Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

Evaluación de herramientas industriales para Gemelos Digitales

Autor: Pablo González Camacho

Tutor: D. Juan Manuel Escaño González

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

Sevilla, 2021

Trabajo Fin de Grado
Grado en Ingeniería en Tecnologías Industriales

Evaluación de herramientas industriales para Gemelos Digitales

Autor:
Pablo González Camacho

Tutor:
D. Juan Manuel Escaño González
Profesor Ayudante Doctor

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo Fin de Grado: Evaluación de herramientas industriales para Gemelos Digitales

Autor: Pablo González Camacho

Tutor: Juan Manuel Escaño González

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A todos aquellos que se acuerdan cada día de mí

Agradecimientos

A mi familia, de la que he recibido todo y nunca habrá palabras suficientes para agradecer todo lo que me han dado y me siguen dando. Aunque a veces no me haya dado cuenta, siempre habéis estado allí detrás de mí apoyándome con cariño.

A mis amigos, sobre todo a aquellos que lo son de verdad y siempren han estado ahí, hasta en los momentos malos.

A mis profesores, especialmente a mi tutor Juan Manuel Escaño, por haber confiado en mí desde el primer momento.

A Vicente, por acompañarme siempre todos estos años

Pablo González Camacho

Sevilla, 2021

Resumen

En el presente proyecto se ha redactado una breve introducción sobre los Gemelo Digitales y se han hecho evaluaciones de diversas herramientas industriales para la creación de los mismos. Además, se ha diseñado y programado un alimentador de piezas a modo de ejemplo del potencial que puede llegar a tener la creación de un Gemelo Digital.

Abstract

In this project, a brief introduction on Digital Twins has been written and evaluations have been made of various industrial tools for their creation. In addition, a parts feeder has been designed and programmed as an example of the potential that the creation of a Digital Twin can have.

Índice

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

El término Industria 4.0 fue oficialmente anunciado en 2013 como una iniciativa estratégica alemana para tomar un papel pionero en la industria, la cual ya estaba revolucionando el sector de la producción [1]. Hoy en día, esta cuarta revolución industrial es una realidad en todo el planeta.

El mundo físico crece a la par que el mundo virtual, y conceptos tales como el Internet de las Cosas (IdC), término que apareció ya por primera vez en 1999 y que se puede ver en la actualidad no solo en la industria, sino en una gran cantidad de hogares, conectando objetos de la vida cotidiana con Internet, vienen a lograr esa interacción entre ambos mundos. Así, los sistemas ciber físicos consiguen esta conexión mediante el uso del Internet de las cosas y servicios, haciendo posible en cada vez mayores industrias el objetivo de una producción mayor, más eficiente, con prevención de errores,… Esto es lo que se conoce hoy día como "Smart manufacturing".

Siguiendo en esta línea de continua evolución y desarrollo de todas estas tecnologías, una de las herramientas para conectar ese espacio físico con el virtual, que es una de las claves en la Industria 4.0, es el gemelo digital (denominado en inglés "digital twin", DT). Al integrar el espacio físico y el espacio virtual, un gemelo digital puede obtener una gran cantidad de datos que pueden ser procesados mediante analítica avanzada. Posteriormente, esos resultados pueden ser empleados para mejorar el desarrollo del producto o del proceso en el espacio físico [2].

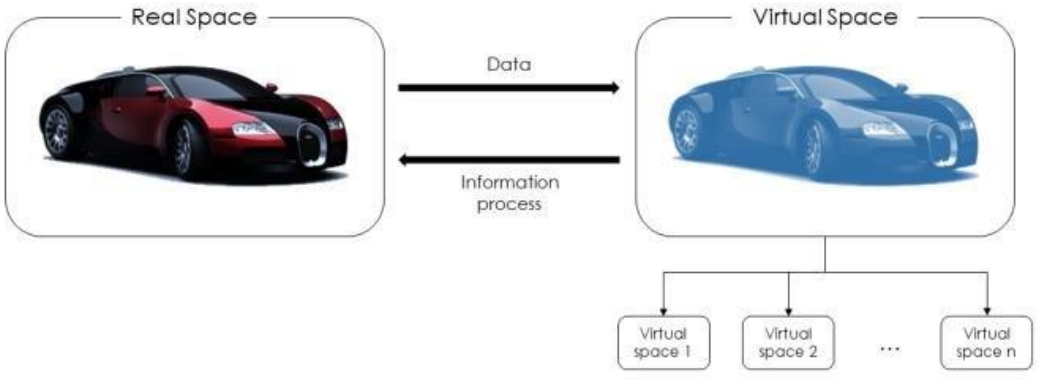

Figura 1: Modelo de un gemelo digital [3], adaptación del propuesto por M. Grieves

1.1 Antecedentes

Este concepto de gemelo digital fue propuesto por primera vez en 2003, de la mano de Michael Grieves, en un curso sobre la gestión del ciclo de vida del producto (en inglés, "Product Lifecycle Management", PLM). En este primer modelo, el gemelo digital contenía tres partes: el producto físico en un espacio real, el producto virtual en un espacio virtual y las conexiones de datos e información que unían ambos productos. Comienza

así una primera etapa de formación en lo que al desarrollo de este concepto se refiere, como se muestra en la Figura 2.

En 2011, se publica el primer artículo de investigación en una revista de ingeniería, el cual trata sobre el uso de los gemelos digitales para obtener un modelo que prediga la vida de la estructura de un avión, así como todos los retos técnicos que implican el desarrollo de un gemelo digital [4]. Se empieza a desarrollar la etapa de incubación y aparecen cada vez más artículos hablando sobre gemelos digitales.

Un año más tarde, se formaliza la definición de gemelo digital que se tenía por entonces. En un artículo publicado por la NASA, se dice que los gemelos digitales son una simulación probabilística y de alta fidelidad, con múltiples físicas y escalas, que refleja el estado de su correspondiente modelo físico, basándose en datos históricos, datos de sensores en tiempo real y datos físicos del propio modelo [5].

La etapa en la que se encuentra ahora el concepto de gemelo digital, que consiste en una etapa de crecimiento, empezó en 2014. Fue en dicho año cuando Michael Grieves publicó el primer "White paper", que reflejaba el crecimiento y la evolución del concepto de gemelo digital [6]. A partir de aquí, el impacto del gemelo digital comienza a aumentar, de manera que en 2017 y 2018, se clasifica el gemelo digital como una de las mayores tendencias tecnológicas de la próxima década (Gartner, empresa consultora y de investigación de las tecnologías de la información. Sede: Stamford, EEUU).

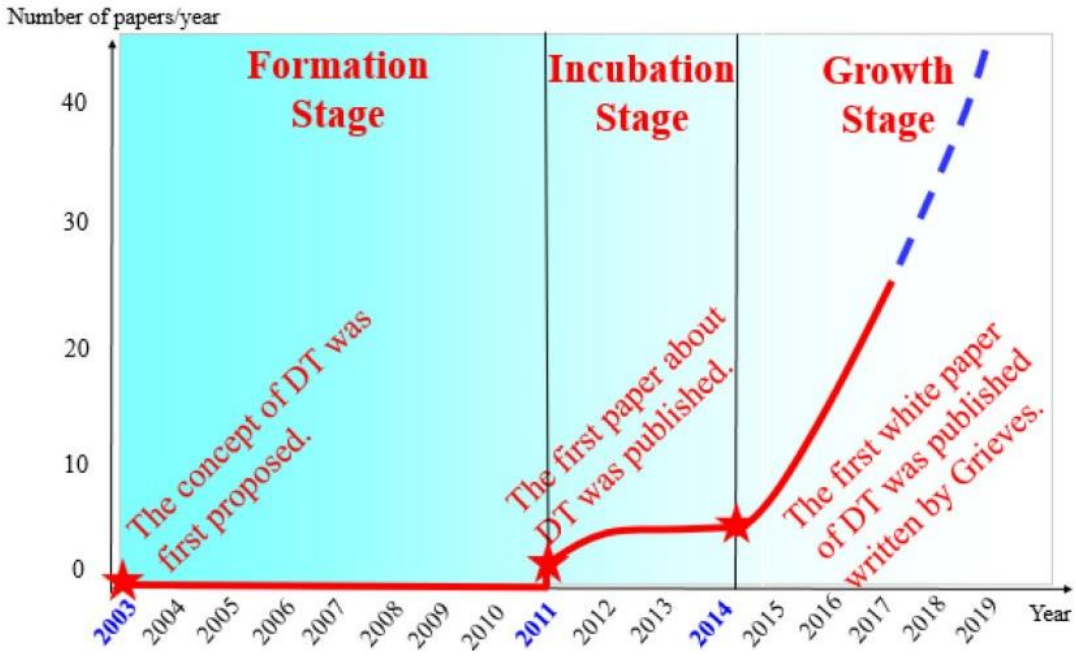

Figura 2: Tendencia de desarrollo de la investigación de los Gemelos digitales [7]

1.2 Estado del arte

En la actualidad, el gemelo digital de tres dimensiones propuesto por Grieves se ha convertido en un modelo de cinco dimensiones: una entidad física, un modelo virtual, los datos, servicios y conexiones entre ambos (Ver Figura 3).

La entidad física es la base del gemelo digital, y ese mundo físico que se quiere simular puede ser desde un simple dispositivo o producto hasta proceso industrial o incluso una fábrica industrial entera. Para realizar un buen gemelo digital, hay que conocer bien los ambientes físicos de las entidades, con sus leyes y comportamientos.

El modelo virtual debe ser lo mas parecido a la entidad física, ya que trata de reproducir la geometría física, las propiedades y reglas, simulando el comportamiento de esa entidad física en un ambiente

concreto. No solo debe ser un modelo que se parezca visualmente al modelo físico, sino que debe reflejar la física real de los objetos y los fenómenos que puedan aparecer tales como las deformaciones, las roturas, etc.

Así mismo, también debe tener un modelo de comportamiento que sepa describir lo más fielmente posible el comportamiento de la entidad física, como los estados de transición, la degradación ante el uso, etc.

Los datos del gemelo constituyen también uno de los campos más importantes, ya que los gemelos digitales tratan con datos en diferentes escalas de tiempo, datos con dimensiones diversas y de fuentes variadas. Parte de los datos son obtenidos de las entidades físicas, y pueden ser estáticos o dinámicos. Otra parte de los datos es generada por los modelos virtuales, como resultado de las simulaciones de los mismo. Se obtienen también datos de los servicios; de conocimiento que ya existía previamente, o de conocimiento de expertos en la materia; de la fusión de datos de todos los campos mencionados anteriormente… [8]

Por último, no se podría hablar de gemelo digital sin los servicios que estos proporcionan a los usuarios, tales como la verificación de una determinada característica de un producto, o la optimización energética de un proceso industrial, además de pronósticos, diagnósticos, seguimiento… Para ello, se emplean servicios de algoritmos, de datos, etc.

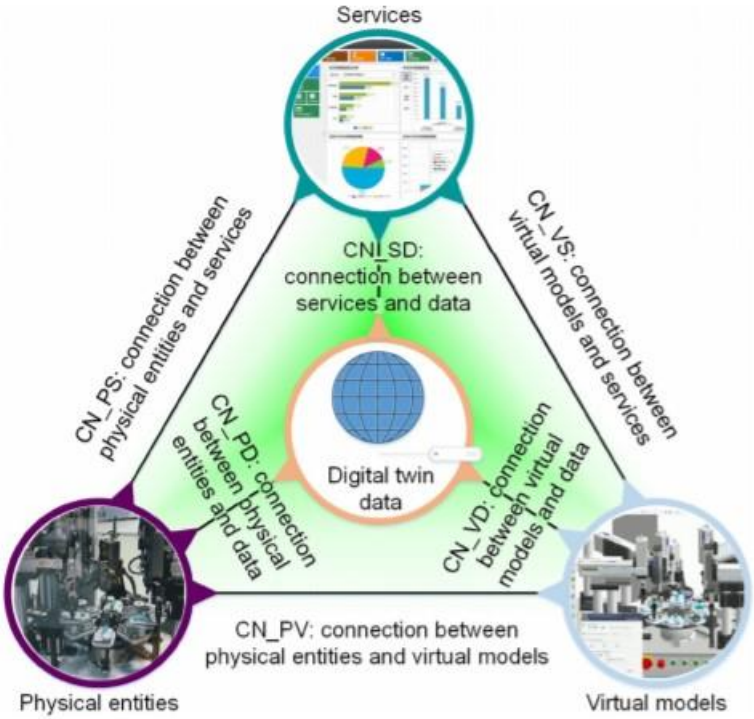

Figura 3: Modelo de gemelo digital con sus cinco elementos [8]

Todos estos campos de un gemelo digital están conectados entre sí de forma dinámica, como se puede ver en la Figura 3, de forma que permite a las cuatros partes colaborar entre sí, y ahí es donde un gemelo digital de verdad consigue todo su potencial. Gracias a dichas conexiones, se pueden establecer intercambios de datos e información, y esto permite, por ejemplo, que se enriquezca el modelo virtual creado gracias a los datos aportados por la entidad física, mientras se optimiza la puesta en marcha de un determinado proceso, gracias a la información que aportan las múltiples simulaciones hechas en el entorno virtual.

Por otro lado, son muchas las disciplinas que intervienen en los fundamentos teóricos del desarrollo de los gemelos digitales en la industria: ciencias de la información, de la computación, de datos, ingeniería de la producción, ingeniería aeroespacial, … Los cuatro grandes pilares que componen las funciones asociadas a los Gemelos Digitales pueden resumirse en: VV&A (Verificación, validación y acreditación), fusión de datos, interacción y colaboración, y servicios.

También se tienen una gran cantidad de aplicaciones industriales de gemelos digitales que han sido recogidas a través de publicaciones, patentes, etc. por líderes industriales en todo el mundo. Dentro de las aplicaciones industriales se encuentra el uso de gemelos digitales en el ciclo de vida de un producto, el cual pasa por un serie de etapas en donde son de gran utilidad los gemelos digitales.

La primera de ellas es el diseño del producto. Los gemelos digitales se pueden emplear para diseñar nuevos

productos de manera más eficiente, y contando con una mayor información.

La segunda es la etapa de producción. Mediante el empleo de gemelos digitales se consigue una producción más flexible, la cual se puede predecir y, por tanto, genera una mayor confianza. En particular, los gemelos digitales visualizar y cargar el estado en tiempo real, lo cual es muy útil para monitorizar un proceso de producción. Esto es gracias a la sincronización entre el espacio físico y virtual y, así, un operador puede usar los DTs para monitorizar un proceso complejo de producción, hacer ajustes de tiempo o, simplemente, optimizar un proceso. Además, como los gemelos digitales integran una gran variedad de datos, los sistemas autónomos pueden responder a cambios de estado, incluso durante una operación en curso [9].

Sin embargo, el área más popular de aplicaciones industriales con el uso de gemelos digitales corresponde a la tercera etapa del ciclo de vida de un producto: PHM, previsión y mantenimiento (del inglés *Prognostics and health manegement*). De hecho, los gemelos digitales fueron usados por primera vez en el PHM de aviones [7]. Se han desarrollado modelos para predecir comportamientos en un sistema ciber-físico [10] para predecir la velocidad de enfriamiento de un proceso de fabricación [11], etc.

