam = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    private float bandderecha;
    private float bandizquierda;
    private int contador;
    private int aviso;

    void Start()
    {
        bandderecha = 0;
        bandizquierda = 0;
        contador = 0;
        aviso = 0;
    }
```

```
public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.05f)
            {
                direction.x = -0.2f;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.05f)
            {
                direction.x = 0.2f;
            }
            if (Math.Abs(cambios.y - transform.position.y) > 0.5f)
            {
                if (cambios.y > transform.position.y)
                {
                    direction.z = 0.2f;
                }
                else
                {
                    direction.z = 0f;
                }
                if (cambios.y < transform.position.y)
                {
                    direction.z = -0.2f;
                }
            }
            if (Math.Round(cambios.z - transform.position.z, 1) > 0.05f)
            {
                direction.y = 0.2f;
            }
            else
            {
                direction.y = 0f;
            }
            if (Math.Round(cambios.z - transform.position.z, 1) < 0.05f)
            {
                direction.y = -0.2f;
            }
            actualizar = 0;
            if (bandizquierda == 1)
```

```
                {
                    if (direction.x == 0 || direction.x == -0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandizquierda = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                if (bandderecha == 1)
                {
                    if (direction.x == 0 || direction.x == 0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
```

```
                            bandderecha = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                dist = Math.Abs(cambios.z + cambios.y + cambios.x -
transform.position.x - transform.position.y - transform.position.z);
                speed = dist / (tnuevo - t);
                if (speed > 10 && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 0.1f;
                }
                if (speed < 0.1f && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 10;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && direction.z == 0
&& bandderecha == 0)
            {
                if (direction.y == 0.2f)
```

```
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0.0f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 90.0f,
0.0f);
                }
                bandizquierda = 1;
                aviso = 1;
            }
            if (direction.x == -0.2f && bandderecha == 0 && direction.z == 0
&& bandizquierda == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -90.0f,
0.0f);
                }
                bandderecha = 1;
                aviso = 1;
            }
            if (bandizquierda == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = -0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = 0.2f;
```

```
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            if (bandderecha == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = 0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = -0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (bandizquierda == 1)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandizquierda == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
```

```
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (direction.z == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
```

```
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cam = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
```

```
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cam = 1;
        mov = 0;
    }
    public void comparacion()
    {
        while (cam == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cam = 0;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
```

```
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.z - coordy[i]) >
0.1f)
                        {
                            aux = Convert.ToSingle(coordy[i]); ;
                            cambios.z = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.z = transform.position.z;
                        }
                    }
                    if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.y - coordz[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordz[i]); ;
                            cambios.y = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cam == 0)
                    {

                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        cam = 0;
                        dist = 0;
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    if (cam == 1)
                    {
                        j++;
```

```
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 3

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot3 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cam = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    private float bandderecha;
    private float bandizquierda;
    private int contador;
    private int aviso;

    void Start()
    {
```

```
        bandderecha = 0;
        bandizquierda = 0;
        contador = 0;
        aviso = 0;
    }

    public void Update()
    {
        comparacion();
        actualizacion();
    }

    public void actualizacion()
    {
        if (mov == 1)
        {
            if (actualizar == 1)
            {
                if (Math.Round(cambios.x - transform.position.x, 1) > 0.05f)
                {
                    direction.x = -0.2f;
                }
                else
                {
                    direction.x = 0f;
                }
                if (Math.Round(cambios.x - transform.position.x, 1) < 0.05f)
                {
                    direction.x = 0.2f;
                }
                if (Math.Abs(cambios.y - transform.position.y) > 0.5f)
                {
                    if (cambios.y > transform.position.y)
                    {
                        direction.z = 0.2f;
                    }
                    else
                    {
                        direction.z = 0f;
                    }
                    if (cambios.y < transform.position.y)
                    {
                        direction.z = -0.2f;
                    }
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.05f)
                {
                    direction.y = 0.2f;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.05f)
```

```
                {
                    direction.y = -0.2f;
                }
                actualizar = 0;
                if (bandizquierda == 1)
                {
                    if (direction.x == 0 || direction.x == -0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandizquierda = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                if (bandderecha == 1)
                {
                    if (direction.x == 0 || direction.x == 0.2f)
                    {
```

```
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandderecha = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                dist = Math.Abs(cambios.z + cambios.y + cambios.x -
transform.position.x - transform.position.y - transform.position.z);
                speed = dist / (tnuevo - t);
                if (speed > 10 && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 0.1f;
                }
                if (speed < 0.1f && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 10;
                }
```

