Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Interfaz gráfica de los brazos robóticos Scorbot ER IX y V+

Graphic interface for robotic arms Scorbot ER XI and V+

Óscar Josué Catari Gutiérrez

Curso: 2018/19

Convocatoria: septiembre de 2019

D. **Santiago Torres Álvarez**, con N.I.F. 43.369.478-B profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Rafael Arnay del Arco**, con N.I.F. 78.569.591-G profesor Ayudante Doctor de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"*Interfaz gráfica de los brazos robóticos Scorbot ER IX y V+*"

ha sido realizada bajo su dirección por D. **Óscar Josué Catari Gutiérrez**,

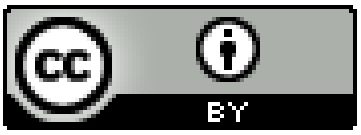
con N.I.F. 79.357.062-R.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 6 de septiembre de 2019

Agradecimientos

Quiero darle las gracias a Santiago Torres Álvarez, quien me ha ayudado a entender el funcionamiento de los Scorbots y sus comandos, y también me gustaría agradecer a Rafael Arnay del Arco sus consejos sobre la interfaz gráfica, los cuales me han guiado para mejorarla de forma progresiva.

Licencia



Resumen

El objetivo de este trabajo ha sido el desarrollo de una interfaz gráfica que permita la manipulación y simulación de los Scorbots ER IX y V Plus.

Dado que para la manipulación de los Scorbots se requiere resolver el problema de la cinemática inversa, de entre las diversas opciones, se ha implementado el algoritmo CCD (Cyclic Coordinate Descent). La interfaz gráfica que emplea el algoritmo anterior, permite la manipulación y simulación en un espacio 3D de los Scorbots ER IX y V Plus. A esto se le añade la posibilidad de guardar y configurar posiciones a las que los Scorbots intentan llegar. Gran parte de las funcionalidades se realizan a través de comandos. En este documento hay un tutorial de la interfaz gráfica para su utilización. También se detalla el entorno en el que se ha desarrollado el programa, al igual que todas sus características y funcionalidades.

Palabras clave: Simulador, Scorbot, Scorbot ER IX, Scorbot ER V Plus, cinemática inversa, CCD, Spline de Catmull-Rom, interfaz gráfica.

Abstract

The aim of this work has been the development of a graphic interface that allows the manipulation and simulation of the robotic structures Scorbot ER IX and Scorbot V Plus.

Since the problem of inverse kinematics is required to be solved for the Scorbot manipulation, among various options, the CCD (Cyclic Coordinate Descent) algorithm has been implemented. The graphic interface which uses the previous algorithm allows the manipulation and simulation in a 3D space of the Scorbots ER IX and V Plus. The possibility of saving and configuring positions that the Scorbots try to reach is added to the mentioned interface. Many of the functionalities are executed through commands. In this document a tutorial of the graphic interface is presented for its usage. The document also details the environment in which the program has been developed, as well as all its features and functionalities.

Keywords: Simulator, Scorbot, ScorbotERIX, ScorbotERVPlus, inverse kinematics, CCD, Catmull-Rom Spline, graphic interface.

Índice general

Índice de figuras

básicas, simulación y orientación.

Índice de tablas

1. Introducción

La robótica tiene muchas aplicaciones hoy en día, una de ellas es la **automatización de procesos** como, por ejemplo, el montaje de vehículos en fábricas mediante brazos robóticos, los cuales pueden trabajar de forma coordinada. Estos brazos robóticos son capaces de ejecutar una amplia variedad de tareas en muchos sectores, sobre todo si se requieren en fábricas. Sin embargo, esto implica la **configuración y programación de los brazos robóticos** en cada tarea específica.

En este proyecto de trabajo de fin de grado se pretende realizar una **interfaz gráfica para los brazos robóticos Scorbot ER IX y V Plus** con el objetivo de facilitar la manipulación de los mismos. Ambos Scorbots ya disponen de un software con una interfaz gráfica llamada ScorBase [10] y también una consola de comandos que acepta instrucciones ACL [3, 4]. Sin embargo, dicha interfaz gráfica carece de un entorno de simulación y usar la consola de comandos requiere el aprendizaje del lenguaje ACL [3, 4], por ello el objetivo es conseguir realizar un **entorno gráfico mejorado y actualizado** con las tecnologías actuales.

1.1. Scorbot ER IX y V Plus

Los Scorbots ER IX y V Plus son **manipuladores que se pueden controlar mediante comandos**. Ambos poseen un controlador, controlador A (Scorbot ER V Plus) y controlador B (Scorbot ER IX), que ejecuta los comandos deseados y también almacena datos como las posiciones creadas. Cada controlador debe estar conectado a un ordenador desde el que se envía un comando mediante un software específico, que en este caso será el programa desarrollado en este trabajo “Simulador de Scorbot”.

Ambos Scorbots presentan una estructura similar que consta de 6 partes que son Base, Shoulder, Elbow, Pitch, Roll y pinza. Hay que destacar que el efector final en el Scorbot ER IX es el centro de su articulación *“Roll”*, mientras que en el Scorbot ER V Plus es el punto centrado del extremo de su pinza.

Articulación	Scorbot ER IX	Scorbot ER V Plus
Base	270°	310º
Shoulder	145°	+130º / -35º
Elbow	210°	+- 130º
Pitch	196°	+- 130º
Roll	737°	+- 570º

Tabla 1.1: Especificaciones del Scorbot ER IX y V Plus.

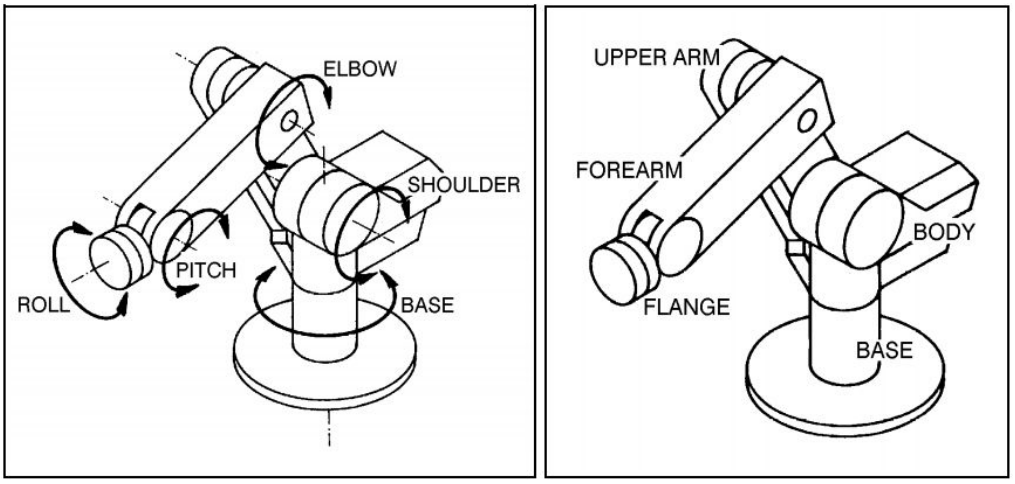


Figura 1.1: Articulaciones del Scorbot ER IX. Base, shoulder, elbow, pitch y roll.

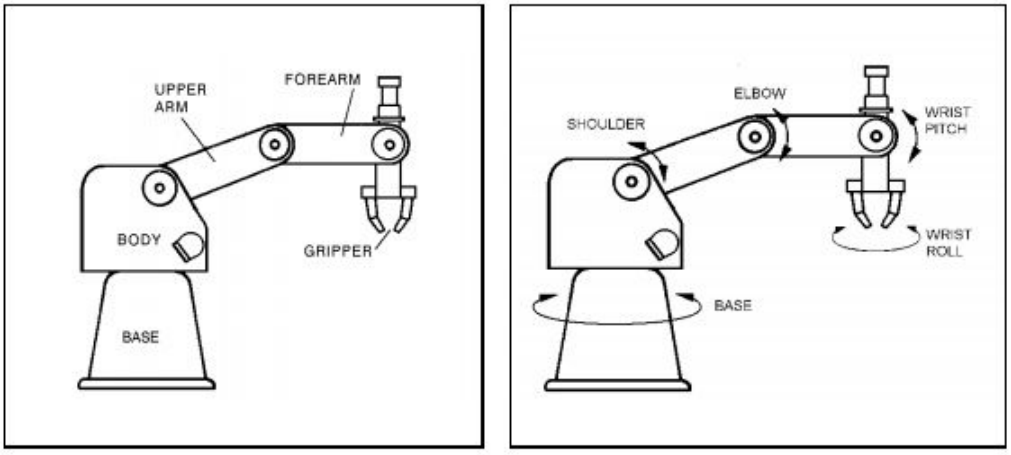


Figura 1.2: Articulaciones del Scorbot V Plus. Base, shoulder, elbow, pitch y roll.

2. Antecedentes y estado actual del tema

A lo largo de los años, se han establecido una serie de procedimientos y herramientas para la resolución de la **cinemática directa** y la **cinemática inversa de los robots manipuladores**. Estas técnicas serán de gran utilidad para la simulación de los brazos robóticos en este trabajo de fin de grado.

Actualmente existen multitud de brazos robóticos que se caracterizan principalmente por su versatilidad para realizar tareas de propósito general y/o específico. Dependiendo de la complejidad de las mismas, y de la tarea en sí, se dota al robot de más o menos grados de libertad para ejercer sus movimientos. Los robots, a su vez, suelen disponer de su propio software específico para poder programarlos con las tareas que vayan a realizar. Algunos de estos programas son **ABB Robot Studio [8] y Kuka Sim [9]**, ambos pertenecientes a fabricantes de diferentes modelos de brazos robóticos. Estos fabricantes ofrecen un entorno gráfico de simulación para facilitar la definición de cada tarea a realizar. Hay que destacar que estos entornos virtuales ofrecen elementos complejos como la posibilidad de interacción con objetos virtuales, la cooperación de múltiple robots en una misma tarea y la interactuación con otros mecanismos como cintas transportadoras, mesas giratorias, zonas de paletización, etc.

Los Scorbot-ER objeto de este trabajo también disponen de su propia plataforma de simulación provista por el fabricante, Intelitek. Se trata de **ScorBase [10]**, cuya interfaz permite el movimiento manual de las articulaciones, la ejecución de programas y la definición de posiciones. Sin embargo, dicho software ha quedado bastante obsoleto, permitiéndose su ejecución solamente en versiones no soportadas actualmente de Windows, y además no cuenta con un simulador gráfico que permite una interactuación sencilla e intuitiva con el manipulador.

3. Objetivos

El objetivo de este trabajo es desarrollar una **interfaz gráfica que permita la manipulación y simulación** del **Scorbot ER IX y V Plus**.

Para llevar a cabo el objetivo descrito, ha sido necesario completar las siguientes tareas:

- ➢ **Modelado** de las brazos robóticos, Scorbot ER IX y V Plus, usando Blender.
- ➢ Interfaz gráfica usando Unity.
    - **Diseño gráfico** intuitivo: Estructura visual de los componentes de la interfaz gráfica en la que sea fácil de reconocer las funcionalidades que se ofrecen.
    - **Estructuras de los datos** que representan los Scorbots: Agrupación de los datos para facilitar la programación orientada a objetos.
    - **Controles de cámara**: Permitir el movimiento y rotación de la cámara principal de Unity para poder navegar por la simulación libremente.
    - **Controles de los Scorbots por teclado**: Manipulación de las articulaciones individuales de los Scorbots mediante teclas.
    - **Controles de los Scorbot por ratón**: Manipulación de las articulaciones individuales de los Scorbots mediante ratón.
    - **Visualización de trayectorias y variables**: Implementar el algoritmo Spline de Catmull-Rom y mostrar información sobre los Scorbots y sus posiciones almacenadas.
    - **Cinemática directa**: Resuelto implícitamente por Unity con modelos 3D mientras su jerarquía de componentes sea adecuada.
    - **Cinemática inversa**: Implementar el algoritmo CDD (Cyclic Coordinate Descent).
    - **Consola de comandos**: Permitir el acceso a una consola en la que se puedan ejecutar comandos ACL de los Scorbots.
    - **Ajustes de personalización**: Parametrizar valores de la simulación como la sensibilidad de la cámara, el Scorbot seleccionado, etc.
    - **Información de ayuda**: Mostrar información básica para el uso de la interfaz gráfica.
    - **Testeo y corrección de errores**: Comprobar que las funcionalidades implementadas funcionan de forma esperada en todos los casos posibles.

4. Entorno de desarrollo y algoritmos

En este capítulo se describe el entorno en el que se ha llevado a cabo el desarrollo de los objetivos propuestos en este trabajo y se presentan en mayor profundidad los algoritmos utilizados.

4.1. Entorno de desarrollo

En esta sección se detalla los programas que han ayudado en el desarrollo de la interfaz gráfica, así como algunas consideraciones a la hora de trabajar con ellos.

4.1.1. Blender

Blender [2] es un programa multiplataforma dedicado especialmente al **modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales**, entre otras cosas. Su utilización en este proyecto ha sido de gran importancia debido a la necesidad de representar en 3d los Scorbots ER IX y V Plus. Para el modelado se utilizaron 4 imágenes de referencia sacadas del los manuales de usuario respectivos, *"SCORBOT-ER IX User's Manual" [6]* y *"SCORBOT ER 5Plus" [7]*.