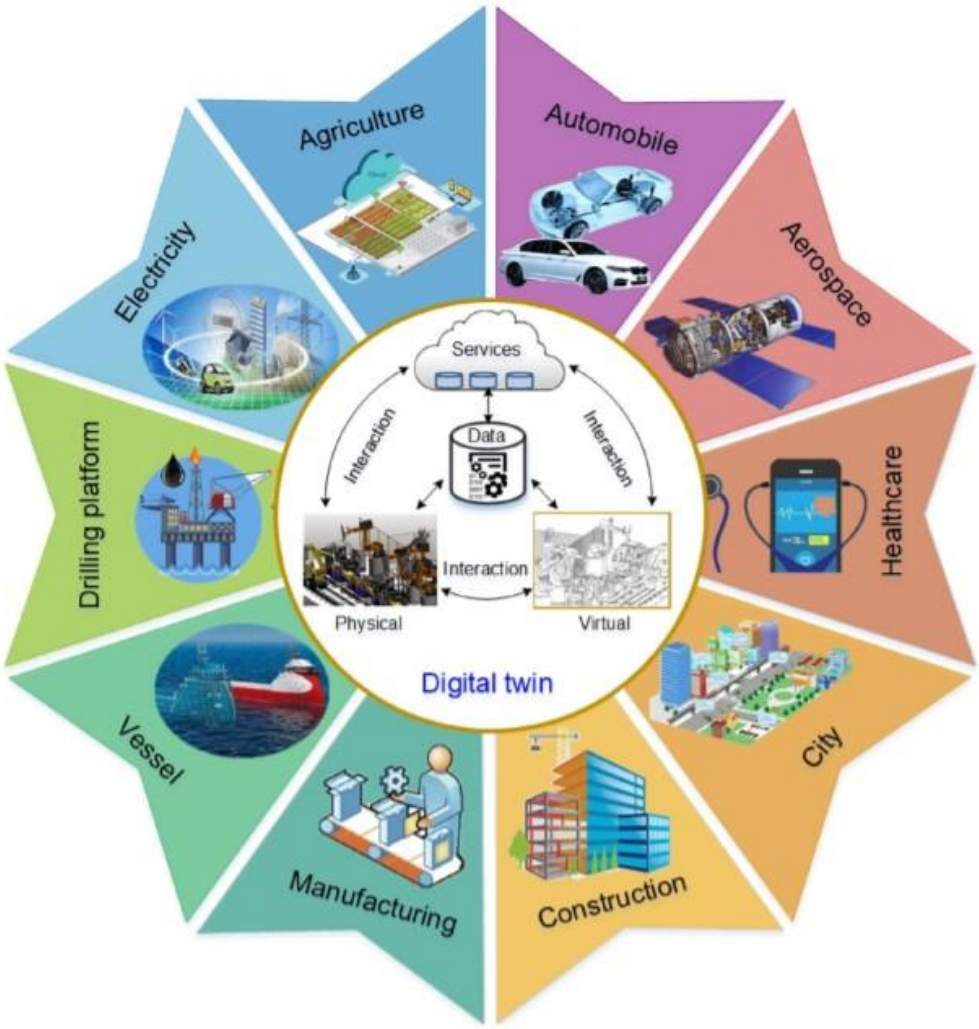

Figura 4: Diferentes aplicaciones del gemelo digital [8]

En el ámbito de las patentes, ya hay muchas empresas que poseen varias que están directamente relacionadas con gemelos digitales. En particular, y por citar algunas, *General Electric* posee cuatro patentes, dos de ellas sobre parques eólicos [12], [13]. *Siemens* también tiene la propiedad sobre cuatro patentes, las cuales ponen el foco en la relación máquina-humano [14]. Así mismo, la empresa americana *Johnson Controls* inventó y patentó un flujo sistemático para crear un gemelo digital de una habitación, lo que es muy útil para construir una fábrica digital [15].

Tras la búsqueda de puestas en marcha de DTs (Abreviatura en inglés de Digital Twins, Gemelos Digitales), la pregunta crítica es cómo construir un modelo viable. No se tiene todavía un modelo unificado de modelado de

gemelo digital, y de igual forma, es importante desarrollar más herramientas de modelado de gemelos digitales. Además, la fusión entre el espacio virtual y el físico es tema relativamente nuevo, para el que no está disponible una red de trabajo universal, y para el que todavía se tienen que abordar muchos puntos, como la robustez de los algoritmos, o la exposición de datos a un ciberataque. Aunque el modelado sea el núcleo de los gemelos digitales, la fusión ciber-física es la mayor dificultad de las aplicaciones de los mismos [7].

Por último, también cabe destacar el estado de investigación en este ámbito de los gemelos digitales por parte de la Universidad de Sevilla. Actualmente, la Universidad de Sevilla es uno de los participantes en el proyecto europeo DENiM (Digital intelligence for collaborative Energy management in Manufacturing) [16], que proporciona una serie de herramientas integradas las cuales brindan servicios digitales avanzados que incluyen Internet de las Cosas (IdC), gemelos digitales, análisis de datos, modelado energético y automatización, todo ello con el objetivo de hacer una evaluación continua del impacto energético, junto con el control y la optimización de la energía en las instalaciones, procesos y máquinas. Este proyecto desarrollará finalmente una plataforma de inteligencia digital interoperable con un enfoque colaborativo para la gestión de la energía industrial, empezando por las empresas participantes en dicho proyecto. Así mismo, aunque no traten directamente sobre los gemelos digitales, varios trabajos de fin de grado se han aproximado a este concepto mediante la simulación de modelos 3D que tratan de emular la realidad [17], [18]

Figura 5: Objetivos clave del proyecto DENiM [16]

1.3 Objetivos del TFG

El objetivo principal es dar una visión introductoria al mundo del gemelo digital y a algunas de las distintas herramientas que existen para su desarrollo. A su vez, se tiene como objetivo secundario, el desarrollo del un gemelo digital de un alimentador de piezas que forma parte de una célula de fabricación industrial que se encuentra en el laboratorio del Departamento de Ingeniería de Sistemas y Automática de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla.

2 HERRAMIENTAS INDUSTRIALES

Debido al auge de los gemelos digitales en los últimos años, son muchas las empresas que han ido creando herramientas industriales para el desarrollo de estos gemelos digitales, desde modelos sencillos, hasta algunos mucho más elaborados. Para este trabajo de fin de grado, se han evaluado herramientas de Visual Components y de Siemens, ambas orientadas a la simulación 3D.

2.1. Tecnomatix Plant Simulation (Siemens)

Tecnomatix es un conjunto de soluciones de fabricación digital. El objetivo que se pretende conseguir es la sincronización de la ingeniería de productos y de fabricación con las operaciones de servicio y la producción para maximizar la eficacia de dicha producción [19].

Dentro de Tecnomatix, el software evaluado ha sido Plant Simulation. Esta es una herramienta de simulación de eventos discretos y modelado logístico 3D que puede optimizar el flujo de materiales, la utilización de recursos y la logística en todos los niveles de planificación de la planta, ya que se pueden ir creando diferentes líneas de producción e irlas añadiendo después a una misma planta para hacer la simulación de una fábrica completa.

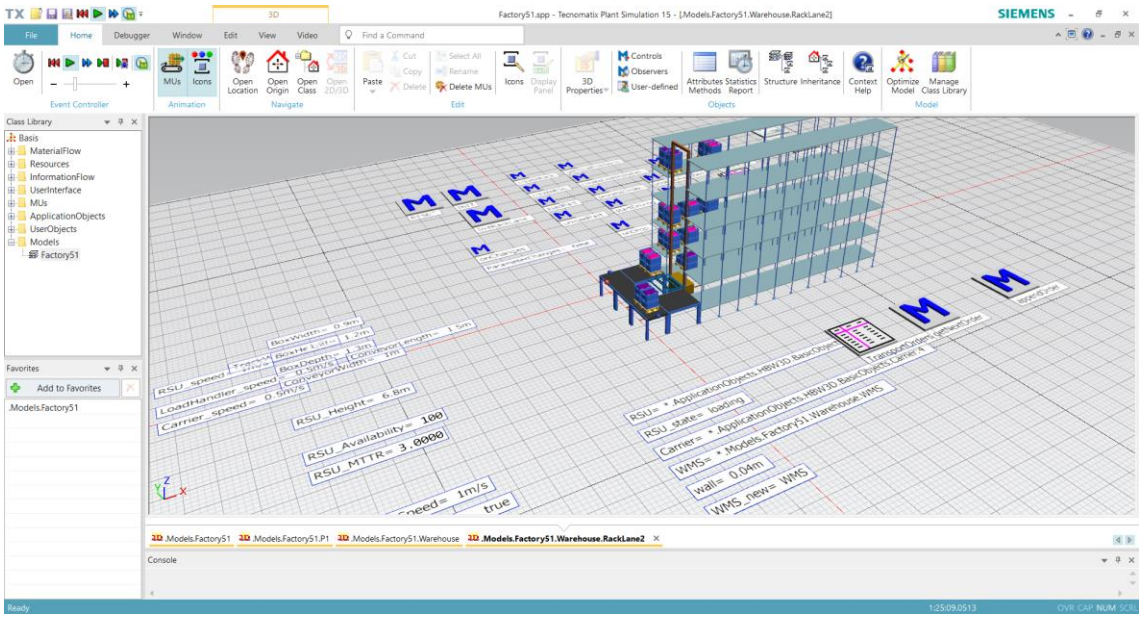

Figura 6: Estantes con elevador automatizado

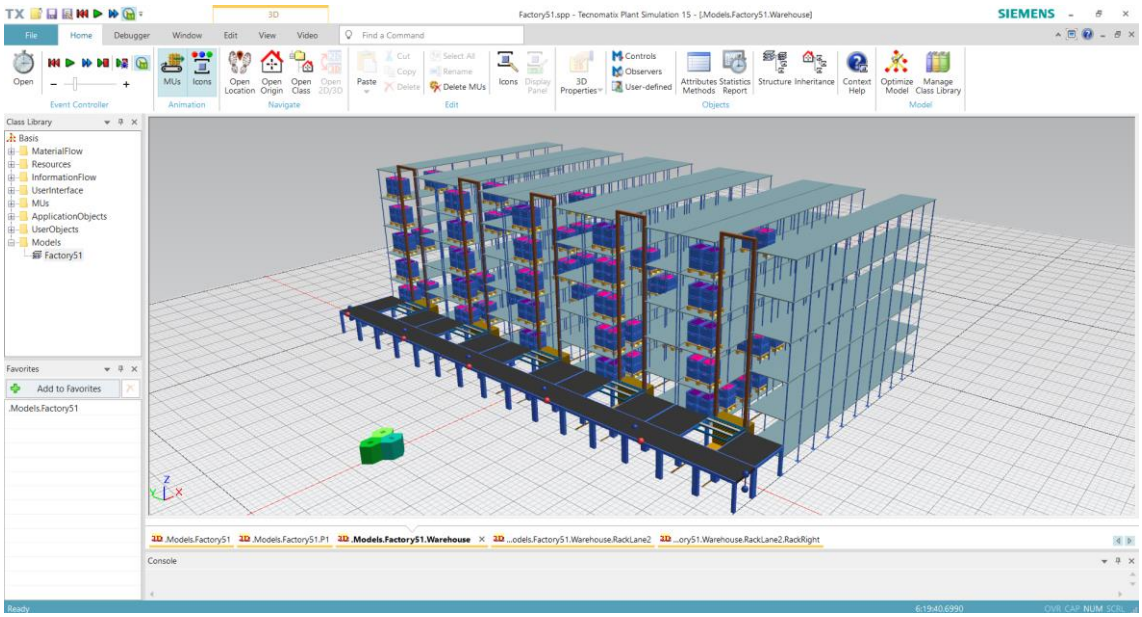

Figura 7: Almacén con todas las líneas de estantes y elevadores

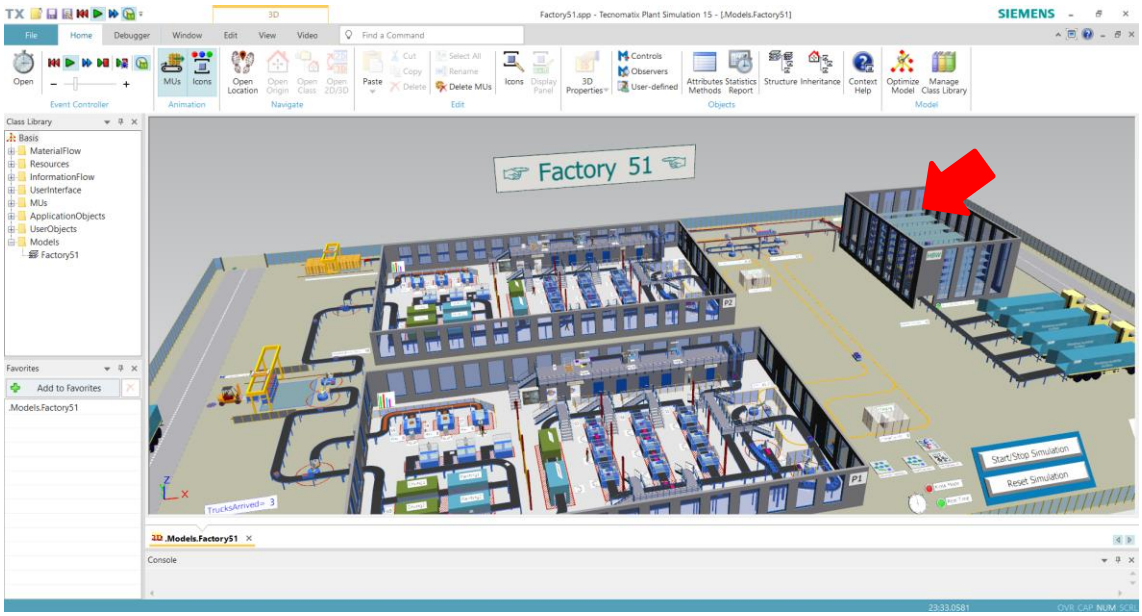

Figura 8: Fábrica completa con el almacén integrado en la línea de producción

Los modelos digitales creados permiten a los usuarios realizar pruebas y trabajar con escenarios hipotéticos, para hacer simulaciones del tipo "¿Qué pasaría si…?", todo ello sin afectar, obviamente, a los sistemas de producción existentes, ya que se todos los experimentos se realizan en el gemelo digital.