```
            }
            if (direction.x == 0.2f && bandizquierda == 0 && direction.z == 0
&& bandderecha == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0.0f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 90.0f,
0.0f);
                }
                bandizquierda = 1;
                aviso = 1;
            }
            if (direction.x == -0.2f && bandderecha == 0 && direction.z == 0
&& bandizquierda == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -90.0f,
0.0f);
                }
                bandderecha = 1;
                aviso = 1;
            }
            if (bandizquierda == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = -0.2f;
                }
                else
                {
```

```
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = 0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            if (bandderecha == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = 0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = -0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (bandizquierda == 1)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandizquierda == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
```

```
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (direction.z == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
```

```
            }
            if (direction.z == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cam = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
```

```
                t = tnuevo;
            }
        }
    }
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cam = 1;
        mov = 0;
    }
    public void comparacion()
    {
        while (cam == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cam = 0;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cam = 0;
                        }
                        else
```

```
                    {
                        cambios.x = transform.position.x;
                    }
                }
                if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordy[i]); ;
                    cambios.z = aux;
                    cam = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.z - coordy[i]) >
0.1f)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cam = 0;
                    }
                    else
                    {
                        cambios.z = transform.position.z;
                    }
                }
                if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordz[i]); ;
                    cambios.y = aux;
                    cam = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.y - coordz[i]) >
0.5f)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cam = 0;
                    }
                    else
                    {
                        cambios.y = transform.position.y;
                    }
                }
                if (cam == 0)
                {

                    mov = 1;
                    actualizar = 1;
                    tnuevo = Convert.ToSingle(coordt[i]);
                    cam = 0;
                    dist = 0;
                    t = Convert.ToSingle(coordt[i - j]);
                    j = 1;
```

```
                    }
                    if (cam == 1)
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 11

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot11 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cam = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    private float bandderecha;
    private float bandizquierda;
    private int contador;
    private int aviso;
```

```
void Start()
{
    bandderecha = 0;
    bandizquierda = 0;
    contador = 0;
    aviso = 0;
}

public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.05f)
            {
                direction.x = -0.2f;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.05f)
            {
                direction.x = 0.2f;
            }
            if (Math.Abs(cambios.y - transform.position.y) > 0.5f)
            {
                if (cambios.y > transform.position.y)
                {
                    direction.z = 0.2f;
                }
                else
                {
                    direction.z = 0f;
                }
                if (cambios.y < transform.position.y)
                {
                    direction.z = -0.2f;
                }
            }
            if (Math.Round(cambios.z - transform.position.z, 1) > 0.05f)
            {
                direction.y = 0.2f;
            }
            else
            {
```

```
                direction.y = 0f;
            }
            if (Math.Round(cambios.z - transform.position.z, 1) < 0.05f)
            {
                direction.y = -0.2f;
            }
            actualizar = 0;
            if (bandizquierda == 1)
            {
                if (direction.x == 0 || direction.x == -0.2f)
                {
                    contador++;
                    if (contador > 1)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                        bandizquierda = 0;
                        contador = 0;
                    }
                    else
                    {
                        aviso = 1;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                else
                {
                    aviso = 1;
                    contador = 0;
                    if (direction.y == 0.2f)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                        speed = speed * 0.1f;
                    }
                    if (direction.y == -0.2f)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                        speed = speed * 0.1f;
                    }
                }
            }
            if (bandderecha == 1)
```

```
                {
                    if (direction.x == 0 || direction.x == 0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandderecha = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                dist = Math.Abs(cambios.z + cambios.y + cambios.x -
transform.position.x - transform.position.y - transform.position.z);
                speed = dist / (tnuevo - t);
                if (speed > 10 && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 0.1f;
                }
                if (speed < 0.1f && bandizquierda == 0 && bandderecha == 0)
```

```
                {
                    speed = speed * 10;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && direction.z == 0
&& bandderecha == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0.0f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 90.0f,
0.0f);
                }
                bandizquierda = 1;
                aviso = 1;
            }
            if (direction.x == -0.2f && bandderecha == 0 && direction.z == 0
&& bandizquierda == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -90.0f,
0.0f);
                }
                bandderecha = 1;
                aviso = 1;
            }
            if (bandizquierda == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = -0.2f;
```

```
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = 0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            if (bandderecha == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = 0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = -0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (bandizquierda == 1)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandizquierda == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
```