En el modelado de ambos Scorbots se ha utilizado una jerarquía de padres e hijos. Esto es importante porque cuando un padre se mueve, el hijo mantiene su posición con respecto al padre. Este comportamiento es el que tienen los Scorbots por lo que construyendo la jerarquía adecuada obtenemos la lógica propia de los Scorbots.

Versión de Blender: Blender 2.79

4.1.1.1. Modelo del Scorbot ER IX

En este modelo la jerarquía de piezas empieza en *"ScorbotERIX"*, que se sitúa en el centro del círculo más cercano al suelo del Scorbot. Las partes claves del modelo que representarán las articulaciones del Scorbot son:

- Base
- Shoulder
- Elbow
- Pitch
- Roll

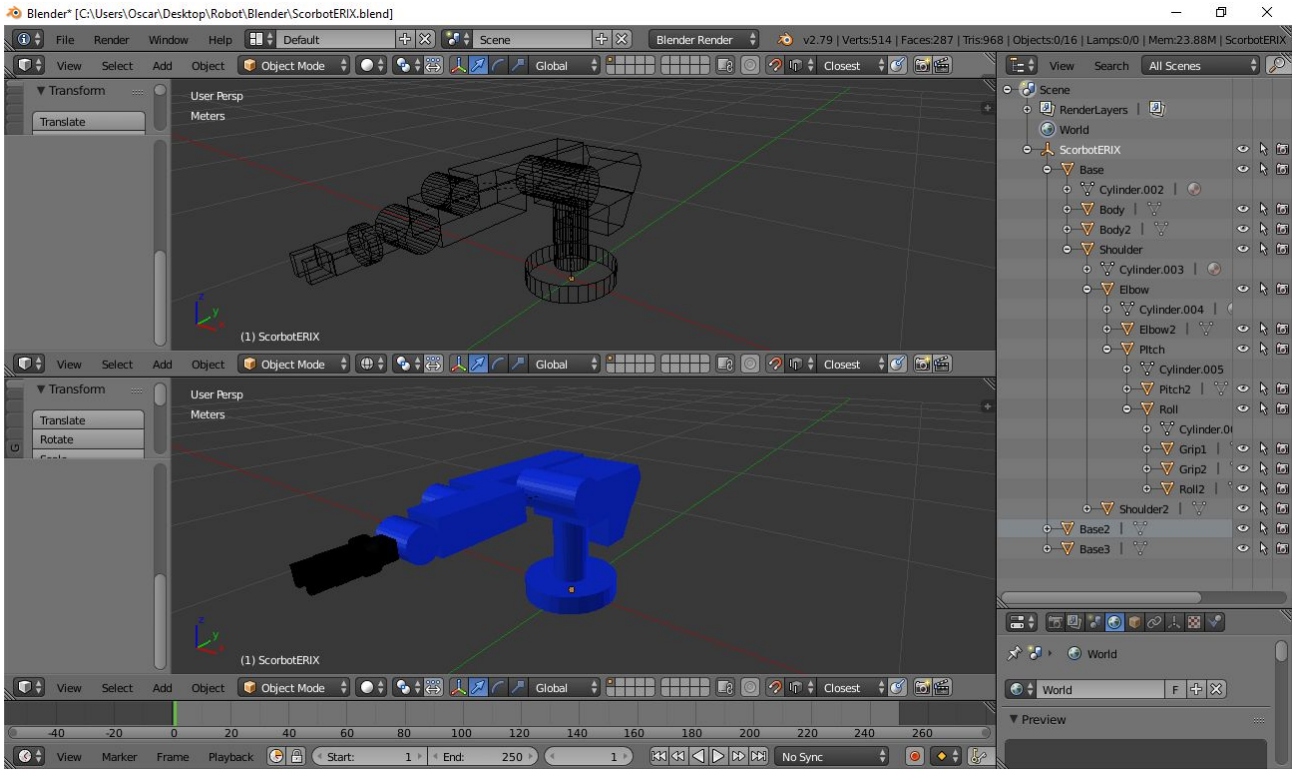


Figura 4.1: Modelo del Scorbot ER IX en Blender junto a la jerarquía de piezas que lo forman.

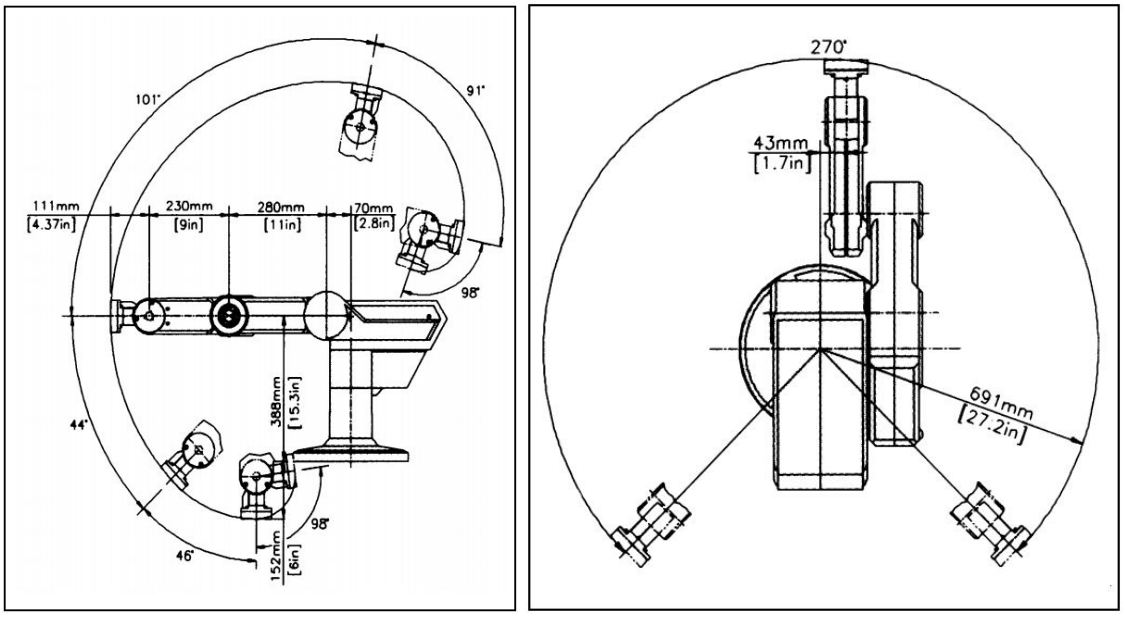


Figura 4.2: Imágenes de referencia del Scorbot ER IX utilizadas en Blender para su modelado.

4.1.1.2. Modelo del Scorbot ER V Plus

En este modelo la jerarquía de piezas empieza en *"ScorbotERVPlus"*, que se sitúa en el centro del círculo más cercano al suelo del Scorbot. Las partes claves del modelo que representarán las articulaciones del Scorbot son:

- ➢ Base
- ➢ Shoulder
- ➢ Elbow
- ➢ Pitch
- ➢ Roll

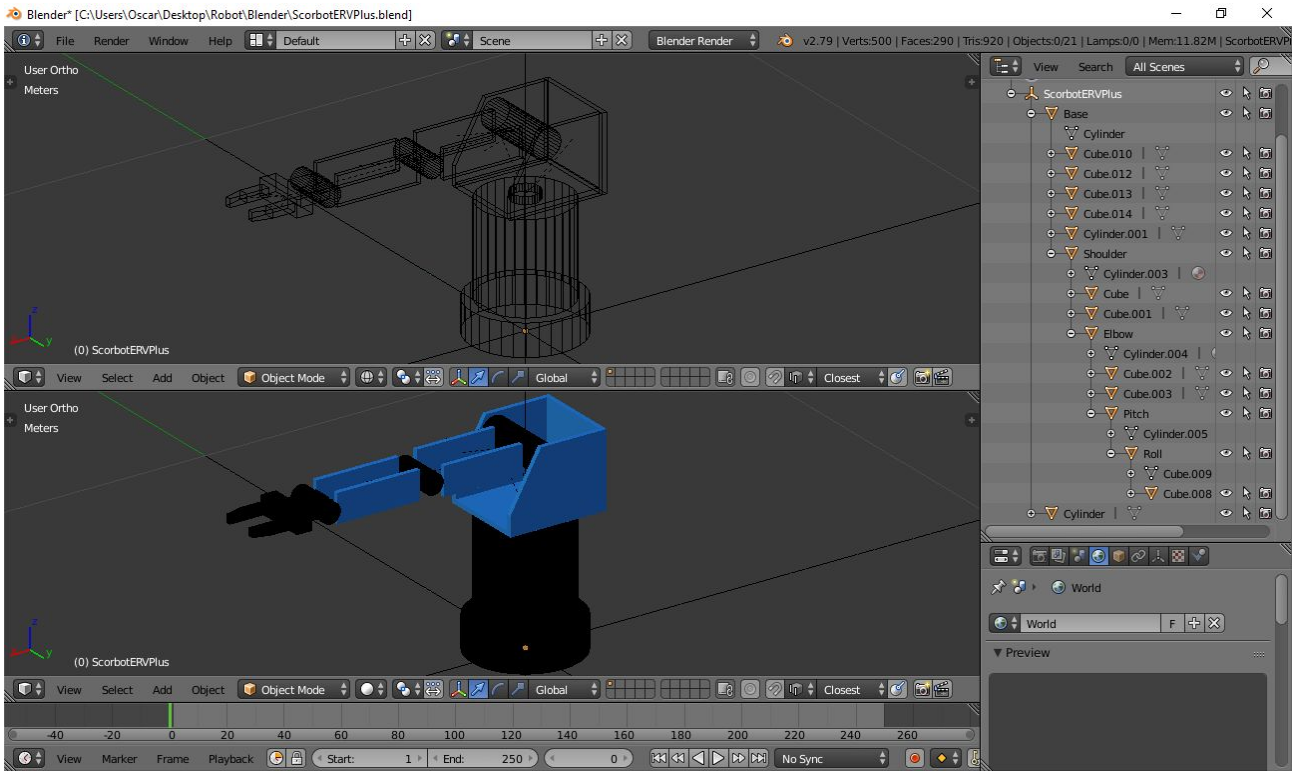


Figura 4.3: Modelo del Scorbot ER V Plus en Blender junto a la jerarquía de piezas que lo forman.

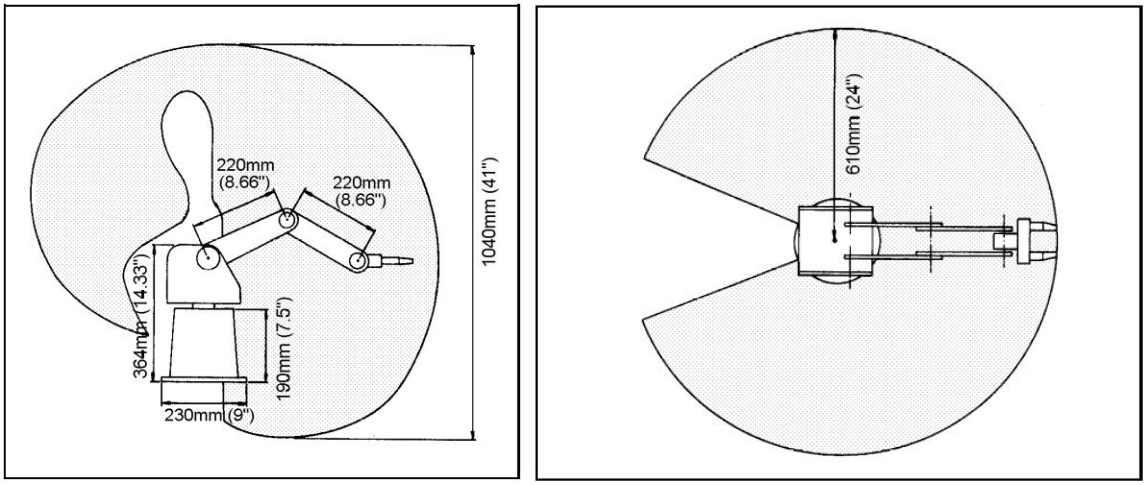


Figura 4.4: Imágenes de referencia del Scorbot ER V Plus utilizadas en Blender para su modelado.

4.1.1.3. Corrección de errores

Blender [2] es compatible con Unity [1] utilizando los propios archivos de Blender [2], sin embargo existen **problemas de escala** que hay que tener en cuenta a la hora de utilizar modelos en Unity [1]. Además, los ejes del **sistema de coordenadas en Unity son distintos** siendo *"y"* el eje vertical, mientras que en blender es el eje *"z"*. Estos problemas se han solucionado relativamente fácil desde Blender [2] mediante el escalado del modelo y la rotación del modelo.

4.1.2. Unity

A grandes rasgos, Unity [1] es un **motor de videojuegos multiplataforma**. En este proyecto se ha utilizado una única escena principal que es el entorno virtual donde se encuentran los Scorbots, el menú principal y los controladores que ejecutan la simulación entre otros elementos.