Además, Plant Simulation también ofrece datos de las simulaciones en tiempo real que, junto a las herramientas de análisis exhaustivo, permiten obtener estadísticas y gráficos que posibilitan a los usuarios evaluar distintos escenarios de fabricación y tomar decisiones rápidas y fiables en las primeras fases de la planificación de la producción, así como analizar donde se encuentran los cuellos de botella en la línea de producción [20]

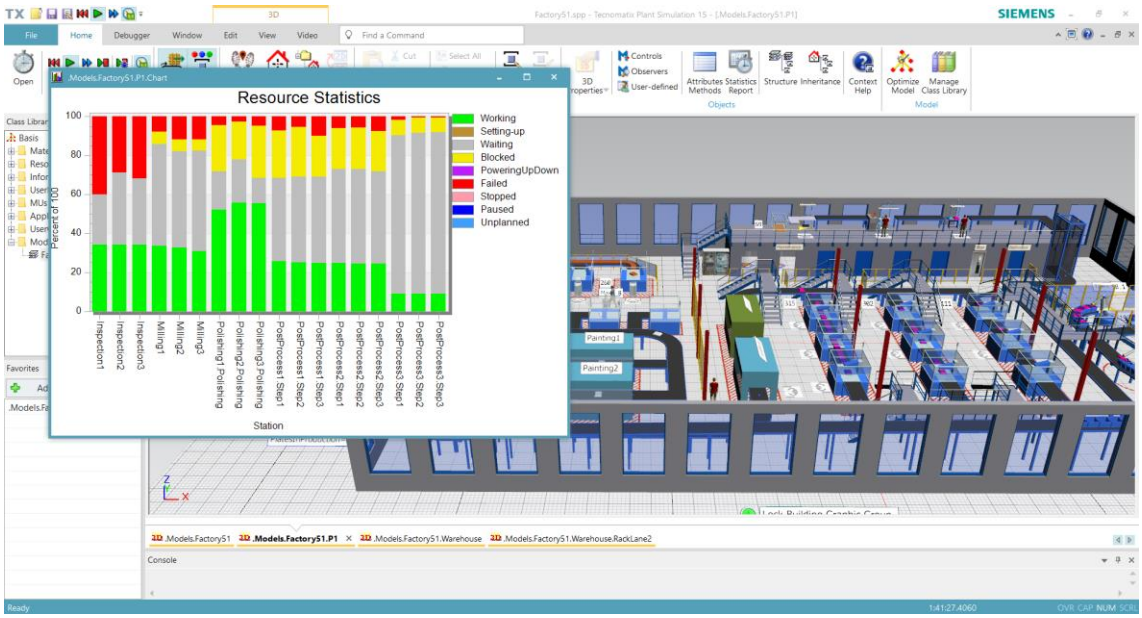

Figura 9: Estadísticas en tiempo real de diversos procesos en la simulación de una fábrica

Por último, este software también admite amplias posibilidades de comunicación con otros softwares de Siemens, con PLCs, usando OPC UA, o con servidores en la nube, usando ODBC.

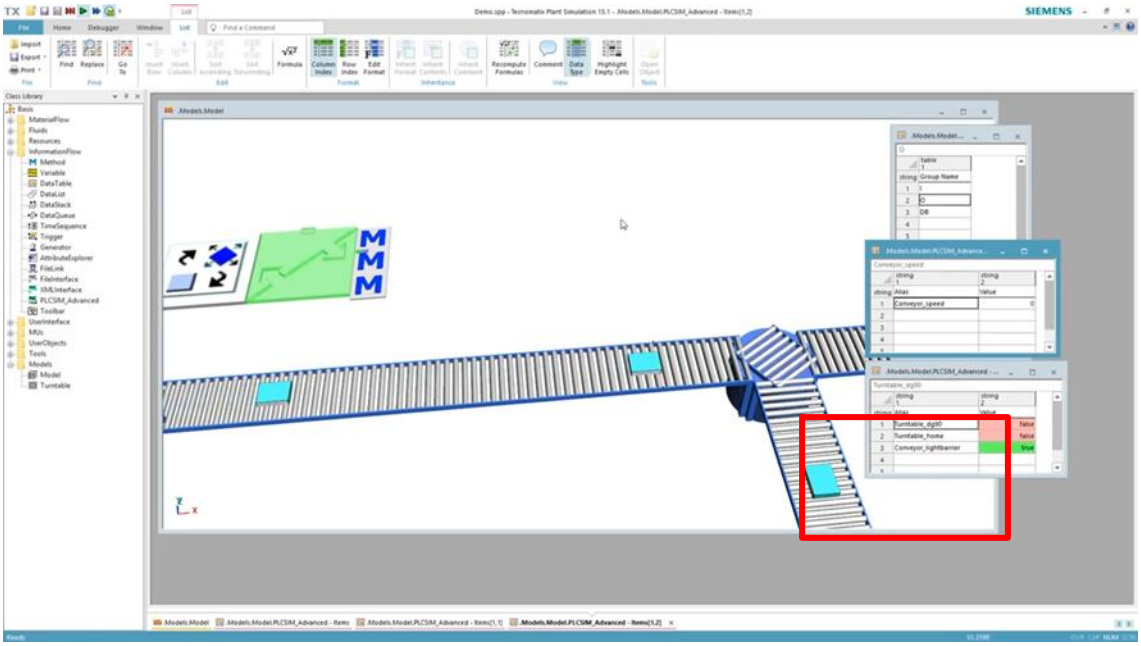

Figura 10: Conexión con PLCSIM Advanced (PLC virtual de Siemens)

2.2. NX MCD (Siemens)

El software NX es una herramienta potente y flexible que dispone de soluciones de diseño, simulación y fabricación. NX integra distintas disciplinas del sector ingenieril, preserva todos los datos y diseños y optimiza el proceso al completo. Da soporte no solo al diseño de productos, también al resto de aspectos del desarrollo de esos productos: simulación, fabricación, mecatrónica, etc [21]

Dentro de la gran variedad de aplicaciones dentro del software NX, la extensión de Mechatronic Concept Design (MCD) ofrece grandes posibilidades para el desarrollo de gemelo digitales. Proporciona un lenguaje común para que tanto ingeniería mecánica, como eléctrica y de automatización puedan trabajar simultáneamente, lo que disminuye los posibles problemas de integración al final del proceso de diseño, ya

que todo se hace desde la misma plataforma.

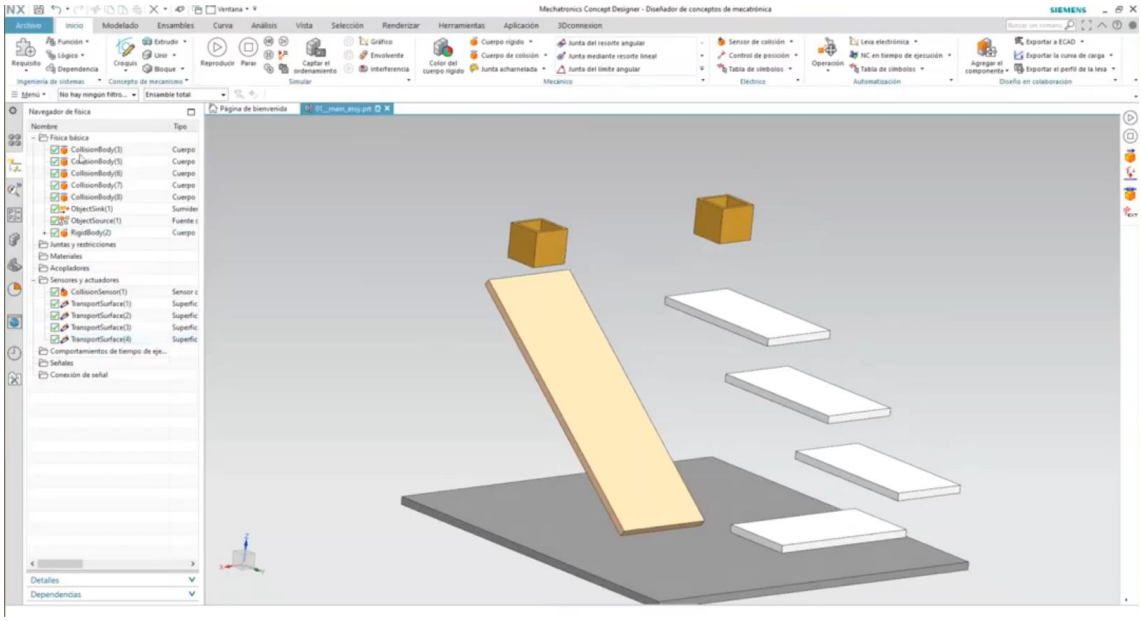

Figura 11: Apartado mecánico, eléctrico y de automatización dentro de la ventana principal MCD [22]

La gran utilidad que tiene esta herramienta es que permite al usuario ir diseñando al mismo tiempo que va dando vida a la simulación del proceso, de forma que se pueden evaluar rápidamente conceptos y problemas de la maquinaria en este entorno virtual. Así mismo, gracias a la física avanzada del software, se pueden ir detectando malfuncionamientos, colisiones, caídas de objetos, etc.

En cuanto al diseño mecánico, se integra a la perfección con los diseños creados en NX, ya que MCD está dentro de la misma plataforma, pero también se puede exportar a otras muchas herramientas de diseño CAD muy empleadas en el mercado, como SolidWorks, CATIA, JT, etc.

En MCD se pueden hacer simulaciones sin necesidad de conexión con un PLC, haciendo la automatización del proceso dentro de la misma aplicación.

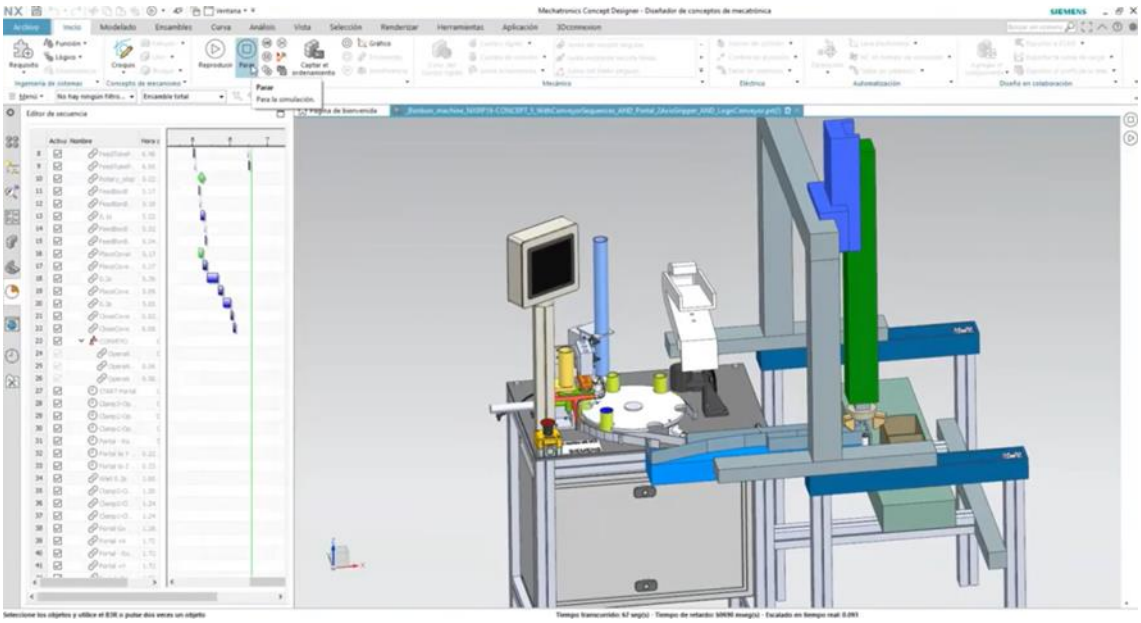

Figura 12: Simulación de un proceso de sellado de botes sin conexión a PLC [22]

Por supuesto, MCD también permite la comunicación con programas externos. Entre estas comunicaciones destacan :

- Comunicación con MATLAB a través del protocolo Modbus sobre TCP/IP
- Conexión con un servidor OPC UA
- Comunicación S7 por medio de PLCSIM Advanced o directamente a través del protocolo PROFINET
- Conexión a una memoria compartida (Shared Memory) para, p. ej., SIMIT
- Conexiones TCP/UDP

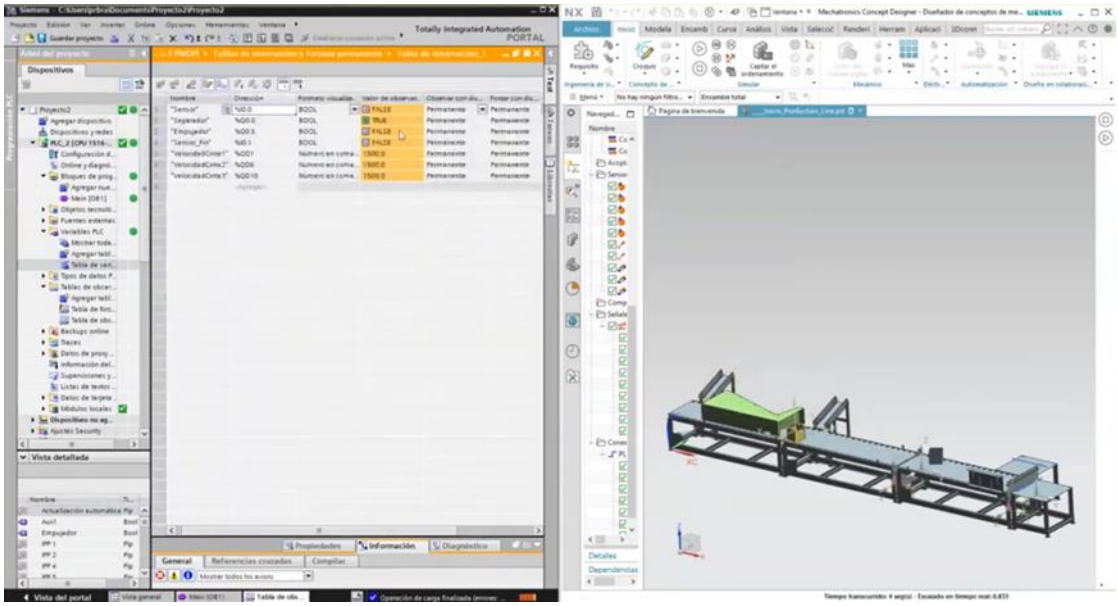

Figura 13: Simulación de una línea de cajas con conexión al software Tia Portal [22]

2.3. Visual Components

Este software de simulación 3D muy visual está diseñado para la producción a nivel profesional y construido en una plataforma potente y flexible. En una única plataforma, de forma similar a la extensión de NX que se presentó en el anterior apartado, MCD, se puede diseñar, simular y emular los proyectos de forma muy visual y tiene presencia en una gran cantidad de industrias, tales como el automóvil, electrónica, automatización, logística, maquinaria, … [22]

El objetivo primordial de esta herramienta es la simulación de líneas de producción o incluso pequeñas fábricas, que pueden incluir robots, vehículos autoguiados, cintas transportadoras, entre otros muchos elementos, y la obtención de datos en tiempo real para su posterior análisis. Entre sus múltiples características, destacan:

- eCatalog: Librería de modelos virtuales que posee más de 2500 componentes, tales como robots, AGVs, cintas transportadoras, máquinas, etc. de las principales marcas de automatización (ABB, Staübli, KUKA, Toshiba, Schneider Electric, …). Esta librería se va actualizando con nuevos elementos, y se puede acceder con facilidad desde la ventana principal de la aplicación. Los componentes se encuentran ordenados por tipo o por fabricante, y también se pueden buscar directamente usando la barra de búsqueda. Además, también es posible personalizar los comportamientos y propiedades de los componentes del catálogo para que el usuario pueda crear una biblioteca personalizada.