```
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (direction.z == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
```

```
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cam = 1;
                cambios.y = 0;
```

```
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cam = 1;
        mov = 0;
    }
    public void comparacion()
    {
        while (cam == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cam = 0;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
```

```
                        cam = 0;
                    }
                    else
                    {
                        cambios.x = transform.position.x;
                    }
                }
                if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordy[i]); ;
                    cambios.z = aux;
                    cam = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.z - coordy[i]) >
0.1f)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cam = 0;
                    }
                    else
                    {
                        cambios.z = transform.position.z;
                    }
                }
                if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordz[i]); ;
                    cambios.y = aux;
                    cam = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.y - coordz[i]) >
0.5f)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cam = 0;
                    }
                    else
                    {
                        cambios.y = transform.position.y;
                    }
                }
                if (cam == 0)
                {

                    mov = 1;
                    actualizar = 1;
                    tnuevo = Convert.ToSingle(coordt[i]);
                    cam = 0;
```

```
                        dist = 0;
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    if (cam == 1)
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 12

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot12 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cam = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
    private int j = 1;

    private float bandderecha;
```

```
private float bandizquierda;
private int contador;
private int aviso;

void Start()
{
    bandderecha = 0;
    bandizquierda = 0;
    contador = 0;
    aviso = 0;
}

public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.05f)
            {
                direction.x = -0.2f;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.05f)
            {
                direction.x = 0.2f;
            }
            if (Math.Abs(cambios.y - transform.position.y) > 0.5f)
            {
                if (cambios.y > transform.position.y)
                {
                    direction.z = 0.2f;
                }
                else
                {
                    direction.z = 0f;
                }
                if (cambios.y < transform.position.y)
                {
                    direction.z = -0.2f;
                }
            }
            if (Math.Round(cambios.z - transform.position.z, 1) > 0.05f)
            {
                direction.y = 0.2f;
```

```
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.05f)
                {
                    direction.y = -0.2f;
                }
                actualizar = 0;
                if (bandizquierda == 1)
                {
                    if (direction.x == 0 || direction.x == -0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandizquierda = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
```

```
                    }
                }
                if (bandderecha == 1)
                {
                    if (direction.x == 0 || direction.x == 0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandderecha = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                dist = Math.Abs(cambios.z + cambios.y + cambios.x -
transform.position.x - transform.position.y - transform.position.z);
                speed = dist / (tnuevo - t);
                if (speed > 10 && bandizquierda == 0 && bandderecha == 0)
                {
```

```
                    speed = speed * 0.1f;
                }
                if (speed < 0.1f && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 10;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && direction.z == 0
&& bandderecha == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0.0f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 90.0f,
0.0f);
                }
                bandizquierda = 1;
                aviso = 1;
            }
            if (direction.x == -0.2f && bandderecha == 0 && direction.z == 0
&& bandizquierda == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -90.0f,
0.0f);
                }
                bandderecha = 1;
                aviso = 1;
            }
            if (bandizquierda == 1 && aviso == 1)
            {
```

```
                if (direction.y == 0.2f)
                {
                    direction.x = -0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = 0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            if (bandderecha == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = 0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = -0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (bandizquierda == 1)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandizquierda == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
```

```
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (direction.z == 0.2f && bandizquierda == 0 && bandderecha ==
0)
```

```
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
```

```
                mov = 0;
                cam = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cam = 1;
        mov = 0;
    }
    public void comparacion()
    {
        while (cam == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cam = 0;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.5f)
```

```
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
                    }
                    else
                    {
                        cambios.x = transform.position.x;
                    }
                }
                if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordy[i]); ;
                    cambios.z = aux;
                    cam = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.z - coordy[i]) >
0.1f)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cam = 0;
                    }
                    else
                    {
                        cambios.z = transform.position.z;
                    }
                }
                if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                {
                    aux = Convert.ToSingle(coordz[i]); ;
                    cambios.y = aux;
                    cam = 0;
                }
                else
                {
                    if (Math.Abs(transform.position.y - coordz[i]) >
0.5f)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cam = 0;
                    }
                    else
                    {
                        cambios.y = transform.position.y;
                    }
                }
                if (cam == 0)
                {

                    mov = 1;
```

```
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        cam = 0;
                        dist = 0;
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    if (cam == 1)
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 13

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot13 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cam = 0;
    private int mov = 0;
    private int length = 0;
    private float aux;
    private int actualizar = 0;
    private float tnuevo = 0;
    private float dist = 0;
```