Versión de Unity: Unity 2017.4.1f1 Personal 64 bits

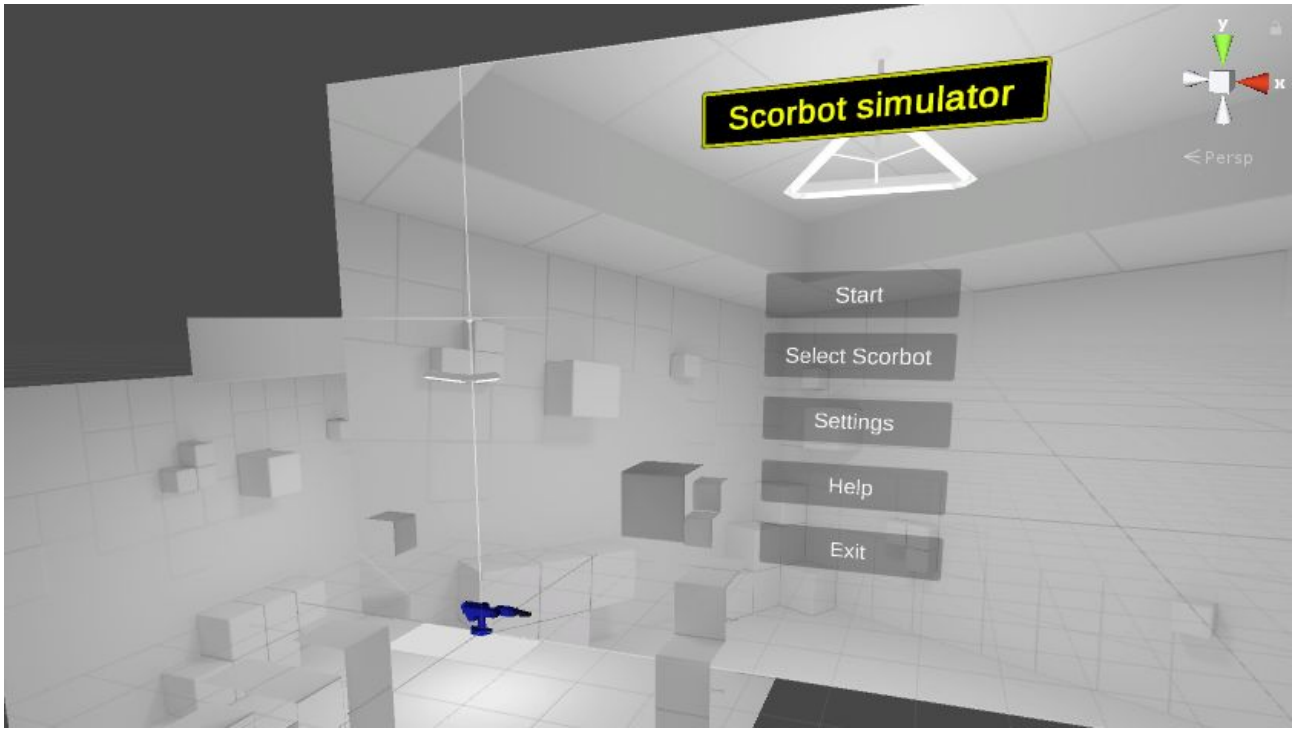


Figura 4.5: Escena principal de Unity donde se lleva a cabo el desarrollo o cambios necesarios.

4.1.3. Visual Studio

En conjunción con Unity [1], aunque se pueden utilizar varios lenguajes de programación, en este proyecto se ha usado **C#**. Además, aunque Unity [1] trae su propio editor de texto, se ha utilizado Visual Studio por su multitud de funcionalidades.

Versión de Microsoft Visual Studio Community 2017: 15.8.7

4.1.4. Estructura del proyecto

El proyecto en sí reside en Unity [1]. Se pueden diferenciar 3 partes importantes: la estructura de **carpetas**, la estructura de la **escena principal** y la estructura del **GameController**.

4.1.4.1. Estructura de carpetas

Hay un total de 9 carpetas para elementos del mismo tipo:

- **Animations**: Controladores de animaciones y clips de animaciones. Para las ventanas y el menú principal.
- **Assets**: Modelos 3d creados.
- **Editor**: Herramientas de Unity.
- **ExternalAssets**: Recursos que han sido importados.
- **Materials**: Materiales que cambian la visualización de los modelos 3d.
- **Scenes**: La escena principal.
- **Scripts**: Todos los scripts C# desarrollados.
  - **Conexión**: Scripts que forman parte de este proyecto (modo online), pero no han sido desarrollados por mi.
  - **Consola**: Scripts que forman parte de este proyecto (modo online), pero no han sido desarrollados por mi.
  - **Controllers**: GameController y otros controladores que ejecutan la simulación.
  - **Enum**: Configuración de los Scorbots, además de otras definiciones.
  - **MainMenu**: Lógica del menú principal.
  - **Models**: Estructura datos de una posición.
  - **Other**: Otros scripts varios.
  - **Scorbot**: Estructura de datos de una articulación, cinemática inversa y otros scripts relacionados con los Scorbots.
- **Shaders**: Contiene efectos especiales de visualización.
- **Textures**: Contiene imágenes que son usadas como texturas.

4.1.4.2. Estructura de la escena principal

En la estructura de la escena principal se pueden ver los objetos que existen durante la simulación, de ellos cabe destacar:

- **GameController**: Contiene todos los controladores en forma de componentes. Se explica en más detalle en la siguiente sección.
- **CubeRoom**: Entorno virtual que rodea al Scorbot.
- **Target**: Ejemplo de una posición. Se utiliza como localización por defecto. Las posiciones son duplicados de este tipo de objetos, pero no este si no otro situado en la carpeta "Assets".
- **Inner**: Se utiliza para mover la pinza del Scorbot mediante cinemática inversa a través de ejes.
- **Canvas**: Interfaz gráfica que se superpone sobre la simulación. Permite acceder a todas las funcionalidades del programa.
    - **NamePanel**: Pequeña ventana de una posición que muestra su nombre y posición. Otras posiciones utilizan duplicados de este tipo de objetos, pero no este si no otro situado en la carpeta *"Assets"*.
    - **StatePanel**: Ventana de mensaje del estado actual.
    - **AxisPanel**: Ventana donde se muestra la orientación de ejes con respecto al origen de coordenadas.
    - **MainMenuButton**: Botón del menú principal.
    - **ButtonPanel**: Botones de otras ventanas.
    - **Panels**: Ventanas.
    - **FunctionalityLeftSide**. Funcionalidades de la parte izquierda de la interfaz gráfica.
    - **Menu**: Menú principal y sus componentes.

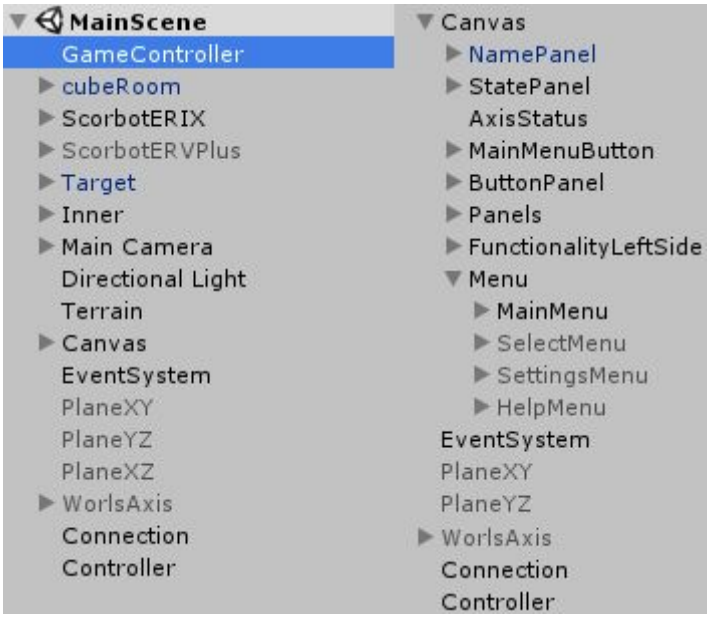


Figura 4.6: Estructura de la escena principal de los objetos que actúan durante la ejecución de la simulación.

4.1.4.3. Estructura del GameController

El GameController (objeto de la escena principal) **contiene varios componentes** (scripts), los cuales son controladores que se encargan de partes específicas de la simulación. Entre estos, el componente más importante es el GameController (script), el cual contiene a varios objetos de la escena que son utilizados por otros controladores.

Por ello, el GameController (script) es donde se conecta casi todas las partes de la simulación, exceptuando algunos casos del menú principal. El resto de componentes obtienen acceso a objetos de la escena a través del GameController (script).

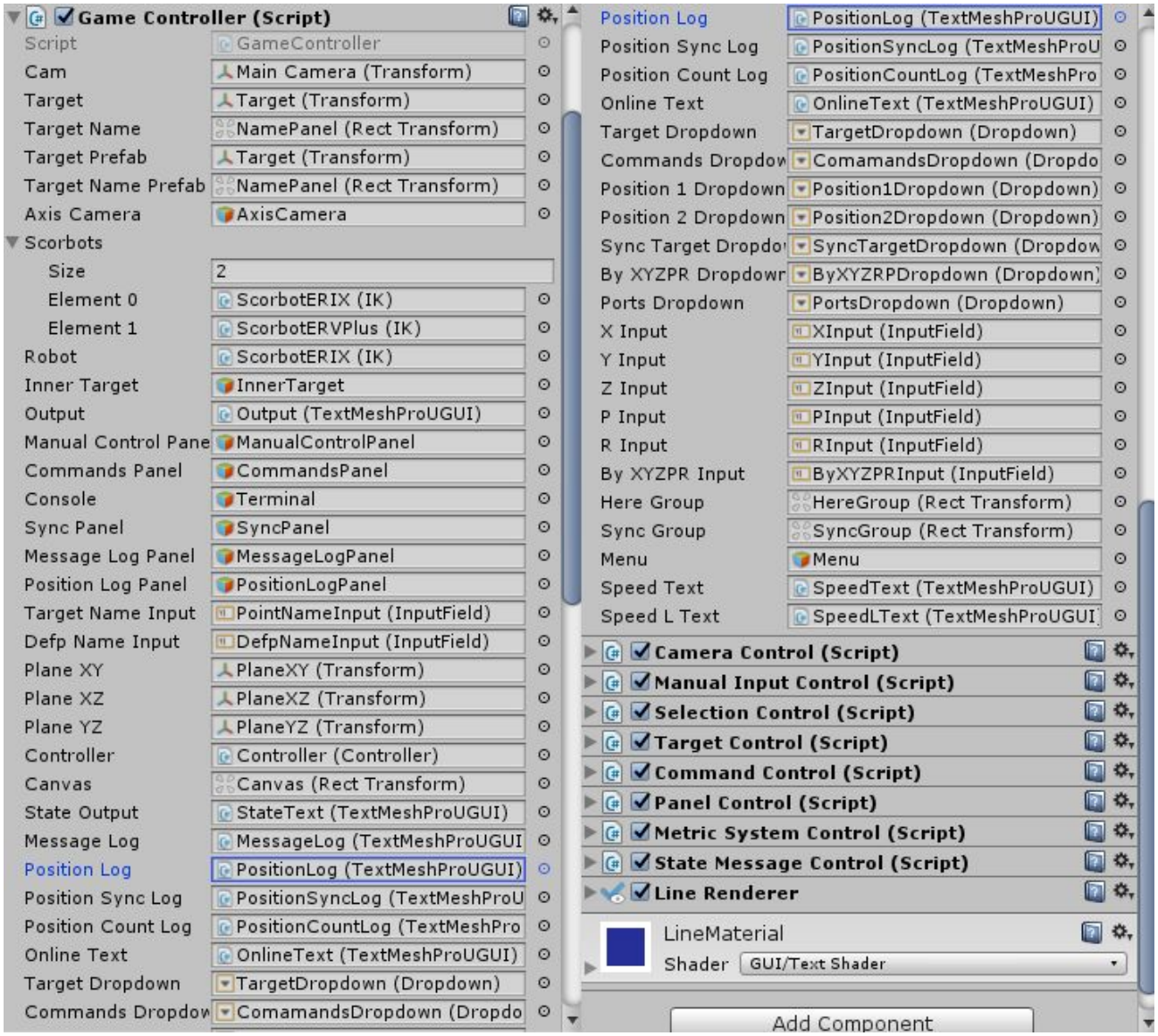


Figura 4.7: Estructura del GameController con todos sus componentes.

4.2. Algoritmos

En este capítulo se describen los algoritmos que se han utilizado en este proyecto, así como su funcionamiento.

4.2.1. CCD (Cyclic Coordinate Descent)

El algoritmo del **descenso de coordenadas cíclicas**, en inglés CCD (**Cyclic Coordinate Descent**), es una técnica de búsqueda heurística iterativa conocida como un algoritmo iterativo que resuelve el problema de la cinemática inversa.

La cinemática inversa se define como el proceso de recuperar los movimientos de un objeto a partir de una posición deseada final. Por ejemplo, esto resulta práctico aplicado a robots industriales para el control de un brazo robótico.