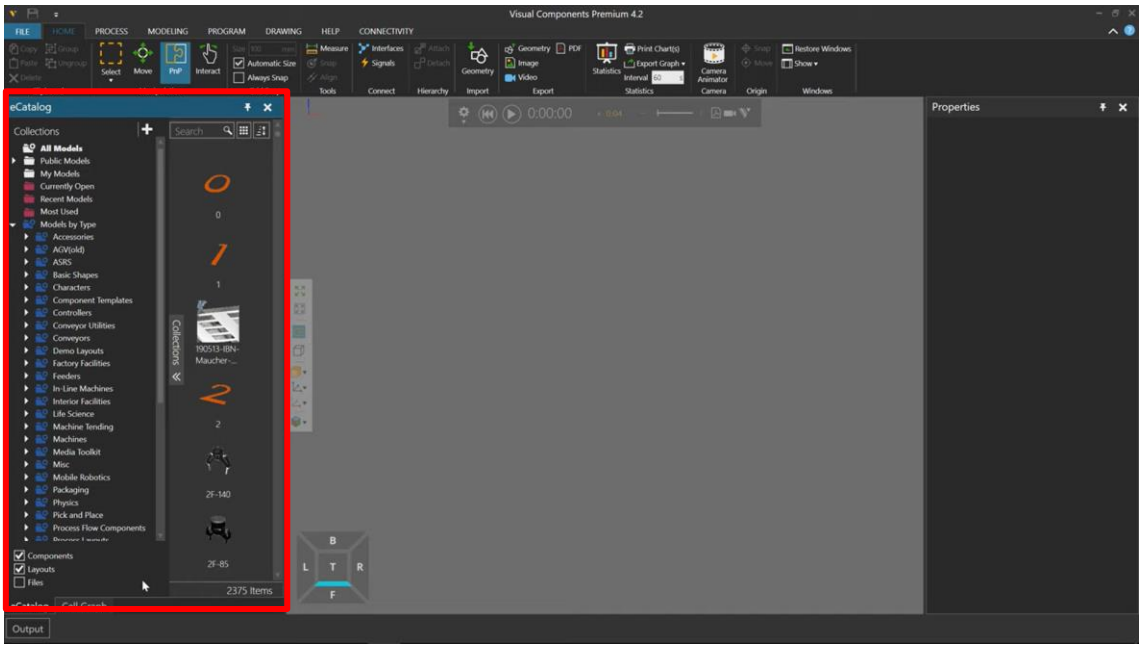

Figura 14: Librería de componentes [24]

- Importación y exportación de archivos: Posibilidad de importar archivos CAD, directamente al mundo 3D del software, y con compatibilidad de la mayoría de formatos y extensiones de archivo de los softwares de diseño más importantes: Autodesk (.dwg, .dxf), CATIA (.3dxml), Revit (.rvt), Solid Edge (.asm, .par, .pwd), Siemens NX (.u3d), JT (.jt), etc. Así mismo, también se pueden exportar los archivos creados con una amplia variedad de extensiones [23]
- Estadísticas: Se pueden crear gráficos de cualquier tipo y obtener datos estadísticos de la simulación en tiempo real. Los datos estadísticos pueden ir desde el estado de un componente

  hasta el número de piezas que se han producido en un tiempo, o la tasa de producción. De igual modo, al poder disponer de estadísticas, se pueden realizar simulaciones de tipo ¿Qué pasaría si…?, adelantando dichas simulaciones y analizando los datos obtenidos.
- Conectividad: Varias opciones de conexión a PLC, emuladores, etc. a través de OPC UA, Siemens S7, SIMIT y WinMOD Net. También existe la posibilidad de conexión con controladores Universal Robot y Staübli CS8 y CS9 para la programación de robots.

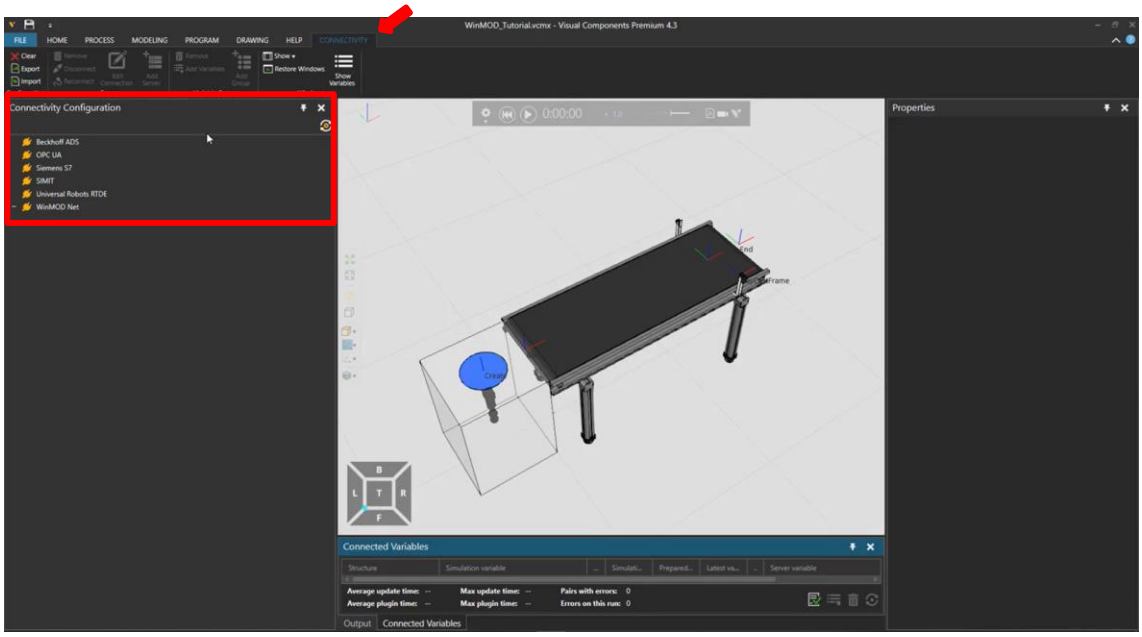

Figura 15: Posibles conexiones de Visual Components [24]

- Plataforma abierta: Visual Components posee dentro de su aplicación una API (siglas en inglés que se traducen como interfaz de programación de aplicaciones) de Python que permite no solo hacer personalizaciones de la propia aplicación, sino también programar robots o diferentes acciones dentro del proceso de simulación, permite generar otros KPIs (siglas en inglés que se traducen como indicadores clave de desempeño) de interés. Además, también existe una API de .NET, gracias a la cual se pueden crear pluggins, pasar mensajes, editar componentes creados durante la simulación, y todo desde un editor de código externo, como por ejemplo Visual Studio Code.

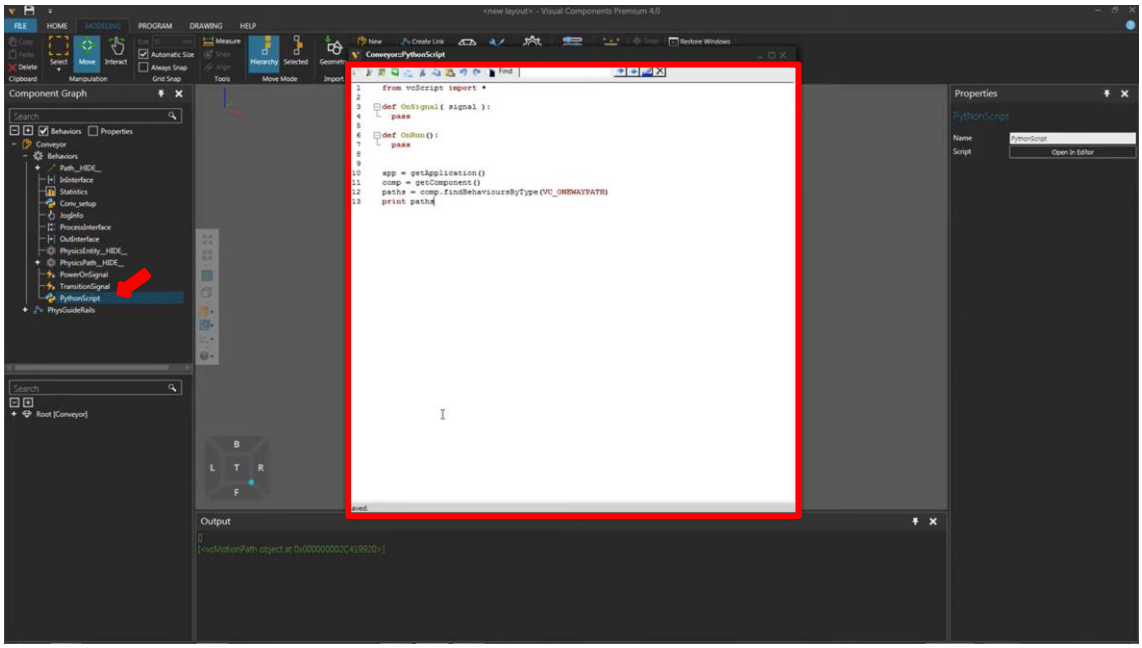

Figura 16: Script de Python dentro de la ventana principal de Visual Components [24]

- Motor físico: Se pueden añadir parámetros físicos a las simulaciones ya que el software incluye el motor físico NVIDIA Physx. Por tanto, se puede dotar de físicas a los componentes de la simulación.

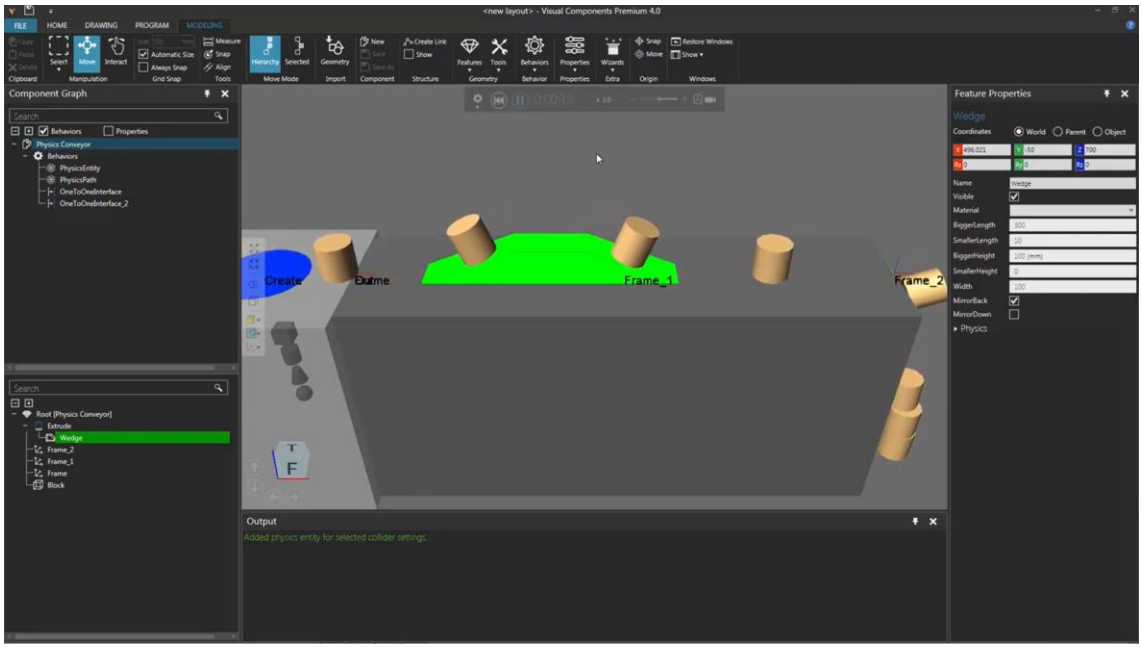

Figura 17: Simulación con físicas [24]

- Realidad Virtual: Conexión con gafas de realidad virtual para simulaciones en un entorno virtual interactiva con diferentes elementos de diseño, todo en tiempo real

3 DESCRIPCIÓN DE UNITY 3D

Unity es un software en un principio diseñado para la creación y desarrollo de videojuegos. Combina un entorno gráfico con programación en lenguaje C#. La rutina básica de desarrollo consiste en la creación de objetos desde una barra de herramientas gráficas y, posteriormente, asignarles programas desarrollados previamente que describan el comportamiento deseado para dichos objetos.

3.1 ¿Por qué Unity 3D?

Se ha visto en el apartado anterior varias herramientas industriales y se ha hecho una evaluación de las mismas, analizando sus características principales. En dichas herramientas, como en tantas otras que no se han estudiado, se pretenden obtener modelos sencillos de procesos industriales los cuales se puedan simular, adaptándose lo máximo posible a la realidad, con la correspondiente simplificación del gemelo digital en sí que esto conlleva.

Esto hace que, si se quiere hacer un gemelo digital haciendo uso de los softwares anteriores, este no pueda ser todo lo complejo que el usuario quiera, no solo porque la mayoría de plataformas no son de código abierto, o haya que adquirir licencias para su uso, sino porque son software que no llegan a bajo nivel, no tienen un motor físico capaz de obtener datos dinámicos, físicos, etc. de la simulación realizada. Se pueden adquirir datos de volumen de producción de un determinado proceso industrial, por ejemplo, pero no qué porcentaje de energía se esta empleando en el motor que actúa en la cinta transportadora del proceso, o la presión que pierde el pistón neumático de un elevador en cada accionamiento del mismo.

Por tanto, también se ha realizado el estudio de este software de simulación, Unity 3D, el cual, al estar pensado para el diseño de videojuegos, intenta imitar la realidad de una manera más exacta que las herramientas

evaluadas anteriormente. Además, es un software gratuito y de código abierto, lo que facilita su acceso y su aprendizaje, a parte de que minimiza esas limitaciones que tienen otras herramientas industriales.

Antes de terminar este capítulo, se ha hecho una tabla comparativa con algunas de las características de los diferentes softwares evaluados, incluyendo también Unity. En ella no se muestran todas las propiedades y características de las herramientas evaluadas, sino solo las más relevantes que un usuario inexperto deba tener en cuenta:

	Tecnomatix Plant Simulation	NX MCD	Visual Components	Unity
Datos en tiempo real	Si	Si	Si	Si
Análisis estadístico	Si	-	Si	Si
Datos a bajo nivel	-	-	-	Si
Librerías de componentes	Solo componentes genéricos	-	Gran variedad de marcas comerciales	-
Diseño 3D	Si	Si	Si	Si
Modelado 3D	-	-	-	Si
Detección de colisiones	-	Si	Si	Si
Motor físico	-	Si, propio	Si, NVIDIA Physx	Si
Realidad Virtual	-	-	Si	Si
Importación de archivos	-	Pocos tipos	Muchos tipos	Muchos tipos
Exportación de archivos	Si	Si	Si	Si
Conectividad	OPC UA	MATLAB, OPC UA, Siemens S7, TCP/UDP y Memorias compartidas	OPC UA, Siemens S7, SIMIT y WinMOD	Si
Plataforma abierta	-	-	API Python y .NET	C#
Fácil aprendizaje	Si	-	Si	-
Ayuda/Tutoriales en la red	Si	-	Si	Si
Precio	Alto	Alto	Medio	Gratis[1]

Tabla 1: Comparativa características softwares evaluados

Es importante resaltar que todas las características que poseen las diferentes herramientas evaluadas, Unity también puede implementarlas, pero hay que programar scripts para, por ejemplo, hacer un análisis estadístico del volumen de producción de una planta.