```
private int j = 1;

private float bandderecha;
private float bandizquierda;
private int contador;
private int aviso;

void Start()
{
    bandderecha = 0;
    bandizquierda = 0;
    contador = 0;
    aviso = 0;
}

public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.05f)
            {
                direction.x = -0.2f;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.05f)
            {
                direction.x = 0.2f;
            }
            if (Math.Abs(cambios.y - transform.position.y) > 0.5f)
            {
                if (cambios.y > transform.position.y)
                {
                    direction.z = 0.2f;
                }
                else
                {
                    direction.z = 0f;
                }
                if (cambios.y < transform.position.y)
                {
                    direction.z = -0.2f;
                }
            }
```

```
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.05f)
                {
                    direction.y = 0.2f;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.05f)
                {
                    direction.y = -0.2f;
                }
                actualizar = 0;
                if (bandizquierda == 1)
                {
                    if (direction.x == 0 || direction.x == -0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandizquierda = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
```

```
                            transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                if (bandderecha == 1)
                {
                    if (direction.x == 0 || direction.x == 0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandderecha = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
```

```
                dist = Math.Abs(cambios.z + cambios.y + cambios.x -
transform.position.x - transform.position.y - transform.position.z);
                speed = dist / (tnuevo - t);
                if (speed > 10 && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 0.1f;
                }
                if (speed < 0.1f && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 10;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && direction.z == 0
&& bandderecha == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0.0f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 90.0f,
0.0f);
                }
                bandizquierda = 1;
                aviso = 1;
            }
            if (direction.x == -0.2f && bandderecha == 0 && direction.z == 0
&& bandizquierda == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -90.0f,
0.0f);
                }
```

```
                bandderecha = 1;
                aviso = 1;
            }
            if (bandizquierda == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = -0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = 0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            if (bandderecha == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = 0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = -0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (bandizquierda == 1)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandizquierda == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
```

```
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
```

```
                    }
                }
            }
            if (direction.z == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
```

```
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cam = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cam = 1;
        mov = 0;
    }
    public void comparacion()
    {
        while (cam == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cam = 0;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
```

```
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.z - coordy[i]) >
0.1f)
                        {
                            aux = Convert.ToSingle(coordy[i]); ;
                            cambios.z = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.z = transform.position.z;
                        }
                    }
                    if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.y - coordz[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordz[i]); ;
                            cambios.y = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.y = transform.position.y;
                        }
```

```
                    }
                    if (cam == 0)
                    {

                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        cam = 0;
                        dist = 0;
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    if (cam == 1)
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 14

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot14 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);

    private int i = 2;
    private int cam = 0;
    private int mov = 0;
```

```
private int length = 0;
private float aux;
private int actualizar = 0;
private float tnuevo = 0;
private float dist = 0;
private int j = 1;

private float bandderecha;
private float bandizquierda;
private int contador;
private int aviso;

void Start()
{
    bandderecha = 0;
    bandizquierda = 0;
    contador = 0;
    aviso = 0;
}

public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.05f)
            {
                direction.x = -0.2f;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.05f)
            {
                direction.x = 0.2f;
            }
            if (Math.Abs(cambios.y - transform.position.y) > 0.5f)
            {
                if (cambios.y > transform.position.y)
                {
                    direction.z = 0.2f;
                }
                else
                {
                    direction.z = 0f;
                }
```

```
                    if (cambios.y < transform.position.y)
                    {
                        direction.z = -0.2f;
                    }
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.05f)
                {
                    direction.y = 0.2f;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.05f)
                {
                    direction.y = -0.2f;
                }
                actualizar = 0;
                if (bandizquierda == 1)
                {
                    if (direction.x == 0 || direction.x == -0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandizquierda = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
```

```
                        speed = speed * 0.1f;
                    }
                    if (direction.y == -0.2f)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                        speed = speed * 0.1f;
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == 0 || direction.x == 0.2f)
                {
                    contador++;
                    if (contador > 1)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                        bandderecha = 0;
                        contador = 0;
                    }
                    else
                    {
                        aviso = 1;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
-100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
-80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                else
                {
                    aviso = 1;
                    contador = 0;
                    if (direction.y == 0.2f)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f, -
100.0f, 0.0f);
                        speed = speed * 0.1f;
                    }
                    if (direction.y == -0.2f)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f, -
80.0f, 0.0f);
                        speed = speed * 0.1f;
```