El funcionamiento del algoritmo en una iteración empieza desde el efector final hacia la base, de forma que cada articulación realiza un movimiento que acerca el efector final hacia la posición final. El proceso llega a su fin cuando se alcanza la posición final, se está lo suficientemente cerca de la posición final o se realizan un máximo de iteraciones.

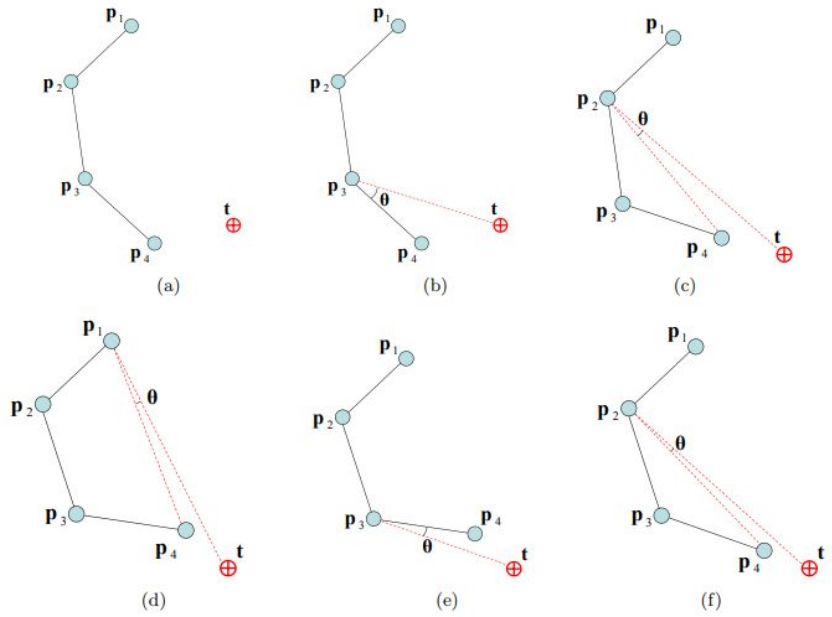


Figura 4.8: Ejemplo de iteraciones del algoritmo CCD paso a paso para acercar el efector final de un brazo robótico a un objetivo *"t"*.

En la figura 4.8 podemos ver visualmente un ejemplo usando el algoritmo CCD.

- (a): Posición inicial del manipulador y el objetivo.
- (b): Iteración 1. Encontrar el ángulo θ entre el efector final, P3 y el objetivo. Rotar la articulación P3 con el ángulo θ.
- (c): Iteración 1. Encontrar el ángulo θ entre el efector final, P2 y el objetivo. Rotar la articulación P2 con el ángulo θ.
- (d): Iteración 1. Encontrar el ángulo θ entre el efector final, P1 y el objetivo. Rotar la articulación P1 con el ángulo θ.
- (e): Iteración 2: Repetición del proceso.
- (f): Iteración 2: Repetición del proceso. Parar cuando el efector final se encuentre en la posición final, lo suficientemente cerca o se realizan el máximo de iteraciones.

En la simulación, este algoritmo se emplea para mover el Scorbot a una posición dada y hace posible la implementación de varios comandos como *"move", "movel", "teach",* etc.

4.2.2. Spline de Catmull-Rom

El algoritmo de spline de Catmull-Rom es una técnica para **interpolar curvas suavizadas mediante un polinomio cúbico**. Se puede utilizar para conectar puntos mediante una curva y, por ejemplo, formar un sistema de navegación.

El número de puntos mínimos para formar un spline de Catmull-Rom es 4 (p0, p1, p2, p3). Sin embargo, el primero (p0) y el último (p3) son puntos para controlar la forma de la curva, por lo que todos los puntos generados están entre el punto p1 y p2.

Para generar los puntos es necesario hacerlo mediante el polinomio cúbico siguiente:

$\mathbf{p(t) = 0.5 * (a + b \cdot t + c \cdot t^2 + d \cdot t^3)}$,

a = $(2 \cdot p1)$,

b = (p2-p0),

c = $(2 \cdot p0 - 5 \cdot p1 + 4 \cdot p2 - p3)$,

d = $(-p0 + 3 \cdot p1 - 3 \cdot p2 + p3)$,

donde **t** toma valores de **0 a 1**, siendo t = 0 el punto p1 y siendo t = 1 el punto p2.

En la simulación, este algoritmo se utiliza únicamente para generar la trayectoria del Scorbot en el comando “movec”. Los puntos son el centro de la articulación “Roll”, una posición intermedia y otra posición final. Por ello, se forma un grupo de 5 puntos (p0, p1, p2, p3, p4), en el que los puntos controladores (p0, p4) son iguales a sus vecinos (p1, p3, respectivamente). Por último, el algoritmo se ejecuta para dos conjuntos p0, p1, p2, p3 y p1, p2, p3, p4.

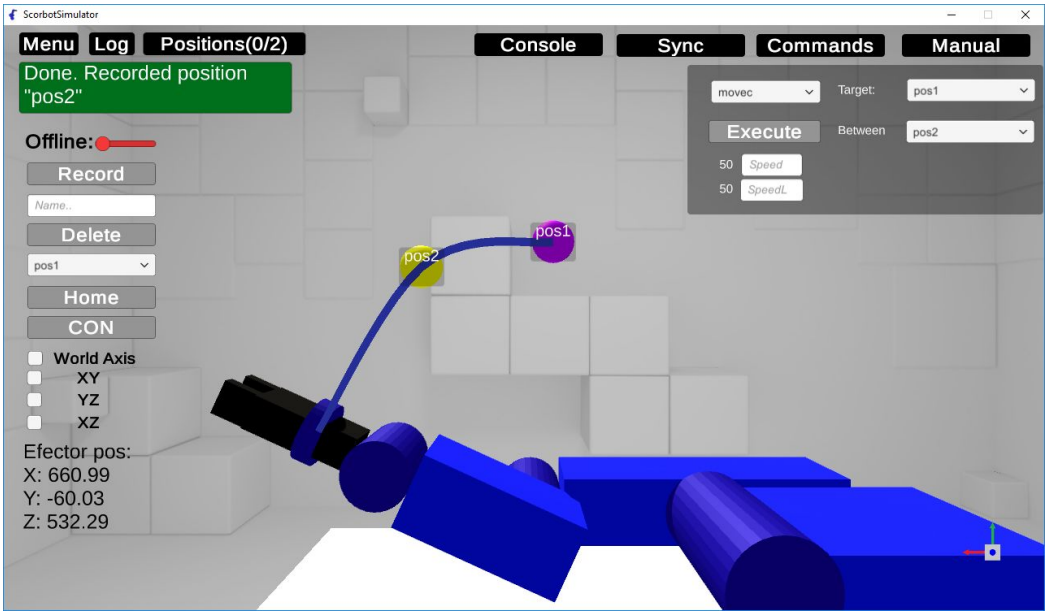


Figura 4.9: Ejemplo del Spline de Catmull-Rom para 5 puntos en la simulación.

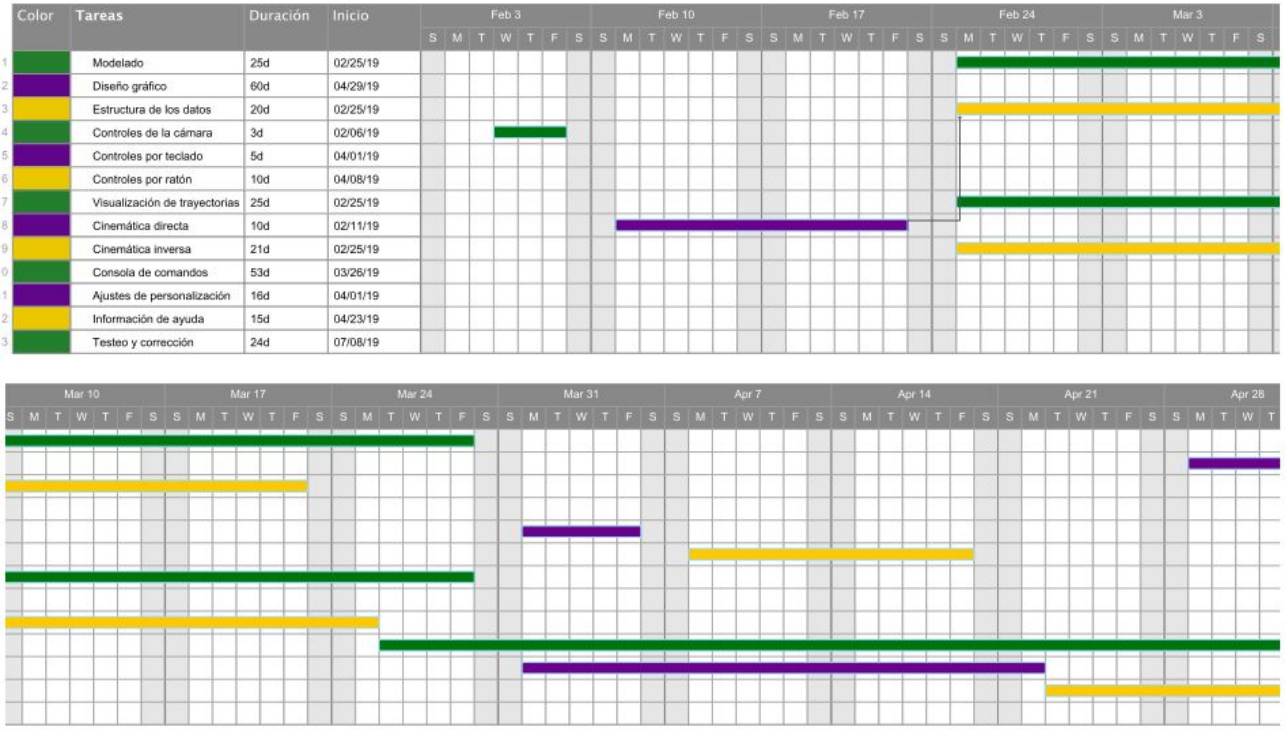
5. Fases y desarrollo del proyecto

El proyecto se desarrolló con un ciclo continuo que constó de la **implementación de diversas funcionalidades** que imitan y mejoran las funcionalidades que presenta la interfaz Scorbase [10] y el lenguaje de programación ACL [3, 4] de los manipuladores. Seguidamente, se procedió a realizar unas **pruebas con los Scorbots reales** y por último se fue **validando** dichas funcionalidades con los tutores para su mejora.

Al tratarse de dos Scorbot diferentes pero a la vez similares, se empezó por el Scorbot ER IX ya que era el más complejo. Paralelamente, se fue ajustando la interfaz para el Scorbot ER V Plus, lo cual requirió ajustes menores al ser una versión reducida del Scorbot ER IX.

En la figura 4.1 se observa el diagrama de Gantt donde se detalla la duración de cada actividad realizada, teniendo en cuenta **132 días** como tiempo total (**6 de febrero - 8 de agosto**).

Desarrollo del proyecto



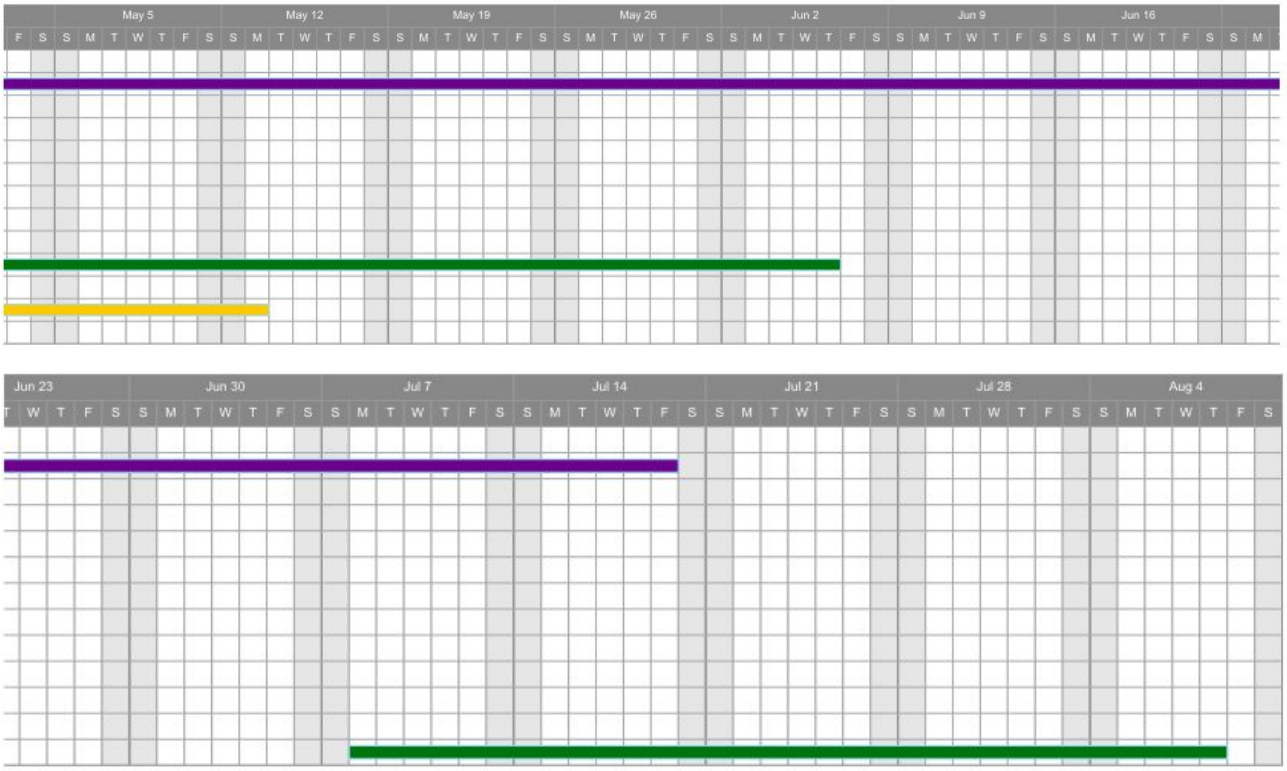


Figura 5.1: Diagrama de Gantt. Tareas durante el desarrollo del proyecto con su duración y fecha.