El ejemplo que se muestra en el capítulo siguiente se ha realizado con Unity 3D. A ello a facilitado también la

[1] Ingresos o fondos inferiores a USD 100 mil en los últimos 12 meses [31]

lectura de otros trabajos realizados previamente, como por ejemplo una simulación virtual de un entorno industrial que imitaba el transporte de cajas por unas cintas transportadoras usando un puente grúa [17]

3.2 Introducción a Unity 3D

La versión empleada de Unity ha sido la 2020.1.17f1, y se ha ultilizado la herramienta de administración Unity Hub (Ver Figura 18), para instalar la mencionada versión de Unity y crear el nuevo proyecto. La descarga de Unity Hub se ha realizado desde la página web de Unity [24].

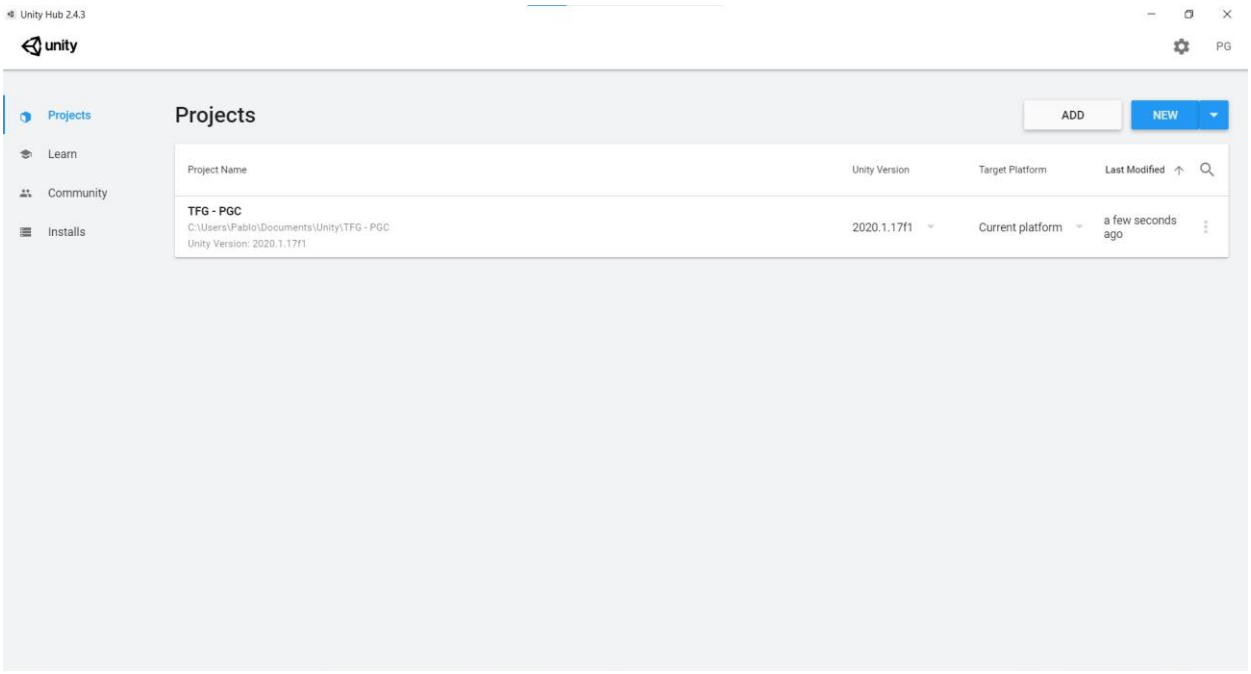

Figura 18: Pantalla projectos de Unity Hub

Al abrir el proyecto de Unity, aparece el editor. En la parte superior de la ventana hay una serie de pestañas que permiten abrir archivos, crear objetos, editarlos, añadir componentes, personalizar las ventanas, ayuda, etc. En la Figura 19 se pueden ver las diferentes ventanas en las que se divide la interfaz de usuario y que aparecen por defecto al iniciar por primera vez el software. Esta disposición se puede cambiar, haciendo que sean visibles más o menos ventanas al mismo tiempo en la pantalla.

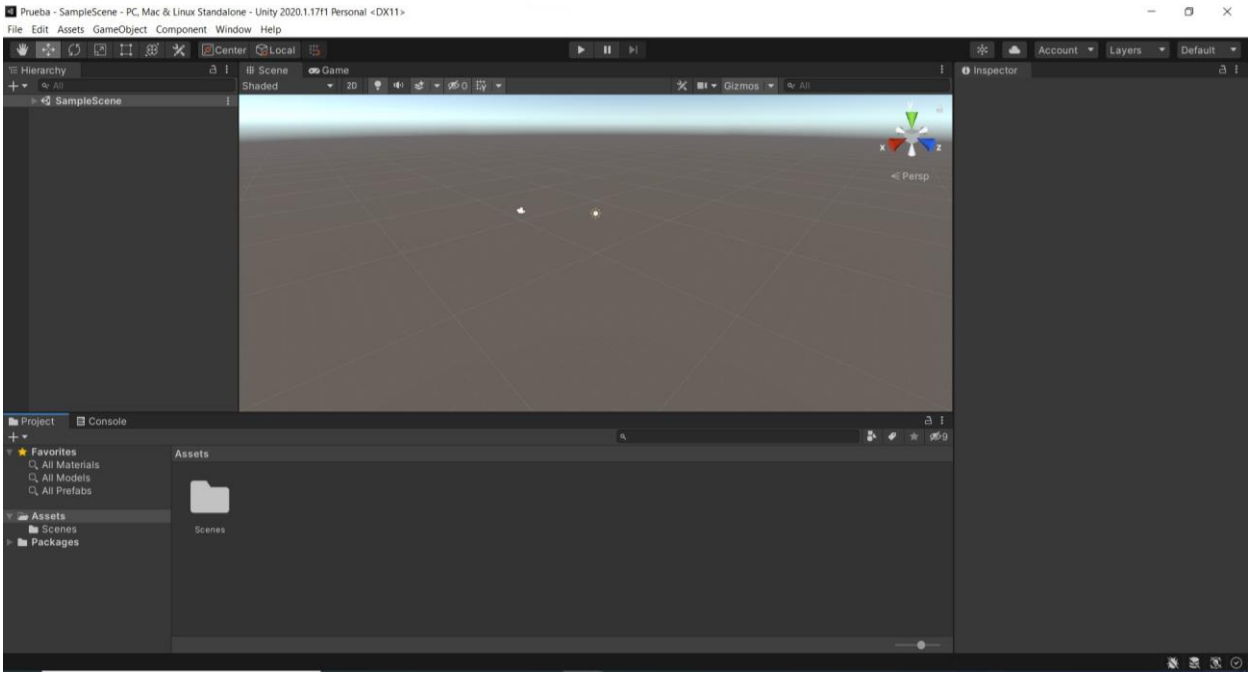

Figura 19: Interfaz de usuario de Unity

Las vistas principales dentro de la interfaz son:

- Explorador

Muestra todos los elementos del proyecto, y gracias a él, se puede ordenar de forma sencilla la aplicación. El usuario puede crear carpetas donde se encuentran las escenas, los códigos de programación, las imágenes, las texturas de los elementos, los objetos importados por el usuario, etc.

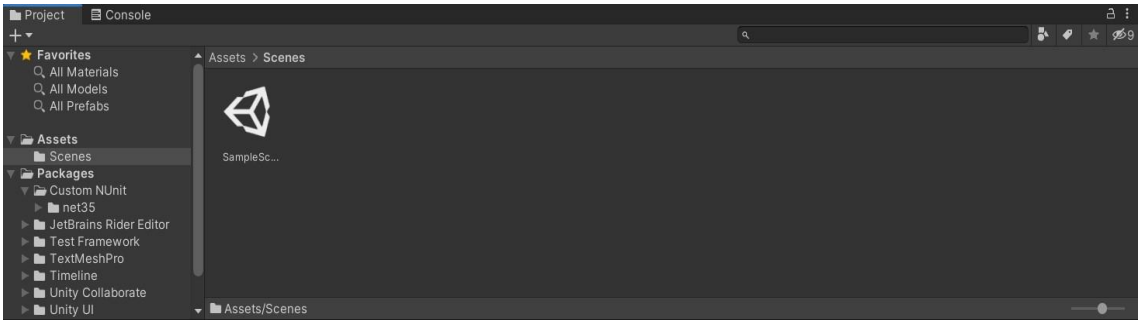

Figura 20: Interfaz de la ventana del explorador

- Inspector

Muestra y define cada una de las propiedades de todos los elementos del proyecto. En la Figura 21 se muestran las propiedades de un cubo, que son la posición en el espacio, el renderizador de malla (que toma la geometría del filtro de malla, en este caso la de un cubo, y la renderiza en la posición defenida anteriormente [25], haciendo que dicho cubo sea visible), y el collider (que permite al cubo detectar a otros objetos y tener colisiones). En esta ventana se le pueden añadir al elemento seleccionado todas las propiedades que el usuario quiera darle a dicho elemento, desde un comportamiento concreto mediante un scrip, o un material concreto, o alguna propiedad física, etc.

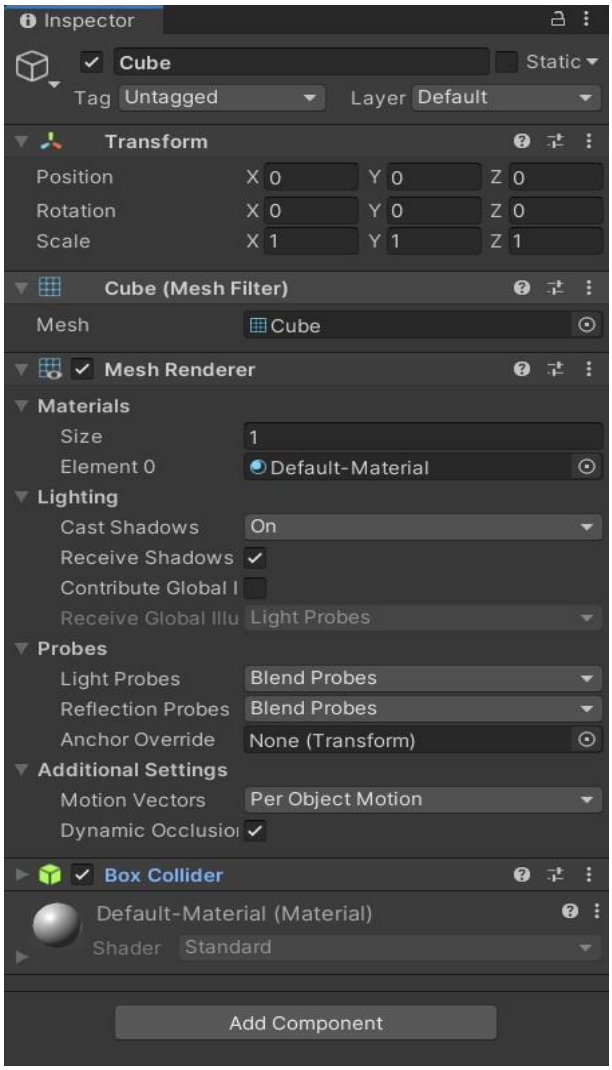

Figura 21: Interfaz de la ventana del inspector

- Jerarquía

Muestra todos los elementos que forman las escena de manera visual y rápida, formando una lista con jerarquías, en la que se puede ver, al seleccionar un objeto, los elementos por los que está formando dicho objeto. Del mismo modo, al pulsar en cada uno de los objetos, aparecerán sus propiedades en la ventana del inspector

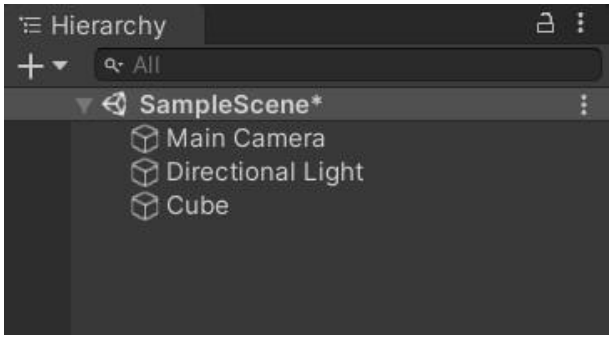

Figura 22: Interfaz de la ventana de jerarquía

- Escena

Muestra el diseño del proyecto completo que se está realizando o una sección de este. En esta pantalla, el usuario puede moverse por todo el entorno e interactuar con cada uno de los objetos que se encuentren en la

escena, modificando su posición, su orientación, etc. Como se puede en la Figura 22 y en la Figura 23, los únicos objetos que se encuentran en la escena son el cubo que el usuario ha creado, la luz direccional y la cámara principal, ambos creados automáticamente al abrir un nuevo proyecto

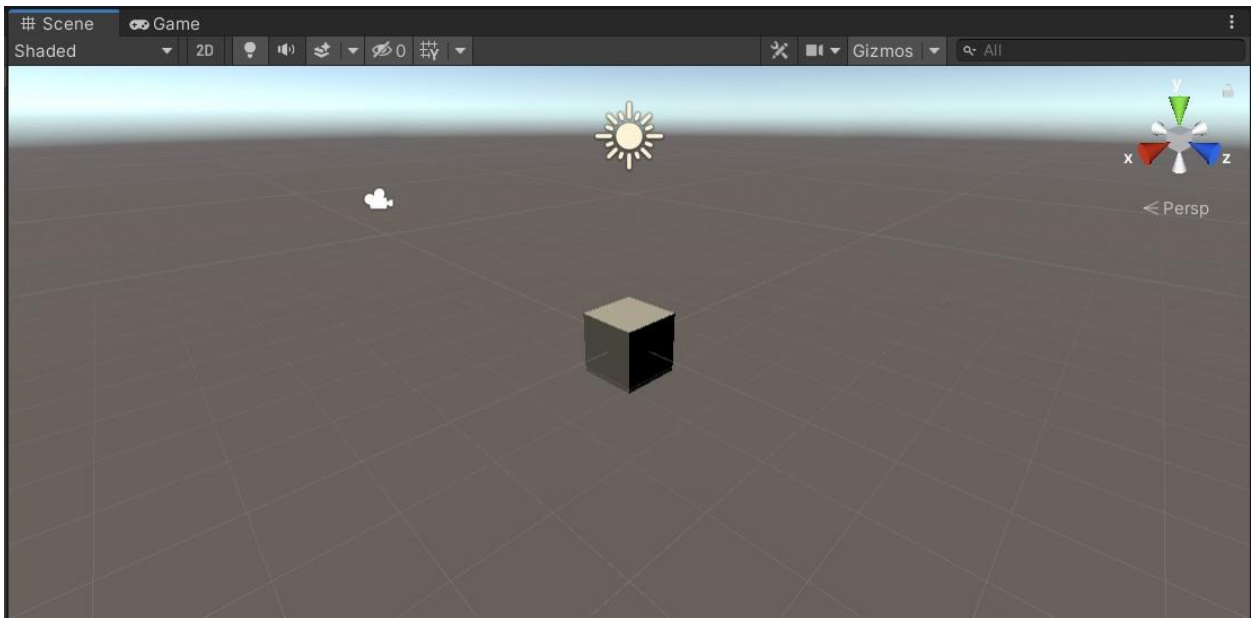

Figura 23: Interfaz de la ventana de escena

- Juego

Muestra la vista del proyecto durante su ejecución. Dicha vista depende de la posición y orientación de la cámara, la cual se puede cambiar en la ventana de escena. Esta ventana de juego no se muestra por defecto junto con las otras hasta que no se ejecuta el proyecto, momento en el que se deja de ver la ventana de escena y se comienza a ver esta nueva pantalla.