```
                        }
                    }
                }
                dist = Math.Abs(cambios.z + cambios.y + cambios.x -
transform.position.x - transform.position.y - transform.position.z);
                speed = dist / (tnuevo - t);
                if (speed > 10 && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 0.1f;
                }
                if (speed < 0.1f && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 10;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && direction.z == 0
&& bandderecha == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0.0f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 90.0f,
0.0f);
                }
                bandizquierda = 1;
                aviso = 1;
            }
            if (direction.x == -0.2f && bandderecha == 0 && direction.z == 0
&& bandizquierda == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0)
                {
```

```
                    transform.rotation = Quaternion.Euler(-90.0f, -90.0f,
0.0f);
                }
                bandderecha = 1;
                aviso = 1;
            }
            if (bandizquierda == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = -0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = 0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            if (bandderecha == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = 0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = -0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (bandizquierda == 1)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandizquierda == 1)
            {
                if (direction.x == -0.2f)
                {
```

```
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
```

```
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (direction.z == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f && bandizquierda == 0 && bandderecha ==
0)
```

```
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cam = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cam = 1;
        mov = 0;
    }
    public void comparacion()
    {
        while (cam == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cam = 0;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
```

```
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.z - coordy[i]) >
0.1f)
                        {
                            aux = Convert.ToSingle(coordy[i]); ;
                            cambios.z = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.z = transform.position.z;
                        }
                    }
                    if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.y - coordz[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordz[i]); ;
                            cambios.y = aux;
                            cam = 0;
                        }
```

```
                        else
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cam == 0)
                    {

                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        cam = 0;
                        dist = 0;
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    if (cam == 1)
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 15

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot15 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;

    Vector3 direction = new Vector3(0, 0, 0);
    Vector3 cambios = new Vector3(0, 0, 0);
```

```
private int i = 2;
private int cam = 0;
private int mov = 0;
private int length = 0;
private float aux;
private int actualizar = 0;
private float tnuevo = 0;
private float dist = 0;
private int j = 1;

private float bandderecha;
private float bandizquierda;
private int contador;
private int aviso;

void Start()
{
    bandderecha = 0;
    bandizquierda = 0;
    contador = 0;
    aviso = 0;
}

public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.05f)
            {
                direction.x = -0.2f;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.05f)
            {
                direction.x = 0.2f;
            }
            if (Math.Abs(cambios.y - transform.position.y) > 0.5f)
            {
                if (cambios.y > transform.position.y)
                {
                    direction.z = 0.2f;
                }
```

```
                    else
                    {
                        direction.z = 0f;
                    }
                    if (cambios.y < transform.position.y)
                    {
                        direction.z = -0.2f;
                    }
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.05f)
                {
                    direction.y = 0.2f;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.05f)
                {
                    direction.y = -0.2f;
                }
                actualizar = 0;
                if (bandizquierda == 1)
                {
                    if (direction.x == 0 || direction.x == -0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandizquierda = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
```

```
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                if (bandderecha == 1)
                {
                    if (direction.x == 0 || direction.x == 0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandderecha = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
```

```
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                dist = Math.Abs(cambios.z + cambios.y + cambios.x -
transform.position.x - transform.position.y - transform.position.z);
                speed = dist / (tnuevo - t);
                if (speed > 10 && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 0.1f;
                }
                if (speed < 0.1f && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 10;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && direction.z == 0
&& bandderecha == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0.0f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 90.0f,
0.0f);
                }
                bandizquierda = 1;
                aviso = 1;
            }
            if (direction.x == -0.2f && bandderecha == 0 && direction.z == 0
&& bandizquierda == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -80.0f,
0.0f);
```

```
                    speed = speed * 0.1f;
                }
                if (direction.y == 0)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -90.0f,
0.0f);
                }
                bandderecha = 1;
                aviso = 1;
            }
            if (bandizquierda == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = -0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = 0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            if (bandderecha == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = 0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = -0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (bandizquierda == 1)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandizquierda == 1)
```

```
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
```

```
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (direction.z == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
```

```
            }
            if (direction.x == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cam = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cam = 1;
        mov = 0;
    }
    public void comparacion()
    {
        while (cam == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cam = 0;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
```

```
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.z - coordy[i]) >
0.1f)
                        {
                            aux = Convert.ToSingle(coordy[i]); ;
                            cambios.z = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.z = transform.position.z;
                        }
                    }
                    if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.y - coordz[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordz[i]); ;
```