El desarrollo constó de **5 fases**:

- ➢ Fase 1: Se comenzó por la creación de un **prototipo** de entorno virtual conteniendo un primer modelo del brazo robótico Scorbot ER IX, con la manipulación básica del mismo (Actividad 4 y 8).
- ➢ Fase 2: Se procedió a **evolucionar el entorno virtual** con ayudas visuales y se añadió un control más avanzado para los Scorbots (Actividad 3, 7 y 9) mediante el algoritmo CCD (comando “move”). Paralelamente, se modelaron los 2 brazos robóticos finales (Actividad 1).
- ➢ Fase 3: Se continuó desarrollando una **manipulación más avanzada** con teclado y ratón de los Scorbots y se empezó a diseñar una interfaz gráfica siguiendo un diseño gráfico intuitivo (Actividad 2, 5 y 6), además de un menú principal. En paralelo se extendieron los comandos aplicables a los Scorbots (Actividad 10).
- ➢ Fase 4: Finalmente, el desarrollo terminó con la generación de **información de ayuda** y la posibilidad de modificar el programa en base a **preferencias** del usuario (Actividad 11 y 12).
- ➢ Fase 5: **Testeo** y corrección de errores con el programa completado.

6. Scorbot Simulator

En este capítulo se detalla todas las características y funcionalidades de la interfaz gráfica desarrollada, a la que se ha dado el nombre de ***"Scorbot Simulator"***. Además, en la última sección 6.8, se presenta un tutorial de uso básico de dicha interfaz gráfica.

6.1. Sistema de coordenadas

El sistema de coordenadas en la simulación se mide en **milímetros**. Los **ejes horizontales son x e y, mientras que el eje vertical es z**. Con el Scorbot extendido en horizontal podemos llegar a casi 700 milímetros, mientras que en vertical se supera por poco los 1000 milímetros.

Hay que tener en cuenta que en Unity, 1 unidad equivale a 10 milímetros para la simulación, y además los ejes son en realidad *"x, z, y"* que transformados quedan como *"x, y, z"*.

6.2. Estructuras de datos

En esta sección se describen las estructuras definidas para almacenar el flujo de datos de la simulación.

6.2.1. Posición

Las posiciones de la simulación se almacenan en objetos de tipo ***"TargetModel"***. Los datos de más importancia son:

- **Posición**: Vector de Float. Valores x, y, z del sistema de coordenadas.
- **Nombre de la posición**: String. Identifica la posición.
- **Ángulos**: Vector de Vector de Float. Conjunto de ángulos (local) para cada articulación del Scorbot
- **Pitch**: Float. Valor de la inclinación frontal o ángulo de cabeceo (global).
- **Roll**: Float. Valor de la inclinación lateral o ángulo de alabeo (global).
- **Válido**: Bool. Si la posición está dentro del alcance del Scorbot.
- **Sincronizado**: Bool. Si la posición está sincronizada con el controlador del Scorbot real.

6.2.2. Articulación

Las articulaciones del Scorbot de la simulación se guardan en objetos de tipo ***"Articulation"***. Los datos destacables son:

- **Ángulo**: Vector de Float. Ángulo actual de la articulación.
- **Plano**: String. Plano sobre el que la articulación puede rotar.
- **Número de conteos mínimo**: Cuentas del encoder del Scorbot real en el límite mínimo para la articulación.
- **Número de conteos máximo**: Cuentas del encoder del Scorbot real en el límite máximo para la articulación.
- **Conteos del HOME**: Cuentas del encoder del Scorbot real en la posición HOME para la articulación.
- **Grados**: Float. Ángulo máximo de rotación empezando en 0.
- **Offset de Unity**: Ángulo inicial para establecer el ángulo 0.
- **Velocidad mínima**: Ángulo mínimo por segundo que se puede rotar.
- **Velocidad máxima**: Ángulo máximo por segundo que se puede rotar.

6.2.3. Scorbot

En la simulación, la configuración de los Scorbots se guarda en objetos de tipo ***"ScorbotModel***". Los datos que incorpora son:

- Objetivo: Transform. Posición objetivo a la que se desea llegar.
- Efector: Transform. Posición en la que se encuentra el efector final del Scorbot.
- Articulaciones: Vector de Articulation. Conjunto de articulaciones que forman parte del Scorbot.

6.3. Controles

En cuanto a los controles, la navegación del menú principal se puede hacer mediante **ratón o teclado**, y para interactuar con la simulación es necesario un ratón. Hay que destacar que para poder usar los controles manuales de cada articulación durante la simulación, es necesario tener activa la ventana "Manual".


Botón izquierdo (ratón): Selección de un botón, una articulación (Scorbot), una posición, etc...


Botón derecho (ratón): Rotación de la cámara


Rueda (ratón): Zoom o hacia delante/atrás de la cámara.


Botón de la rueda (ratón): Movimiento de la cámara.


"Esc" (teclado): Abrir/cerrar menú principal.

 Flechas (teclado): Navegación del menú principal (arriba/izquierda, abajo/derecha).

 “1” o “q” (teclado): Control de la articulación “Base”.

 “2” o “w” (teclado): Control de la articulación “Shoulder”.

 “3” o “e” (teclado): Control de la articulación “Elbow”.

 “4” o “r” (teclado): Control de la articulación “Pitch”.

 “5” o “t” (teclado): Control de la articulación “Roll”.

 “6” o “y” (teclado): Cerrar/abrir pinza.

6.4. Comandos

Los comandos se ejecutan por defecto en modo offline, lo cual modifica solamente la simulación. Sin embargo, cuando el modo online está activado, se modifica tanto la simulación como el Scorbot real. En algunos casos puede que se ejecute solamente el modo online o offline para evitar redundancia de cálculos.

- **Home**: Mueve el Scorbot a su posición HOME.
  - Online: Mueve el Scorbot real a su posición HOME. Necesario para iniciar el Scorbot real.
- **Move**: Mueve el Scorbot a una posición ya definida. Requiere una posición.
  - Online: Mueve el Scorbot real a una posición ya definida en su controlador.
- **Movel**: Mueve el Scorbot a una posición ya definida. El movimiento es una línea recta. Requiere una posición.
  - Online: Mueve el Scorbot real a una posición ya definida en su controlador. El movimiento es una línea recta.
- **Movec**: Mueve el Scorbot a una posición ya definida pasando por otra posición. El movimiento es una curva. Requiere dos posiciones.
  - Online: Mueve el Scorbot real a una posición ya definida en su controlador pasando por otro posición. El movimiento es una curva.
- **Listpv**: Se muestran los valores de una posición los cuales son una posición (x, y, z), inclinación frontal (pitch) e inclinación lateral (roll).
- **Teach**: Modifica una posición con nuevos valores los cuales son una posición (x, y, z), inclinación frontal e inclinación lateral. Requiere una posición.
  - Online: Modifica una posición en el controlador del Scorbot real con nuevos valores los cuales son una posición (x, y, z), inclinación frontal e inclinación lateral.
- **Teachr**: Define una posición relativa a otra posición con nuevos valores los cuales son una posición (x, y, z), inclinación frontal e inclinación lateral. La posición permanecerá relativa. Requiere dos posiciones.

    - Online: Define una posición relativa a otra posición en el controlador del Scorbot real con nuevos valores los cuales son una posición (x, y, z), inclinación frontal e inclinación lateral. La posición permanecerá relativa
- **Here**: Modifica una posición con los valores actuales del Scorbot. Requiere una posición.
    - Online: Modifica una posición en el controlador con los valores actuales del Scorbot real o el de la simulación. En modo “From simulation” el Scorbot es el de la simulación, mientras que en modo “From Scorbot” es el Scorbot real.
    - Funcionamiento (online): En modo “From simulation” se ejecuta el comando “Teach” (online) con los valores del Scorbot de la simulación. En modo “From Scorbot” se ejecuta el comando “Here” (online) en el Scorbot real, seguidamente de “Listpv” (online) para recuperar los datos del “Here” y se realiza un “Teach” (Offline) para cargar esos datos.
- **Shiftc**: Modifica una posición sumando valores los cuales son una posición (x, y, z), inclinación frontal e inclinación lateral. Requiere una posición.
    - Online: Modifica una posición en el controlador del Scorbot real sumando valores los cuales son una posición (x, y, z), inclinación frontal e inclinación lateral.
- **Defp**: Crea una posición en una localización por defecto o en el mismo lugar que otra posición previamente seleccionada. Requiere el nombre de una posición.
    - Online: Define una posición en el controlador del Scorbot real. La posición creada requiere sincronización.
- **Con**
    - Online: Activa el control del Scorbot real.
- **Speed**: Modifica la velocidad del comando “Move” en el Scorbot. Límite de 1 a 100. Es necesario pulsar la tecla “Enter” tras introducir el valor deseado.
    - Online: Modifica la velocidad del comando “Move” en el Scorbot real.
- **SpeedI**: Modifica la velocidad del comando “Movel” y “Movec” en el Scorbot. Límite de 1 a 300. Es necesario pulsar la tecla “Enter” tras introducir el valor deseado.
    - Online: Modifica la velocidad del comando “Movel” y “Movec” en el Scorbot real.

6.5. Menú principal

El menú principal se puede ver después de iniciar el programa. Para su navegación se puede usar el teclado, del cual las flechas de dirección permiten subir/bajar o ir a la izquierda/derecha. También es posible usar el ratón y hacer un click izquierdo para seleccionar una opción. Por último, se puede activar/desactivar el menú con la tecla *“Esc”*.

El menú principal está formado por 5 opciones, las cuales son:

- **Start**: Empieza la simulación. El Scorbot por defecto elegido es el Scorbot ER IX.
- **Select Scorbot**: Selecciona uno de los Scorbots disponibles y empieza la simulación. Los Scorbots a elegir son el Scorbot ER IX y el Scorbot ER V Plus.
- **Settings**: Permite modificar parámetros de la simulación.
    - Cámara: La rotación, el movimiento (arriba/abajo o izquierda/derecha) y el zoom (adelante/atrás). Todo esto afecta a la navegación por la simulación.
    - Selección: La rotación de una articulación y el movimiento de una posición a través de los ejes.

- Puerto: Lista con los puertos disponibles para usar como conexión al Scorbot real.
- Entorno: La activación/desactivación del entorno de simulación.

➢ **Help**: Permite ver información de ayuda.
- Controles: Breve descripción de los controles que se pueden utilizar en este programa.
- Comandos: Breve descripción sobre los comandos disponibles para usar.
- Sobre este programa: Información sobre la versión de Scorbot simulator y el entorno de desarrollo.

➢ **Exit**: Sale del programa.

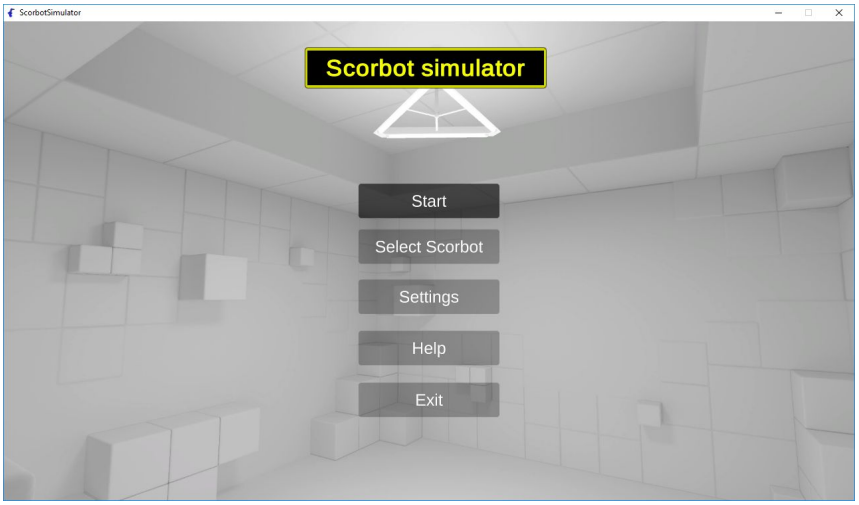


Figura 6.1: Menú principal con las opciones iniciales al arrancar la interfaz gráfica.