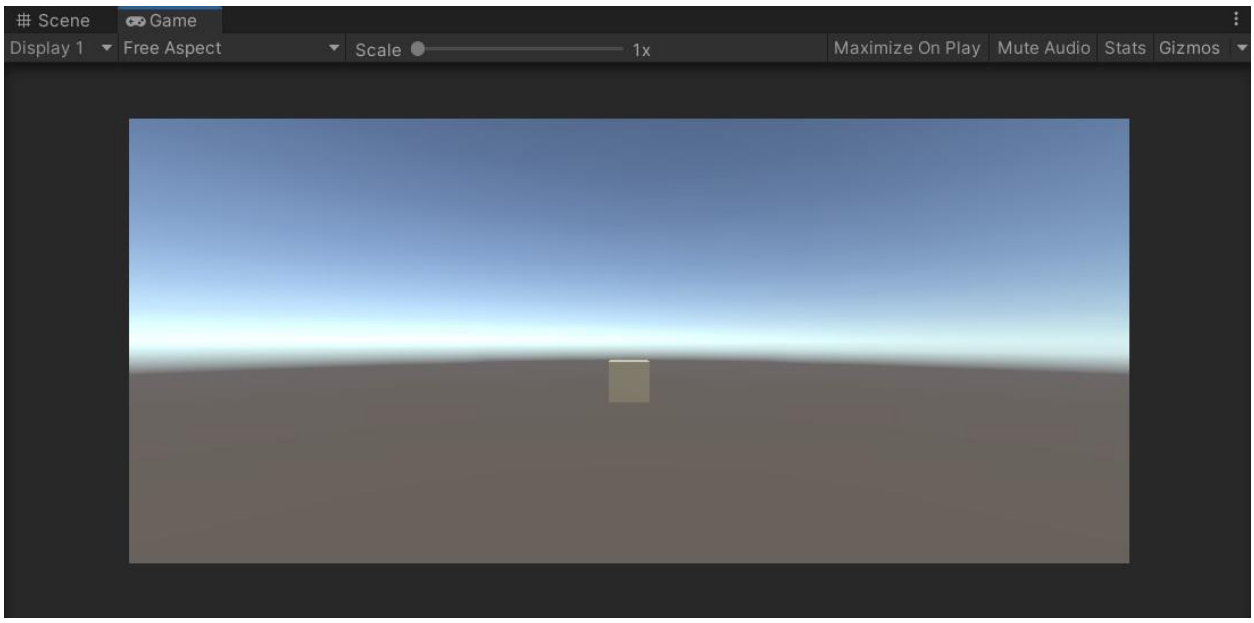

Figura 24: Interfaz de la ventana de juego

Aunque no se ha mencionado, también existe otra ventana, la consola, que muestra los mensajes que envía los scrip al ejecutarse el proyecto. Se puede ver minimizada en la Figura 20, aunque como ya se ha dicho, también se puede modificar su posición en la interfaz del usuario.

4 DESARROLLO DEL GEMELO DIGITAL DE UN ALIMENTADOR DE PIEZAS

A modo de ejemplo, se ha realizado un modelo sencillo de gemelo digital de un alimentador de piezas que forma parte de una célula de fabricación flexible. Una célula de fabricación flexible no es más que un grupo de estaciones de trabajo, como por ejemplo una máquina taladradora o una máquina de verificación de calidad. Estas estaciones se encuentran automatizadas e interconectadas entre sí a través de un sistema de transporte de materiales, como por ejemplo cintas transportadoras, el cual también está automatizado. El objetivo de estas células es la producción de una gran cantidad de productos diversos.

Figura 25: Célula de fabricación flexible[2]

[2] Esta imagen y las dos siguientes han sido cedidas por el profesor Luis Fernando Castaño Castaño

Este alimentador de piezas modelado es un elemento más de una célula de fabricación flexible que se encuentra en el laboratorio del Departamento de Ingeniería de Sistemas y Automática de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla. Dicha célula, cuya función es eminentemente educativa, consiste en un sistema de montaje de piezas sobre palets a lo largo de unas cintas transportadoras dispuestas formando un rectángulo, como se puede en la Figura 25 .De esta célula de fabricación, se ha modelado solamente el alimentador de piezas negras y blancas que se van poniendo en los diferentes palets.

El alimentador, como se puede ver en la Figura 26, consta principalmente de dos primas rectangulares metálicos, apoyados en base también metálica donde se sitúan los dos motores del alimentador. Estos motores son los encargados de mover una pequeña plataforma que se mueve por el interior de los prismas y que es la que hace que las piezas aparezcan en la parte de arriba del alimentador. Cada motor se conecta a la plataforma de uno de los prismas por medio de una correa. Además, en la parte de arriba se tiene otra base sobre la que se apoyan los sensores encargados de detectar las piezas a extraer o a almacenar en el alimentador. Los sensores son capacitivos y están situados en ambas salidas de piezas.

Figura 26: Alimentador de piezas

El funcionamiento del alimentador es muy simple. Se pueden extraer piezas, para lo cual la plataforma que sostiene dichas piezas dentro del alimentador subirá, dejando la pieza detectada por los sensores, fuera de la estructura (Ver Figura 27), a la espera de ser retirada por un brazo robótico o por un operario; o se pueden almacenar piezas, dejando la pieza en la misma posición que para la extracción, de forma que, al detectarla el sensor, la plataforma descenderá hasta almacenar la pieza

Figura 27: Almacenamiento y extracción de piezas del alimentador de piezas

4.1 Creación del modelo en 3D

Para crear el modelo 3D del alimentador de piezas, se ha empleado Cinema 4D®, un software de modelado, animación, simulación y renderizado 3D, el cual permite hacer diseños más accesibles y eficientes gracias su conjunto de herramientas rápido, flexible y potente [26]. Existen muchos softwares de modelado 3D, pero se ha escogido este por su potencia y fácil aprendizaje.

El esquema de diseño que se a seguido a sido:

1. Diseño de la estructura del alimentador

En primer lugar, se modelo el diseño del prisma rectangular que hace de alimentador de piezas. Para obtener el hueco, se crearon dos cubos, uno más pequeño que el otro, y se modificó su geometría para adaptarla a la del modelo físico, creandose así dos primas, el más pequeño dentro del más grande. Luego, se hizo uso de la función boole, restando uno al otro y obteniéndose así el prisma hueco. Finalmente, se elimino uno de los planos laterales en los que estaba dividido el prisma y se le añadió otro plano para cerrar el lateral superior, el cual no tiene la abertura. El resultados se muestra en la Figura 28.

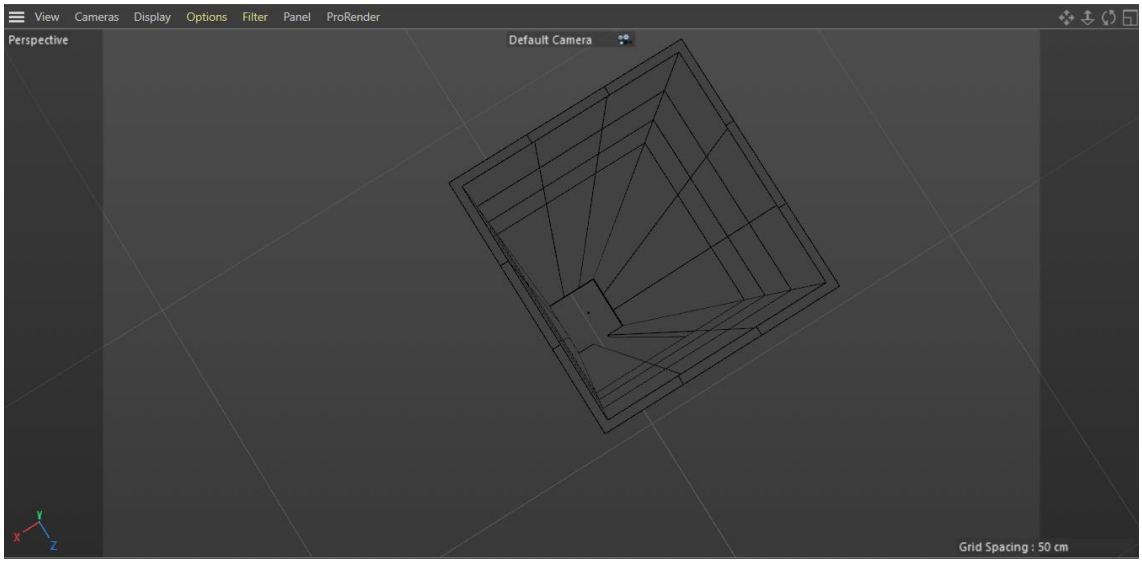

Figura 28: Vista superior del prisma hueco con líneas ocultas

En el siguiente paso, se le aplica al prisma creado una simetría, para obtener así los dos alimentadores, como se muestra en la Figura 29. Por último, para terminar la estructura del alimentador, se crea una base, modificando un cubo para obtener un prisma rectangular, como se ve en la Figura 30.

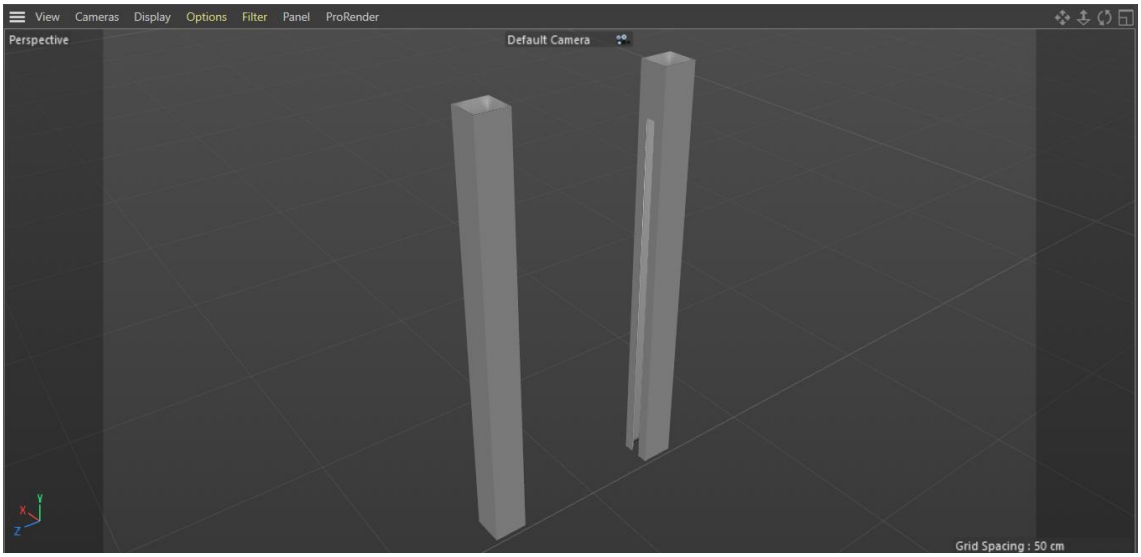

Figura 29: Prismas huecos con abertura para la plataforma de piezas

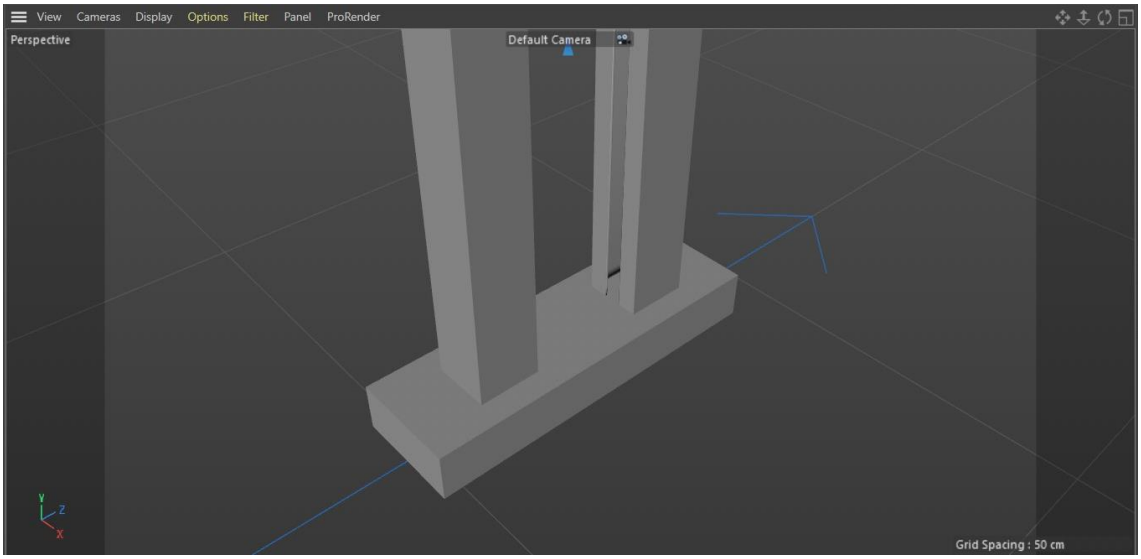

Figura 30: Alimentadores con base

2. Diseño de la plataforma de piezas

Para la plataforma de piezas, se ha tomado un cubo, al cual se le han hecho tres divisiones en cada uno de sus planos. Después, se han metido hacia adentro de la pieza las divisiones centrales, mientras que la división de otro de los laterales también a acortado su tamaño, y la división del otro lateral se aumenta hasta tomar el tamaño del hueco de los alimentadores creados, quedando como resultado el mostrado en la Figura 31

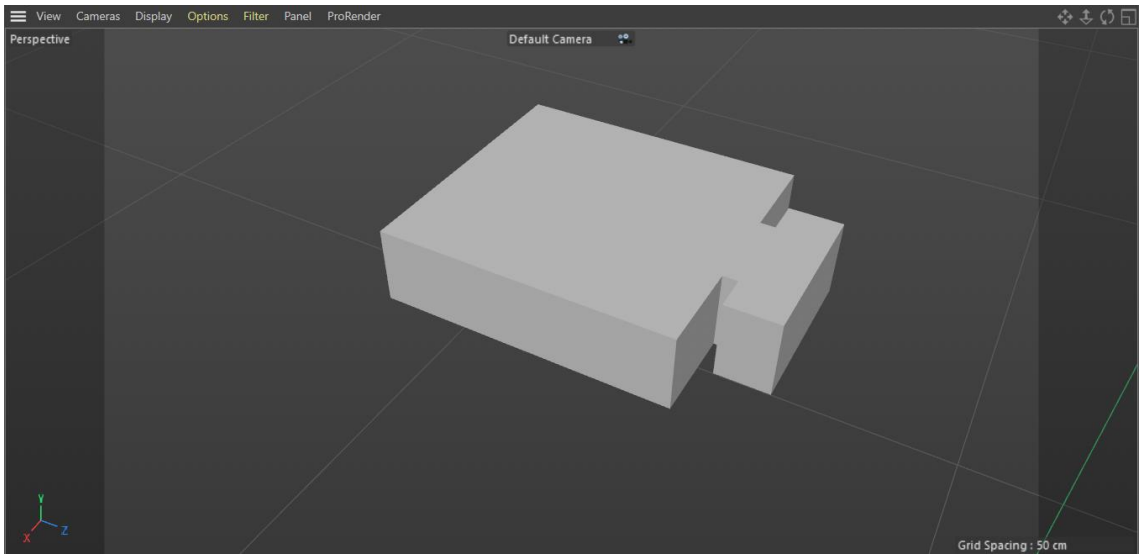

Figura 31: Base de la plataforma de piezas

Por último, se ha añadido un cubo y se ha modificado para que coincidad con la base de la plataforma de piezas, dejando libre la pestaña y el saliente. Además, se ha aumentado su tamaño en el eje Y, para que la plataforma sea más grande. Por último, al igual que a los alimentadores, se ha aplicado simetría para generar la otra plataforma, como se ve en la Figura 32.