```
                            cambios.y = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cam == 0)
                    {

                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        cam = 0;
                        dist = 0;
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    if (cam == 1)
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 16

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot16 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
    private double[] coordb;
    private double[] coordt;

    private float speed;
```

```
Vector3 direction = new Vector3(0, 0, 0);
Vector3 cambios = new Vector3(0, 0, 0);

private int i = 2;
private int cam = 0;
private int mov = 0;
private int length = 0;
private float aux;
private int actualizar = 0;
private float tnuevo = 0;
private float dist = 0;
private int j = 1;

private float bandderecha;
private float bandizquierda;
private int contador;
private int aviso;

void Start()
{
    bandderecha = 0;
    bandizquierda = 0;
    contador = 0;
    aviso = 0;
}

public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.05f)
            {
                direction.x = -0.2f;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.05f)
            {
                direction.x = 0.2f;
            }
            if (Math.Abs(cambios.y - transform.position.y) > 0.5f)
            {
                if (cambios.y > transform.position.y)
```

```
                    {
                        direction.z = 0.2f;
                    }
                    else
                    {
                        direction.z = 0f;
                    }
                    if (cambios.y < transform.position.y)
                    {
                        direction.z = -0.2f;
                    }
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.05f)
                {
                    direction.y = 0.2f;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.05f)
                {
                    direction.y = -0.2f;
                }
                actualizar = 0;
                if (bandizquierda == 1)
                {
                    if (direction.x == 0 || direction.x == -0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandizquierda = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
```

```
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                if (bandderecha == 1)
                {
                    if (direction.x == 0 || direction.x == 0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandderecha = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
-80.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                        }
                    }
                    else
                    {
                        aviso = 1;
                        contador = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
100.0f, 0.0f);
```

```
                        speed = speed * 0.1f;
                    }
                    if (direction.y == -0.2f)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f, -
80.0f, 0.0f);
                        speed = speed * 0.1f;
                    }
                }
            }
            dist = Math.Abs(cambios.z + cambios.y + cambios.x -
transform.position.x - transform.position.y - transform.position.z);
            speed = dist / (tnuevo - t);
            if (speed > 10 && bandizquierda == 0 && bandderecha == 0)
            {
                speed = speed * 0.1f;
            }
            if (speed < 0.1f && bandizquierda == 0 && bandderecha == 0)
            {
                speed = speed * 10;
            }
        }
        if (direction.x == 0.2f && bandizquierda == 0 && direction.z == 0
&& bandderecha == 0)
        {
            if (direction.y == 0.2f)
            {
                transform.rotation = Quaternion.Euler(-90.0f, 100.0f,
0.0f);
                speed = speed * 0.1f;
            }
            if (direction.y == -0.2f)
            {
                transform.rotation = Quaternion.Euler(-90.0f, 80.0f,
0.0f);
                speed = speed * 0.1f;
            }
            if (direction.y == 0.0f)
            {
                transform.rotation = Quaternion.Euler(-90.0f, 90.0f,
0.0f);
            }
            bandizquierda = 1;
            aviso = 1;
        }
        if (direction.x == -0.2f && bandderecha == 0 && direction.z == 0
&& bandizquierda == 0)
        {
            if (direction.y == 0.2f)
            {
                transform.rotation = Quaternion.Euler(-90.0f, -100.0f,
0.0f);
                speed = speed * 0.1f;
            }
            if (direction.y == -0.2f)
```

```
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -90.0f,
0.0f);
                }
                bandderecha = 1;
                aviso = 1;
            }
            if (bandizquierda == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = -0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = 0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            if (bandderecha == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = 0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = -0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (bandizquierda == 1)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
```

```
                }
            }
            if (bandizquierda == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
```

```
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (direction.z == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
```

```
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cam = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cam = 1;
        mov = 0;
    }
    public void comparacion()
    {
        while (cam == 1)
        {
            if (i < length)
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cam = 0;
                    direction.x = 0;
```

```
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.z - coordy[i]) >
0.1f)
                        {
                            aux = Convert.ToSingle(coordy[i]); ;
                            cambios.z = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.z = transform.position.z;
                        }
                    }
                    if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
                        cambios.y = aux;
                        cam = 0;
                    }
                    else
                    {
```

```
                        if (Math.Abs(transform.position.y - coordz[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordz[i]); ;
                            cambios.y = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cam == 0)
                    {

                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        cam = 0;
                        dist = 0;
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    if (cam == 1)
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```