6.6. Interfaz gráfica

Durante la simulación, es necesario utilizar la interfaz gráfica para acceder a las funcionalidades del programa. La interfaz gráfica está compuesta por 5 partes:

➢ **Ventanas**: Activan/desactivan pequeñas ventanas que contienen funcionalidades.
➢ **Estado actual:** Contiene información acerca de la última acción realizada ya sea con éxito o fracaso.
➢ **Funcionalidades básicas**: Contiene funcionalidades elementales para crear y eliminar posiciones, además del *"Home"* que mueve el Scorbot a su posición inicial y *"CON"* que activa los controles del Scorbot real**.** Por último, *"Efector pos"* es la posición del efector del Scorbot.
➢ **Orientación**: Orientación ortogonal con respecto al origen de coordenadas.
➢ **Simulación**: Espacio virtual con un sistema de coordenadas en cuyo origen se encuentra el Scorbot.

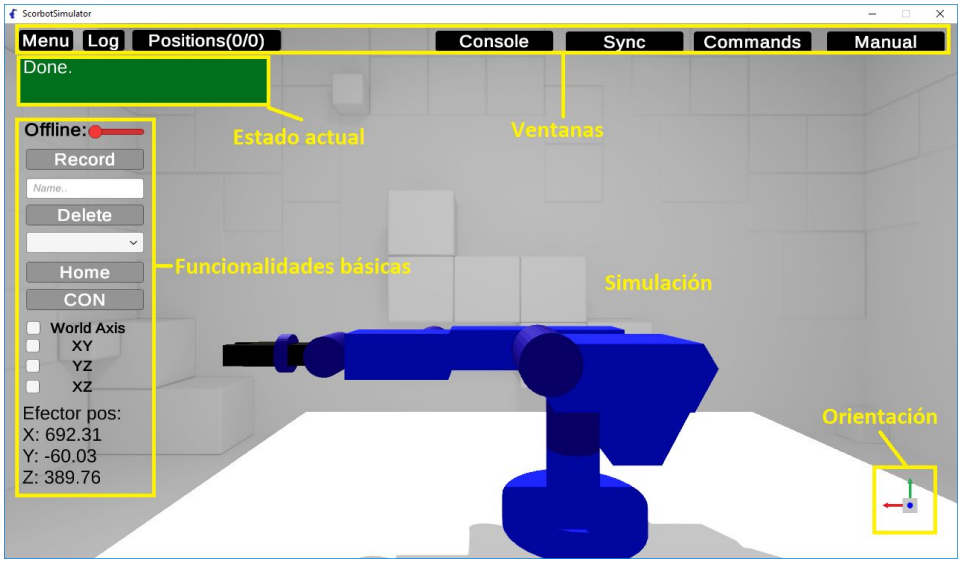


Figura 6.2: Partes de la interfaz gráfica. Ventanas, estado actual, funcionalidades básicas, simulación y orientación.

6.6.1. Ventanas

Las ventanas se activan/desactivan haciendo click izquierdo sobre uno de los botones de la parte superior de la interfaz. Debido a que utilizan el mismo espacio en pantalla, al activarse solo puede estar activa una ventana a la vez, exceptuando ”Console”.

Hay un total de 7 ventanas activables:

- **Menu**: Abrir/cerrar menú principal.
- **Log**: Registro de mensajes sobre el estado de las acciones realizadas. Cada mensaje está prefijado con el número de operación que es. Algunos mensajes consecutivos pueden tener el mismo prefijo, lo que significa que pertenecen a un mismo bloque de acciones como resultado de la ejecución de un único comando. Límite máximo de 100 mensajes que una vez superado elimina el mensaje más antiguo. Los **bloques de mensajes de “Log”** se explican en mayor detalle en la siguiente **sección “6.6.2 Estado actual”**.
- **Positions(0/0)**: Registro de todas las posiciones creadas, marcando las que no están sincronizadas con el Scorbot real con *“No sync”* en rojo. El formato 0/0 es el número de posiciones que están sincronizadas entre el número total de posiciones.
- **Console**: Consola que permite enviar comandos al Scorbot real.
- **Sync**: Sincroniza posiciones del simulador hacia el Scorbot real (modo *“From simulation to Scorbot”*) o viceversa (modo *“From Scorbot to simulation”*). Se puede hacer sobre una única posición (*“Sync now”*) o todas a la vez (*“Sync all”*). Requiere estar en modo online.
  - Modo *“From simulation to Scorbot”*: Se define la posición con *“defp”* (online) en el controlador del Scorbot real para luego ejecutar un “teach” (online) en el controlador.
  - Modo *“From Scorbot to simulation”*: Se realiza un *“listpv”* (online) en el controlador del Scorbot real para luego hacer con los datos leídos un *“teach”* (offline) en el simulador.
- **Commands**: Selección de un comando mediante una lista desplegable para luego ejecutar mediante el botón *“Execute”*. Tras cambiar a un comando aparecen los

elementos que se permiten como entrada de datos para ese comando. Estos elementos pueden ser listas desplegables, casillas check y campos de entrada de texto o números (enteros o decimales).

También contiene dos campos para cambiar los 2 tipos de velocidad del Scorbot (*"speed"*, *"speedl"*). Para ambos tipos de velocidad es necesario pulsar la tecla *"Enter"* tras introducir el valor deseado.

- **Manual**: Controles para mover las articulaciones del Scorbot de la simulación de forma individual. Esta ventana tiene que estar activa para poder utilizar las siguientes teclas .
  - *"1"* o *"q"* (teclado): Control de la articulación *"Base"*.
  - *"2"* o *"w"* (teclado): Control de la articulación *"Shoulder"*.
  - *"3"* o *"e"* (teclado): Control de la articulación *"Elbow"*.
  - *"4"* o *"r"* (teclado): Control de la articulación *"Pitch"*.
  - *"5"* o *"t"* (teclado): Control de la articulación *"Roll"*.
  - *"6"* o *"y"* (teclado): Cerrar/abrir pinza.

6.6.2. Estado actual

Este componente de la interfaz se encarga de **comunicar mediante mensajes el estado** en el que se encuentra la **última acción ejecutada** ya sea con éxito, lo cual se representa con el color verde y un mensaje con el prefijo *"Done."*, o con fracaso, cuyas características son el color rojo y un mensaje con el prefijo *"Error"*. Todos estos mensajes se añaden automáticamente a la ventana "Log" en forma de historial de mensajes.

Sin embargo, debido a la naturaleza de algunas acciones, un mensaje puede o no formar parte de un bloque de mensajes. Para ver los **bloques** es necesario activar la ventana *"Log"* y allì se puede buscar el mensaje de interés, el cual tendrá un prefijo que es el número de operación realizada. Los mensajes que tengan ese mismo prefijo pertenecen al mismo bloque y se encuentran en el orden en el que se ejecutaron.

El formato de un mensaje es:

- Prefijo: *"Done."* o *"Error."*
- *"Online"*: Sí contiene esta palabra es que es una operación que se ejecutó en el controlador del Scorbot real.
- Nombre de la operación (Ej: *"MOVE"* o *"SYNC(LISTPV)"*): En el primer caso se trata de una operación simple, mientras que en el segundo tenemos un bloque de acciones de *"SYNC"* en el que se ejecutó la acción *"LISTPV"*.

6.6.3. Funcionalidades básicas

Las funcionalidades básicas se encuentran en la **parte izquierda de la interfaz gráfica**. Ahí tenemos acceso a la creación de posiciones mediante el botón ***"Record"*** y un campo para escribir el nombre de una posición. Dicho nombre debe ser válido, lo cual es posible sólo si el nombre tiene una longitud máxima de 5 caracteres y no está repetido. La nueva posición se creará en una posición por defecto o en la misma localización de otra posición

previamente seleccionada. Por otro lado, para la eliminación de posiciones se puede usar el botón ***"Delete"*** en conjunción con una lista desplegable. La posición seleccionada en la lista se representa mediante una espera de color violeta, mientras que las no seleccionadas son de color amarillo.

También están visibles el comando *"home"* y *"CON"*, el primero move el Scorbot a su posición HOME y el segundo sólo funciona en modo online ya que permite la activación del control sobre el Scorbot real.

Además hay 4 casillas check de las que podemos distinguir dos grupos. En uno está ***"World axis"***, el cual permite ver el origen del sistema de coordenadas en la simulación a través de 3 axis de color rojo (x), azul (y) y verde (z). En el otro grupo tenemos ***"xy", "yz"*** **y *"xz"***. Estos son planos cuadriculados que ayudan a posicionar mejor el desplazamiento de las posiciones. Cada cuadrícula es de 100 milímetros de lado, aunque si se está lo suficientemente cerca, los planos se subdividen en cuadrículas de 10 milímetros de lado. Hay que destacar que solamente puede estar activo un plano a la vez.

Por último, podemos ver ***"Efector pos:"***, que muestra la posición exacta del efector final del Scorbot.

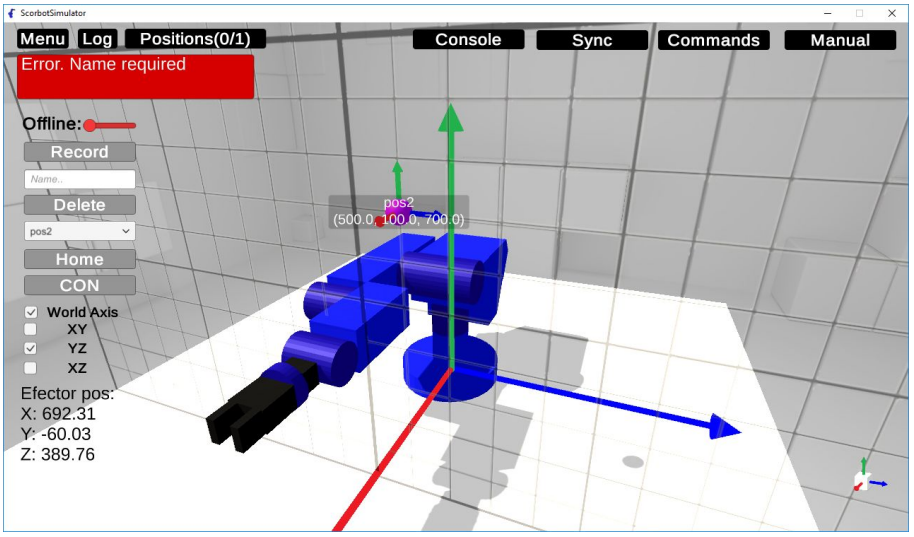


Figura 6.3: World axis del sistema de coordenadas y planos cuadriculados de la simulación.

6.6.4. Orientación

La orientación con respecto al sistema de coordenadas siempre está visible en la parte inferior derecha de la interfaz gráfica. Es la misma orientación que la casilla check *"World axis"*.

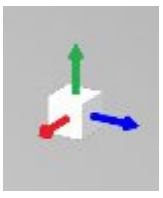


Figura 6.4: Orientación del sistema de coordenadas.

6.6.5. Simulación

El **Scorbot** se encuentra en el origen del sistema de coordenadas de la simulación. Las **posiciones** creadas se pueden visualizar con una esfera y pueden ser seleccionadas para desplazarlas. También pueden seleccionarse las **articulaciones** del Scorbot y su pinza. En el caso de la selección de una posición, aparte del nombre del punto, se puede ver la posición en la que se encuentra dentro del sistema de coordenadas.

Cuando se selecciona un elemento, aparecen **ejes o flechas curvadas** que permiten la interacción con el objeto seleccionado. En el caso de los ejes se puede mover el elemento hacia cualquier sentido de la dirección de un eje si se mantiene pulsado el click izquierdo del mouse sobre dicho eje. En el caso de las flechas curvadas se puede rotar un elemento, nuevamente mantiene pulsado el click izquierdo del mouse sobre una flecha y moviéndola a la izquierda/derecha.

Hay que tener en cuenta que existen limitaciones en el Scorbot, ya que sus articulaciones tienen límites inferiores y superiores. Cuando se llega a estos límites, las articulaciones no se moverán a una posición inalcanzable. De igual manera, aunque las posiciones pueden moverse fuera del alcance del Scorbot, estás serán inválidas si no están dentro del alcance, por lo que algunas acciones no podrán realizarse.

6.7. Controladores

En esta sección veremos los componentes que hacen funcionar la simulación y sus funciones principales. Hay un total de **9 componentes que llamaremos controladores**. Entre éstos destaca “GameController”, el cual es el componente principal de todo el programa.

- GameController
- CameraControl
- CommandControl
- ManualInputControl
- MetricSystemControl
- PanelControl
- SelectionControl
- StateMessageControl
- TargetControl

Los controladores no son más que scripts, los cuales son componentes de un objeto en la escena principal llamado *“GameController”*. Para evitar confusión, hay que destacar que este objeto (*“GameController”*) a su vez tiene un componente o controlador llamado también *“GameController”*.

6.7.1. GameController

El **controlador principal** de la simulación es el *"GameController"*. Su función principal es la de inicializar la simulación y ejecutar los demás controladores para que realicen tareas específicas. Su estructura puede verse en más detalle en la sección *"3.1.4.3 Estructura del GameController".*

Entre sus funciones, también está la creación y eliminación de las posiciones, el manejo de los elementos de la interfaz gráfica que utiliza cada comando y la propia ejecución de los comandos. Además, se encarga del dibujado de trayectorias para "movel" y "movec" con ayuda del algoritmo de spline de Catmull-Rom.