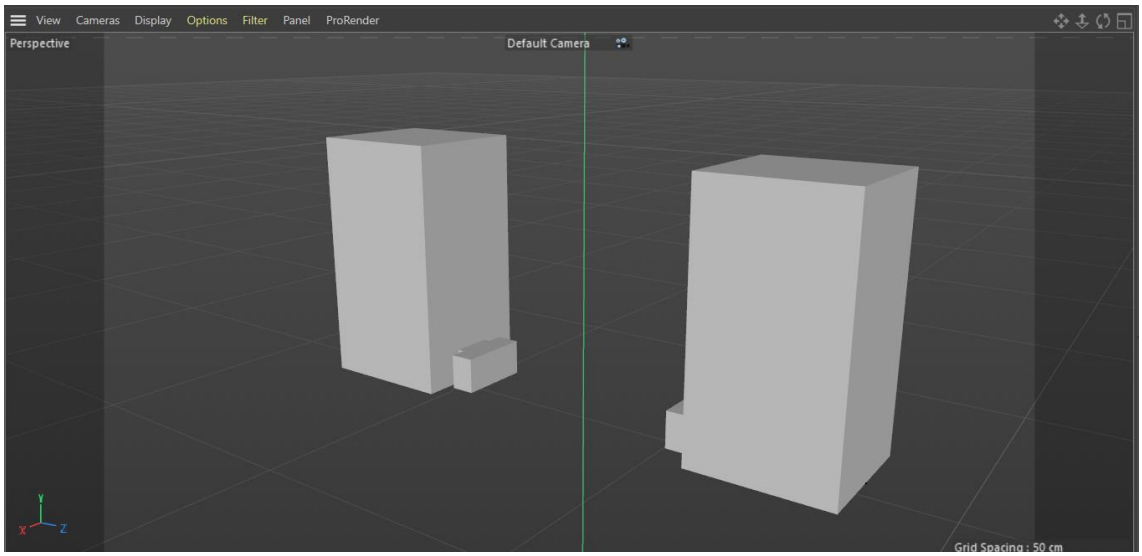

Figura 32: Plataformas de piezas

3. Diseño de las piezas

El diseño de las piezas ha sido muy sencillo, ya que ha únicamente se ha creado un cilindro, con 64 segmentos de rotación, para que no se vea ninguna arista en ninguna parte del borde de la pieza, y se ha modificado su tamaño hasta que la pieza cupiese en el hueco del alimentador. La altura de la pieza es de 1.6 cm.

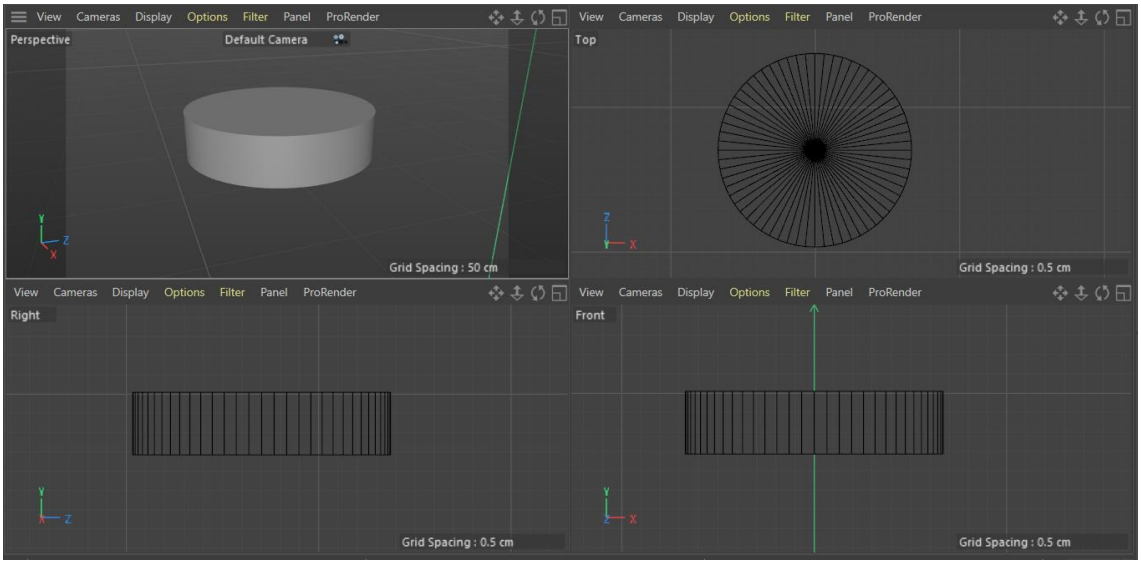

Figura 33: Múltiples vistas del diseño de la pieza

4. Diseño de los sensores y de la plataforma de sensores

La plataforma de los sensores se ha hecho igual que la del alimentador, pero con los huecos de ambos prismas. Para ello, se ha creado dos pequeños cubos, se han modificado su tamaño para que coincida con el hueco de los prismas y, mediante la función boole, se han obtenido ambos huecos. La plataforma se puede ver en la Figura 34.

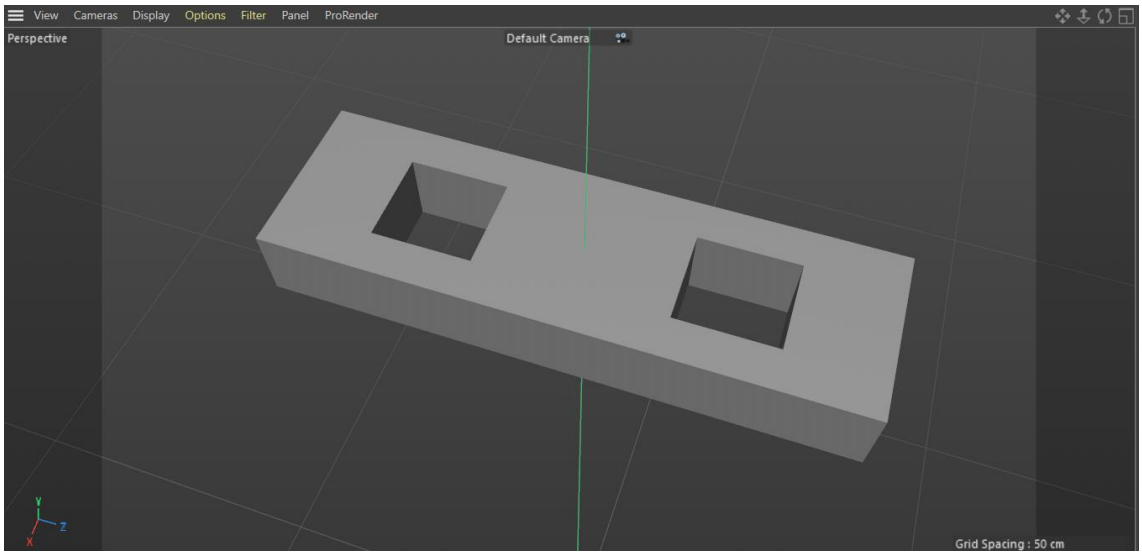

Figura 34: Plataforma de sensores con huecos para los alimentadores

En cuanto al diseño de los sensores, se han tomado un cilindro, al cual se le han suavizado los bordes y cuyos segmentos de rotación se han modificado a 64; dos cilindros más pequeños, cuyo número de segmentos de rotación se ha disminuido a 6, y que se ha "aplastado", para simular el aspecto de una tuerca; y por último, el soporte del sensor, que se ha obtenido modificando un cubo normal. Con todo ello se ha obtenido el sensor de la Figura 35, cuyas líneas negras en el cilíndro del sensor se deben a que también se le ha aplicado la función subdvisión de superficies, para que se vea más curva toda la superficie del cilíndro del sensor.

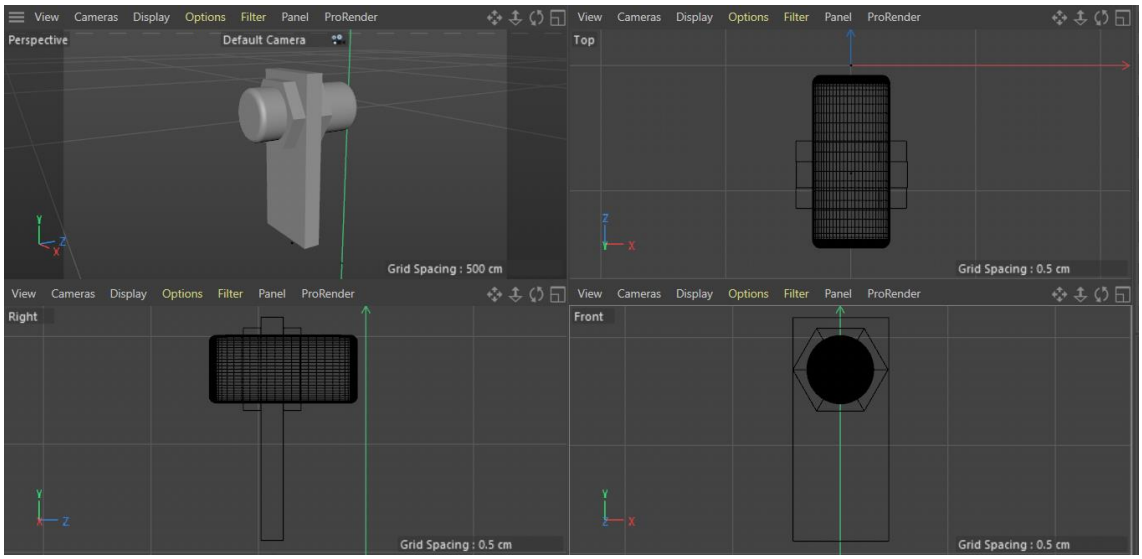

Figura 35: Múltiples vistas del sensor capacitivo

Por último, se ha vuelto a aplicar simetría para obtener el sensor opuesto, y otra nueva simetría para tener los dos conjuntos de sensores, uno para cada salida del alimentador

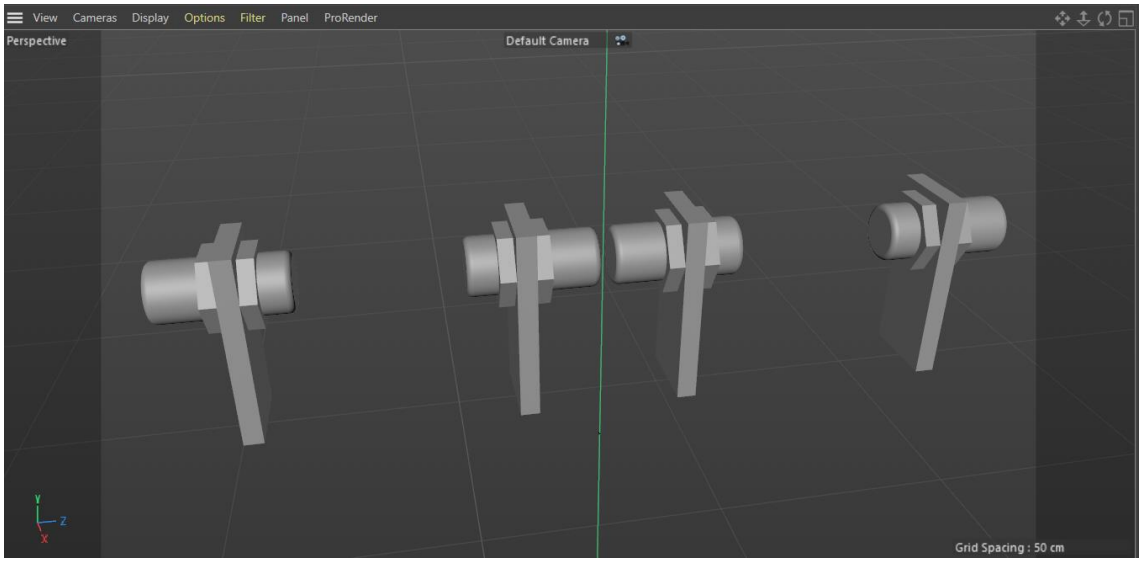

Figura 36: Sensores capacitivos

5. Adición de texturas

Para terminar, se han añadido diferentes texturas a todos los componentes, a fin de que se puedan distinguir mejor entre sí y se tenga la impresiónde de que está construido con un material en concreto. Desde la Figura 37 hasta la Figura 40, se muestra cada componente diseñado con el material elegido. Los materiales que se han tratado de emular son: plástico, de diferentes colores, para las fichas, la plataforma de cada alimentador, la superficie externa del sensor y las tuercas que sujetan este al soporte; metal para la estructura del alimentador; y una aleación de acero para la base de los sensores y sus soportes.