Script Robot 17

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using UnityEngine;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
using System.Text;

public class Robot17 : MonoBehaviour
{
    private float x;
    private float y;
    private float z;
    private float t;

    private double[] coordx;
    private double[] coordy;
    private double[] coordz;
```

```
private double[] coordb;
private double[] coordt;

private float speed;

Vector3 direction = new Vector3(0, 0, 0);
Vector3 cambios = new Vector3(0, 0, 0);

private int i = 2;
private int cam = 0;
private int mov = 0;
private int length = 0;
private float aux;
private int actualizar = 0;
private float tnuevo = 0;
private float dist = 0;
private int j = 1;

private float bandderecha;
private float bandizquierda;
private int contador;
private int aviso;

void Start()
{
    bandderecha = 0;
    bandizquierda = 0;
    contador = 0;
    aviso = 0;
}

public void Update()
{
    comparacion();
    actualizacion();
}

public void actualizacion()
{
    if (mov == 1)
    {
        if (actualizar == 1)
        {
            if (Math.Round(cambios.x - transform.position.x, 1) > 0.05f)
            {
                direction.x = -0.2f;
            }
            else
            {
                direction.x = 0f;
            }
            if (Math.Round(cambios.x - transform.position.x, 1) < 0.05f)
            {
                direction.x = 0.2f;
```

```
                }
                if (Math.Abs(cambios.y - transform.position.y) > 0.5f)
                {
                    if (cambios.y > transform.position.y)
                    {
                        direction.z = 0.2f;
                    }
                    else
                    {
                        direction.z = 0f;
                    }
                    if (cambios.y < transform.position.y)
                    {
                        direction.z = -0.2f;
                    }
                }
                if (Math.Round(cambios.z - transform.position.z, 1) > 0.05f)
                {
                    direction.y = 0.2f;
                }
                else
                {
                    direction.y = 0f;
                }
                if (Math.Round(cambios.z - transform.position.z, 1) < 0.05f)
                {
                    direction.y = -0.2f;
                }
                actualizar = 0;
                if (bandizquierda == 1)
                {
                    if (direction.x == 0 || direction.x == -0.2f)
                    {
                        contador++;
                        if (contador > 1)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                            bandizquierda = 0;
                            contador = 0;
                        }
                        else
                        {
                            aviso = 1;
                            if (direction.y == 0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                                speed = speed * 0.1f;
                            }
                            if (direction.y == -0.2f)
                            {
                                transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                                speed = speed * 0.1f;
```

```
                        }
                    }
                }
                else
                {
                    aviso = 1;
                    contador = 0;
                    if (direction.y == 0.2f)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f,
100.0f, 0.0f);
                        speed = speed * 0.1f;
                    }
                    if (direction.y == -0.2f)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f,
80.0f, 0.0f);
                        speed = speed * 0.1f;
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == 0 || direction.x == 0.2f)
                {
                    contador++;
                    if (contador > 1)
                    {
                        transform.rotation = Quaternion.Euler(-90.0f,
180.0f, 0.0f);
                        bandderecha = 0;
                        contador = 0;
                    }
                    else
                    {
                        aviso = 1;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
-100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
-80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                else
                {
                    aviso = 1;
                    contador = 0;
```

```
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
100.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                        if (direction.y == -0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
80.0f, 0.0f);
                            speed = speed * 0.1f;
                        }
                    }
                }
                dist = Math.Abs(cambios.z + cambios.y + cambios.x -
transform.position.x - transform.position.y - transform.position.z);
                speed = dist / (tnuevo - t);
                if (speed > 10 && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 0.1f;
                }
                if (speed < 0.1f && bandizquierda == 0 && bandderecha == 0)
                {
                    speed = speed * 10;
                }
            }
            if (direction.x == 0.2f && bandizquierda == 0 && direction.z == 0
&& bandderecha == 0)
            {
                if (direction.y == 0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0.0f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, 90.0f,
0.0f);
                }
                bandizquierda = 1;
                aviso = 1;
            }
            if (direction.x == -0.2f && bandderecha == 0 && direction.z == 0
&& bandizquierda == 0)
            {
                if (direction.y == 0.2f)
                {
```