6.7.2. CameraControl

La función principal de este componente es permitir el **control de la cámara** para hacer posible la navegación en la simulación mediante el ratón.

6.7.3. CommandControl

La función principal de este componente es definir las **tareas a realizar por cada comando y ejecutarlas** ya sea en offline u online. La sincronización de posiciones es otra de sus funciones, ya que aunque no es un comando en sí, la sincronización es una sucesión de comandos, por lo que se la trata como una extensión de los comandos.

6.7.4. ManualInputControl

La función principal de este componente es permitir **controlar las articulaciones** del Scorbot de forma individual mediante el teclado, además del cierre y apertura de la **pinza**.

6.7.5. MetricSystemControl

La función principal de este componente es proporcionar un **sistema de medición** a la hora de mover una posición mediante sus axis a través de la visualización de planos. Hay tres planos (*"xy", "xz", "yz"*) que son activables/desactivables mediante casillas check y también están cuadriculadas. Cada cuadrícula es de 100 milímetros de lado, aunque si se está lo suficientemente cerca, los planos se subdividen en cuadrículas de 10 milímetros de lado.

6.7.6. PanelControl

La función principal de este componente es **gestionar las ventanas** de la interfaz gráfica mediante animaciones para ponerlas visibles/no visibles.

6.7.7. SelectionControl

La función principal de este componente es **seleccionar elementos** de la simulación a través de un historial de elementos seleccionados. Entre estos elementos están las articulaciones del Scorbot, su pinza y las posiciones creadas.

A su vez, al seleccionar un elemento pueden aparecer axis o flechas curvadas con las que se pueden interactuar para aplicar un desplazamiento o rotación, respectivamente.

6.7.8. StateMessageControl

La función principal de este componente es **comunicar el resultado de una acción mediante un mensaje** en el "Estado actual" de la interfaz gráfica. También se encarga de mantener el registro de estos mensajes como un historial.

6.7.9. TargetControl

La función principal de este componente es la **creación y eliminación de las posiciones** de la simulación mediante el manejo de una lista de posiciones,además de la validación de sus nombres.

6.8. Tutorial

En esta sección se presenta un tutorial básico para el uso de la interfaz gráfica que se ha llamado *"Scorbot simulator"*. Todas las funcionalidades que no se incorporan en el tutorial se pueden encontrar en detalle en las secciones anteriores de este capítulo *"6. Scorbot simulator"*.

6.8.1. Inicio del programa

Para iniciar el programa hay que ejecutar el archivo *"ScorbotSimulator.exe"*. Aparecerá una ventana de configuración en la que se puede elegir la resolución, la calidad de los gráficos y el modo pantalla completa o ventana. Para empezar pulsar *"Play!"*.

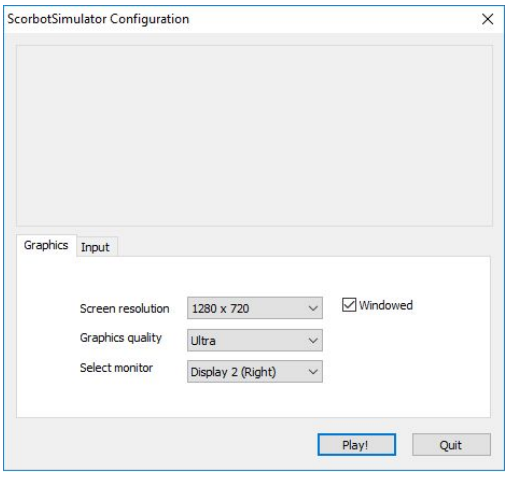


Figura 6.5: Configuración inicial al ejecutar *"ScorbotSimulation.exe"*.

6.8.2. Inicio de la simulación

Una vez el programa se haya iniciado, aparecerá el menú principal. La simulación comenzará si se pulsa *"Start"* o *"Select Scorbot"*, en este último caso se podrá elegir entre el Scorbot ER IX o Scorbot ER V Plus. En *"Settings"* se pueden modificar parámetros del programa y en *"Help"* hay información sobre los controles, comandos y sobre este programa. La tecla *"Esc"* abre y cierra el menú principal.

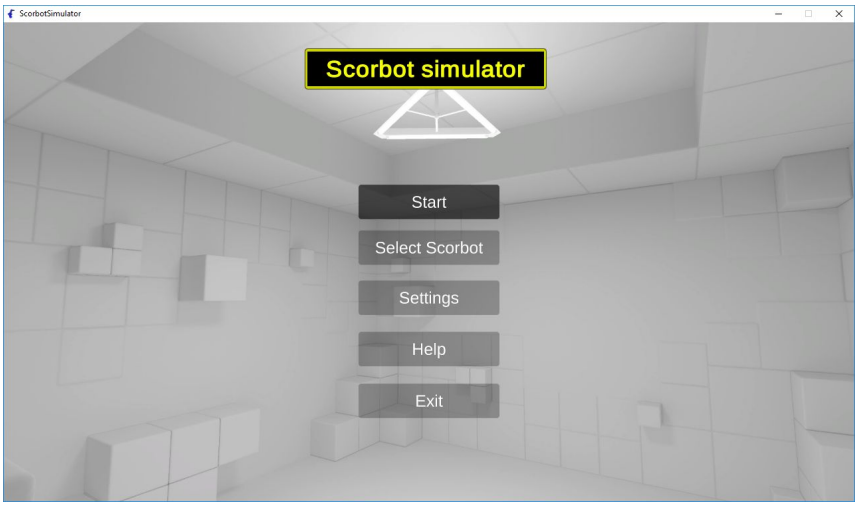


Figura 6.6: Menú principal con las opciones iniciales al arrancar la interfaz gráfica.

6.8.3. Interfaz gráfica

La interfaz gráfica está compuesta por 5 partes:

- **Ventanas**: Activan/desactivan pequeñas ventanas que contienen funcionalidades.
- **Estado actual:** Contiene información acerca de la última acción realizada ya sea con éxito o fracaso.
- **Funcionalidades básicas**: Contiene funcionalidades elementales para crear y eliminar posiciones, además del *"home"* que mueve el Scorbot a su posición inicial y *"CON"* que activa los controles del Scorbot real. Por último, *"Efector pos"* es la posición del efector del Scorbot.
- **Orientación**: Orientación ortogonal con respecto al origen de coordenadas.
- **Simulación**: Espacio virtual con un sistema de coordenadas en cuyo origen se encuentra el Scorbot.

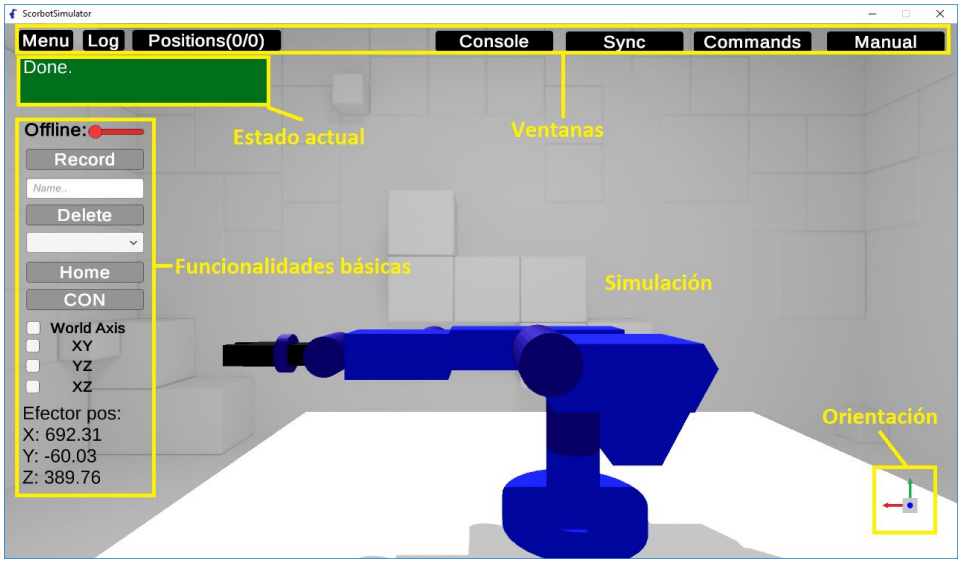


Figura 6.7: Partes de la interfaz gráfica. Ventanas, estado actual, funcionalidades básicas, simulación y orientación.

6.8.4. Creación de una posición

Para crear una posición, hay que escribir un nombre válido y pulsar el botón *"Record"*. La posición se creará en una localización por defecto y se representará mediante una esfera de color violeta. Si se selecciona la posición creada (click izquierdo) y se procede a crear otra posición, la nueva posición estará localizada en el mismo lugar que la primera posición. Si todo ha ido bien, el mensaje de estado actual estará de color verde, si no será rojo (ejemplo, si se intenta crear una posición fuera del alcance del Scorbot).

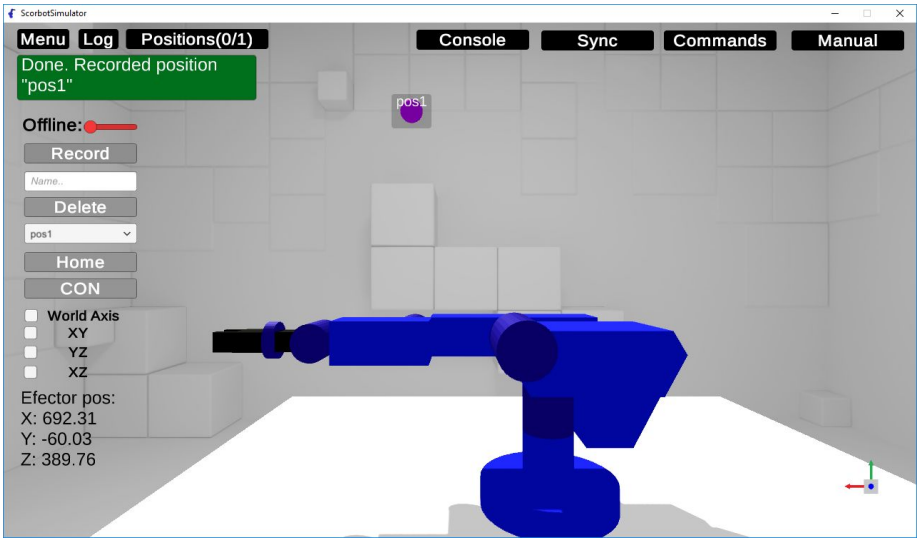


Figura 6.8: Creación de una posición mediante las funcionalidades básicas de la interfaz gráfica.

6.8.5. Manipulación de una posición

Para mover una posición basta con seleccionarla (click izquierdo), y a su vez mover seleccionando (manteniendo click izquierdo) una de las flechas que aparecerán alrededor de la posición. Estas 3 flechas representan los ejes x (rojo), y (azul) y z (verde). También se puede modificar una posición mediante los comandos *"teach"*, *"teachr"*, *"here"* y *"shiftc"*.

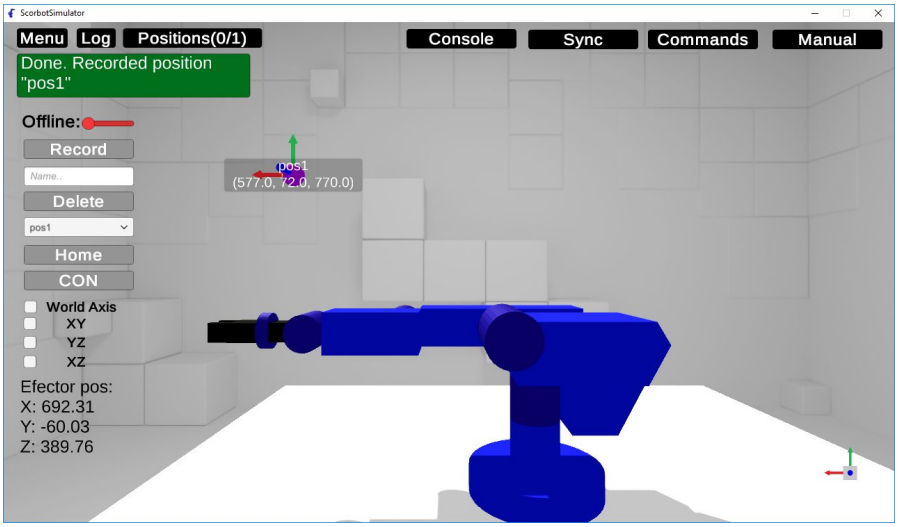


Figura 6.9: Selección de una posición mediante el ratón y sus axis.

6.8.6. Eliminación de una posición

Para eliminar una posición es necesario seleccionar mediante la lista desplegable el nombre de una posición y luego pulsar el botón *"Delete"*. Siempre que haya posiciones, una de ellas estará seleccionada (esfera violeta).

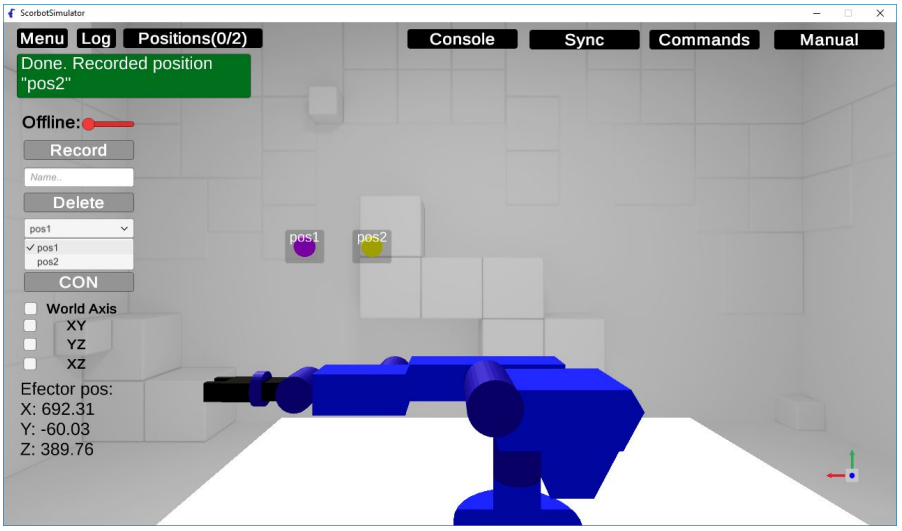


Figura 6.10: Eliminación de una posición mediante las funcionalidades básicas de la interfaz gráfica.

6.8.7. Manipulación del Scorbot

El Scorbot se puede mover manualmente al seleccionar la articulación deseada (click izquierdo) y mover la flecha roja (manteniendo click izquierdo). También es posible hacerlo mediante el teclado activando la ventana *"Manual"* y pulsando uno de los botones o teclas correspondientes.

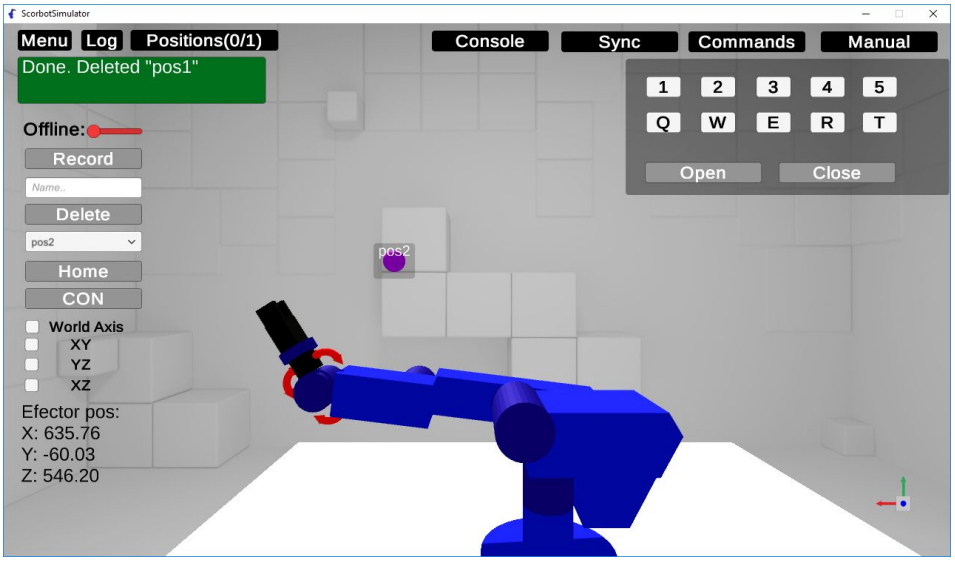


Figura 6.11: Manipulación de las articulaciones del Scorbot mediante sus flechas.

6.8.8. Movimiento del Scorbot hacia una posición

Para mover el Scorbot automáticamente hacia una posición definida, es necesario utilizar la ventana *"Commands"*. El comando más simple es *"move"*, el cual requiere elegir una posición de la lista desplegable. Para ejecutar el comando pulsar *"Execute"*. Para elegir otro comando hay que seleccionarlo de la lista desplegable.

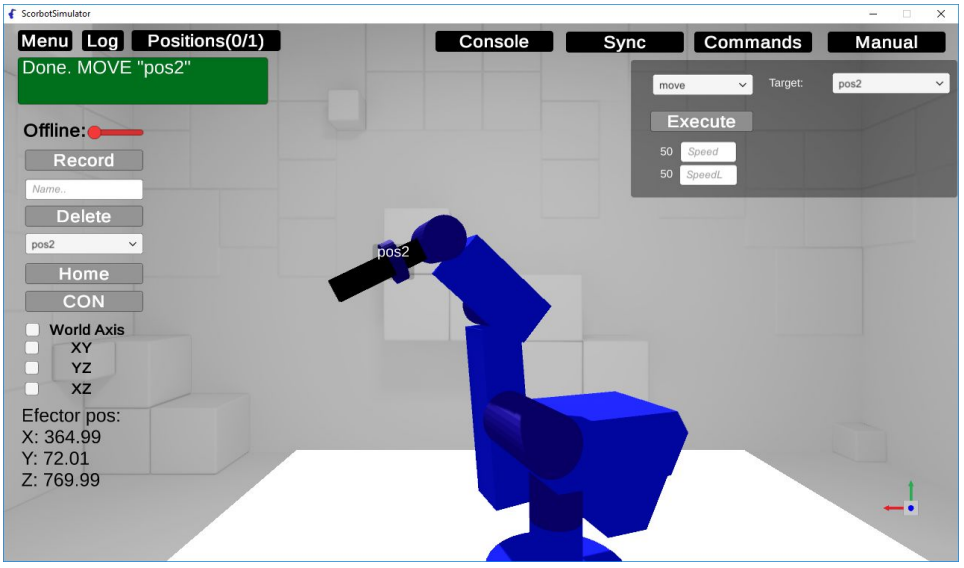


Figura 6.12: Movimiento del Scorbot hacia una posición mediante el comando *"move"*.

6.8.9. Modo online

El modo online se utiliza para poder **manipular un Scorbot real** a través de una conexión por un puerto serial. Para activarlo hace falta mover el switch rojo de izquierda a derecha. El puerto de conexión se puede cambiar a través del menú principal en *"Settings"*.

En el modo online, los comandos **se ejecutan tanto en el simulador como en el Scorbot real**. Por ejemplo, tanto *"Home"*,*"CON"* como *"move"* son comandos, por lo que

al ejecutarse *“move”*, se moverá el Scorbot de la simulación y el Scorbot real. Sin embargo, cuando se crea una posición, incluso en modo online, la posición no estará definida en el Scorbot real.

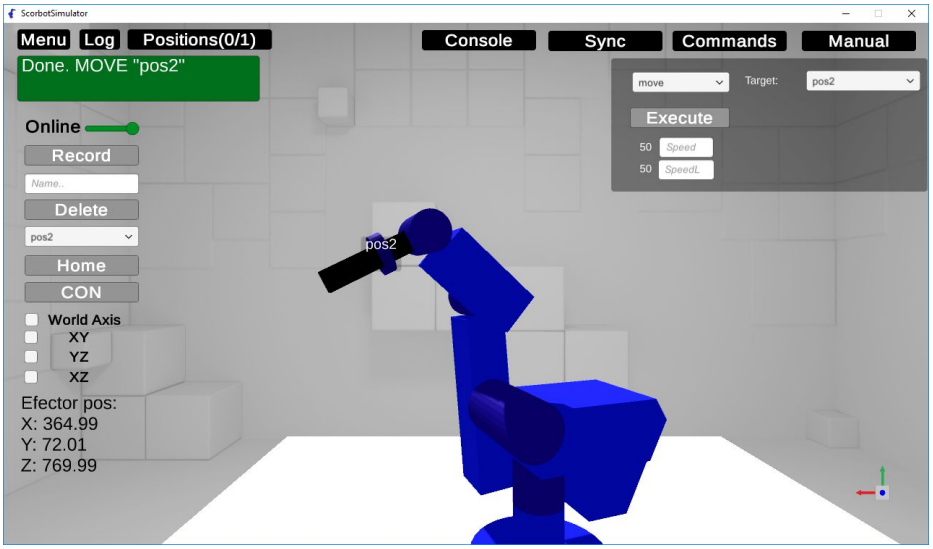


Figura 6.13: Modo online del simulador cuando está activo.

6.8.10. Sincronización de posiciones

La sincronización de posiciones se puede hacer desde la ventana *“Sync”*. Es necesario estar en modo online. Consiste en poder **enviar y recibir información sobre posiciones entre el simulador y el controlador** del Scorbot real.

Si se desea mover el Scorbot real a una posición del simulador, primera primero hay que sincronizar esa posición. Para ello, hay que seleccionar la posición en la ventana *“Sync”* y elegir la opción *“From simulation to Scorbot”* ya que queremos enviar la posición desde el simulador al Scorbot real. Seguidamente, hay que pulsar el botón *“Sync now”* para empezar la sincronización. Si se desea sincronizar todos los puntos de la simulación se puede utiliza el botón *“Sync all”* en la modalidad que se quiera.

Figura 6.14: Ventana de sincronización de las posiciones del simulador.

7. Conclusiones y líneas futuras

La interfaz gráfica desarrollada permite **simular** de forma satisfactoria los Scorbots ER IX y V Plus, permitiendo emular casi sin ninguna diferencia el comportamiento real de los mismos. Además, los comandos implementados para su **manipulación** permiten realizar las **operaciones más comunes** de movimiento de los Scorbots y los mecanismos de control con el ratón facilitan la configuración de posiciones. Por otro lado, tras la implementación del algoritmo **CCD** se ha observado que cuando el efector final de los Scorbots está **muy cerca de la posición objetivo**, éste tiende a realizar movimientos cada vez menos notables. Este comportamiento provoca **numerosas operaciones** que a lo mejor podrían ser reducidas si se aplicara otro algoritmo para resolver la cinemática inversa. Al margen de esto, el objetivo inicial del trabajo se puede dar por cumplido con la interfaz gráfica desarrollada a la que se ha dado el nombre de **"Scorbot simulator"**.

Sin embargo, la interfaz gráfica puede convertirse en un programa aún más útil con la incorporación de **nuevos comandos** ACL como por ejemplo la definición de **array de posiciones** y operaciones de movimiento sobre dichos arrays. También se podría añadir la posibilidad de tener **2 Scorbots a la vez** en una simulación para su cooperación en una única tarea. Por último, sería interesante añadir **objetos virtuales** en la simulación que pudieran interactuar con los Scorbots e incluso proyectar objetos reales a través de una **cámara** en la simulación.

8. Summary and Conclusions

The graphic interface developed can **simulate** the Scorbots ER IX and V Plus in a very similar way to them, allowing to emulate almost without any difference their real behavour. In addition, the commands implemented for **handling** allow the **most common movement operations** for the Scorbots and the mouse control mechanisms facilitate the configuration of positions. On the other hand, after the implementation of the **CCD** algorithm, it has been observed that when the end effector of the Scorbots is **very close to the target position**, it tends to make less and less notable movements. This behavior causes **numerous operations** that might be reduced if another algorithm was applied to solve the inverse kinematics. Apart from this, the initial objective of this work can be considered achieved with the graphic interface developed to which the name of ***"Scorbot simulator"*** has been given.

However, the graphic interface can become an even more useful program with the incorporation of **new ACL commands** such as the definition of **array of positions** and movement operations on these arrays. You could also allow the possibility of having **2 Scorbots at a time** in a simulation for their cooperation in a single task. Finally, it would

be interesting to add **virtual objects** in the simulation that could interact with the Scorbots and even project real objects through a camera in the simulation.

9. Presupuesto

En esta sección se detalla una **aproximación de los precios** sobre los **Scorbots** necesarios para la utilización completa de la interfaz gráfica desarrollada. Sin embargo, hay que recordar que la interfaz ofrece una simulación de los mismos y puede trabajar con ellos al igual que los harían con el Scorbot reales, por lo que **no es necesario poseer ninguno** de los Scorbots.

Tipos	Descripción	Precio
Horas de desarrollo	25 horas por semana. 20 euros/hora.	9000 euros

Tabla 8.1: Resumen de tipos.

Repositorio del proyecto

➢ Github: https://github.com/alu0100825893/ScorbotSimulator

Bibliografía

- [1] Unity: https://unity.com/
- [2] Blender: https://www.blender.org/
- [3] ACL. Advanced Control Language. Versions 1.43, F.44. Reference guide for controller-A. 4th edition.
- [4] ACL. Advanced Control Language. Version F2.28. For controller-B. Reference guide.
- [5] Controller-B. ACL version 2.28. User's Manual.
- [6] SCORBOT-ER IX. User's Manual. 2nd edition.
- [7] SCORBOT-ER 5Plus. User Manual.
- [8] ABB Robot Studio: https://new.abb.com/products/robotics/robotstudio/downloads
- [9] Kuka Sim: https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/software/planificaci%C3%B3n-proyecci%C3%B3n-servicio-seguridad/kuka_sim
- [10] ScorBase: https://www.intelitek.com/robots/robotic-software/scorbase/