```
                    transform.rotation = Quaternion.Euler(-90.0f, -100.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == -0.2f)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -80.0f,
0.0f);
                    speed = speed * 0.1f;
                }
                if (direction.y == 0)
                {
                    transform.rotation = Quaternion.Euler(-90.0f, -90.0f,
0.0f);
                }
                bandderecha = 1;
                aviso = 1;
            }
            if (bandizquierda == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = -0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = 0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            if (bandderecha == 1 && aviso == 1)
            {
                if (direction.y == 0.2f)
                {
                    direction.x = 0.2f;
                }
                else
                {
                    direction.x = 0;
                }
                if (direction.y == -0.2f)
                {
                    direction.x = -0.2f;
                }
                direction.y = 0.2f;
                aviso = 0;
            }
            transform.Translate(direction * speed * Time.deltaTime);
            if (bandizquierda == 1)
```

```
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandizquierda == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f,
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (bandderecha == 1)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.y = 0;
                }
            }
            if (bandderecha == 1)
            {
                if (direction.x == -0.2f)
                {
                    if (transform.position.z < cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
```

```
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
                if (direction.x == 0.2f)
                {
                    if (transform.position.z > cambios.z ||
transform.position.z == cambios.z)
                    {
                        direction.x = 0;
                        if (direction.y == 0.2f)
                        {
                            transform.rotation = Quaternion.Euler(-90.0f, -
90.0f, 0.0f);
                        }
                    }
                }
            }
            if (direction.z == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y > cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.z == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.y < cambios.y || transform.position.y
== cambios.y)
                {
                    direction.z = 0;
                }
            }
            if (direction.y == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z > cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
            }
            if (direction.y == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.z < cambios.z || transform.position.z
== cambios.z)
                {
                    direction.y = 0;
                }
```

```
            }
            if (direction.x == 0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x < cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == -0.2f && bandizquierda == 0 && bandderecha ==
0)
            {
                if (transform.position.x > cambios.x || transform.position.x
== cambios.x)
                {
                    direction.x = 0;
                }
            }
            if (direction.x == 0 && direction.y == 0 && direction.z == 0)
            {
                mov = 0;
                cam = 1;
                cambios.y = 0;
                cambios.x = 0;
                cambios.z = 0;
                dist = 0;
                t = tnuevo;
            }
        }
    }
    public void movimiento(double[] xx, double[] yy, double[] zz, double[]
bb, double[] tt)
    {
        coordx = xx;
        coordy = yy;
        coordz = zz;
        coordb = bb;
        coordt = tt;

        x = Convert.ToSingle(coordx[1]);
        y = Convert.ToSingle(coordz[1]);
        z = Convert.ToSingle(coordy[1]);
        t = Convert.ToSingle(coordt[1]);

        transform.position = new Vector3(x, y, z);
        length = coordx.Length;
        cam = 1;
        mov = 0;
    }
    public void comparacion()
    {
        while (cam == 1)
        {
            if (i < length)
```

```
            {
                if (Convert.ToSingle(coordx[i]) == 0 &&
Convert.ToSingle(coordy[i]) == 0 && Convert.ToSingle(coordz[i]) == 0)
                {
                    cam = 0;
                    direction.x = 0;
                    direction.y = 0;
                    direction.z = 0;
                }
                else
                {
                    if (Math.Abs(coordx[i] - coordx[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordx[i]); ;
                        cambios.x = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.x - coordx[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordx[i]); ;
                            cambios.x = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.x = transform.position.x;
                        }
                    }
                    if (Math.Abs(coordy[i] - coordy[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordy[i]); ;
                        cambios.z = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.z - coordy[i]) >
0.1f)
                        {
                            aux = Convert.ToSingle(coordy[i]); ;
                            cambios.z = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.z = transform.position.z;
                        }
                    }
                    if (Math.Abs(coordz[i] - coordz[i - 1]) > 0)
                    {
                        aux = Convert.ToSingle(coordz[i]); ;
```

```
                        cambios.y = aux;
                        cam = 0;
                    }
                    else
                    {
                        if (Math.Abs(transform.position.y - coordz[i]) >
0.5f)
                        {
                            aux = Convert.ToSingle(coordz[i]); ;
                            cambios.y = aux;
                            cam = 0;
                        }
                        else
                        {
                            cambios.y = transform.position.y;
                        }
                    }
                    if (cam == 0)
                    {

                        mov = 1;
                        actualizar = 1;
                        tnuevo = Convert.ToSingle(coordt[i]);
                        cam = 0;
                        dist = 0;
                        t = Convert.ToSingle(coordt[i - j]);
                        j = 1;
                    }
                    if (cam == 1)
                    {
                        j++;
                    }
                }
                i++;
            }
        }
    }
}
```