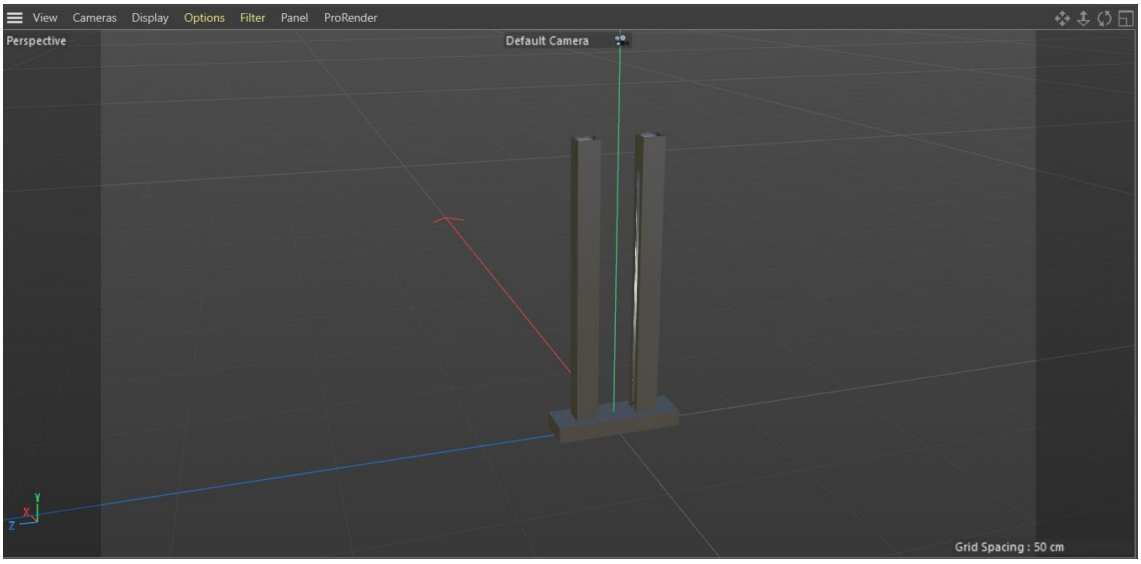

Figura 37: Estructura del alimentador con texturas

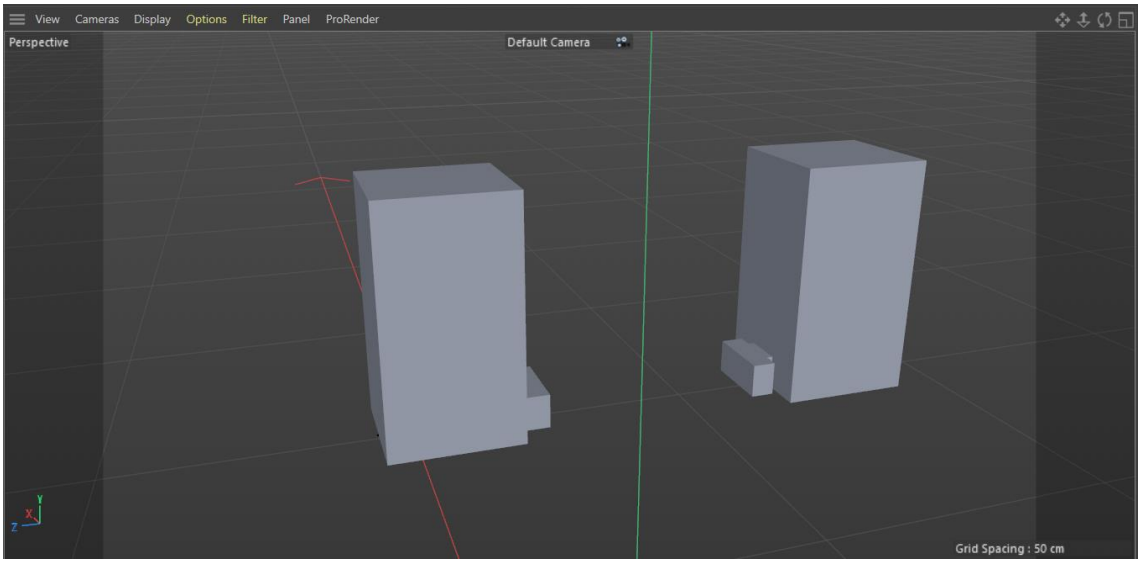

Figura 38: Plataformas de piezas con texturas

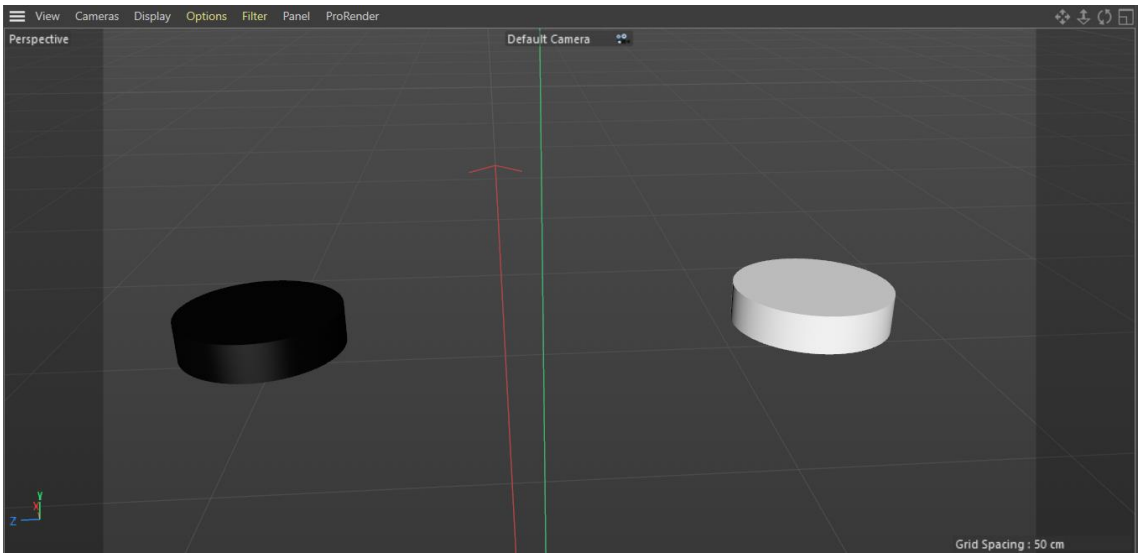

Figura 39: Piezas blanca y negra con texturas

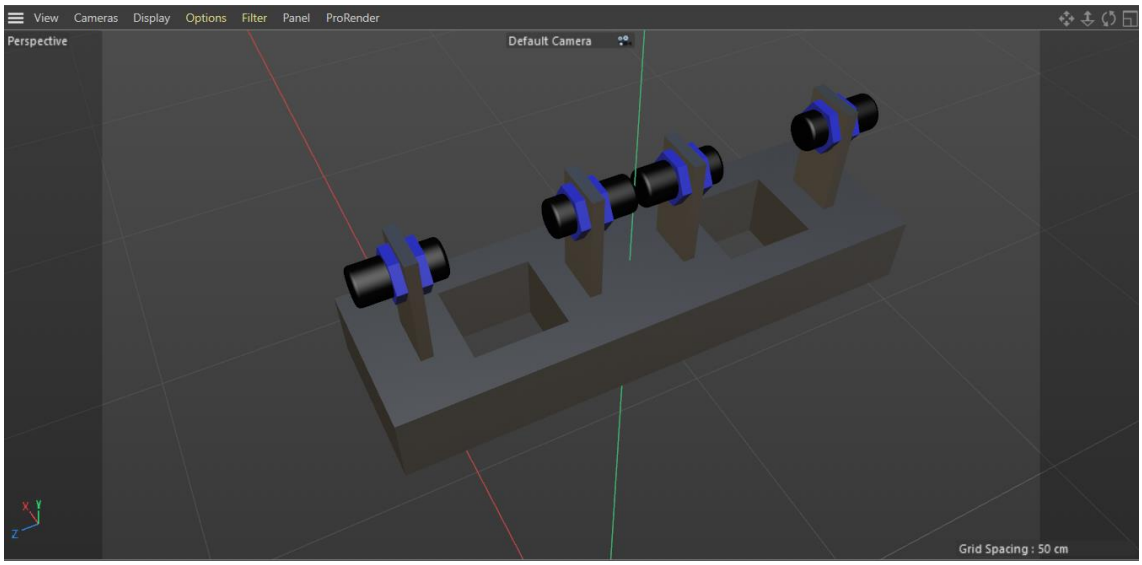

Figura 40: Sensores y soporte con texturas

Antes de pasar a la creación del modelo físico en Unity, se muestran las siguientes figuras (Figura 41-Figura 43) el modelo 3D final del alimentador de piezas.

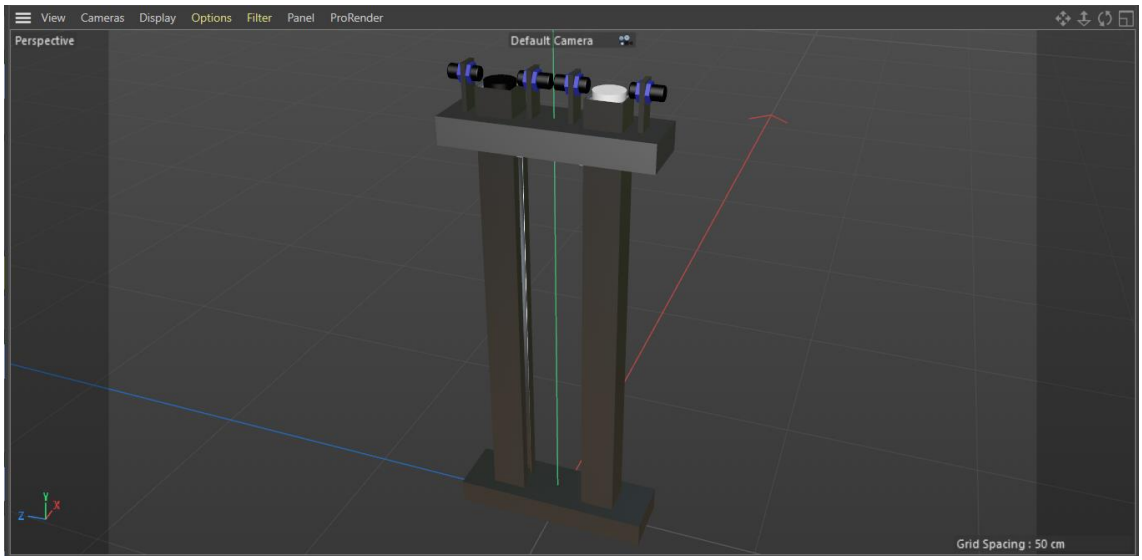

Figura 41: Vista superior diagonal del modelo 3D final

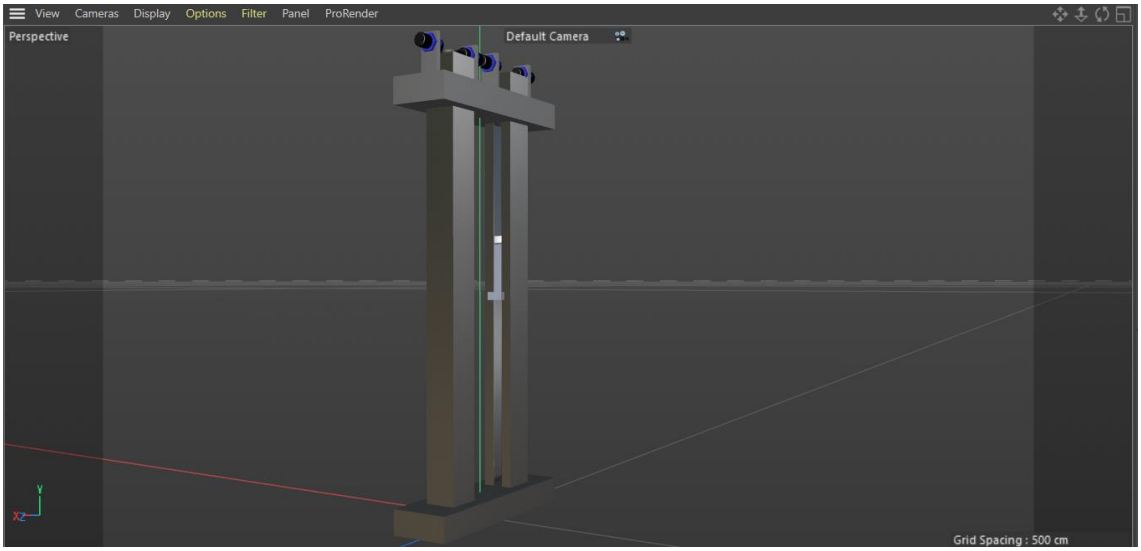

Figura 42: Vista diagonal del modelo 3D final

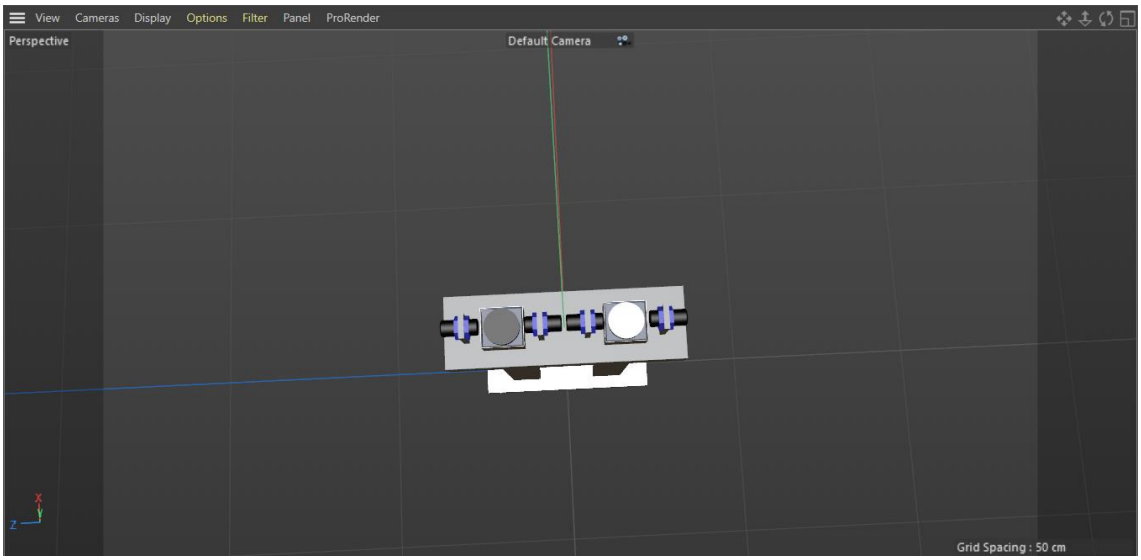

Figura 43: Vista superior del modelo 3D final

Para pasar el proyecto a Unity, se ha exportado el modelo creado como archivo en formato .fbx, que, aunque cambian algunas texturas al incorporarlo a Unity, mantiene toda la estructura y los componentes.

Como nota aclaratoria, al hacer la comparativa del modelo virtual con el físico, no se han creado los motores ni las correas que conectan estos con la plataforma que sostiene las fichas. Se ha hecho el modelo así para simplificarlo un poco, ya que el modelo solo pretende ser un ejemplo del modelo virtual de un gemelo digital, y no se pretende con el trabajo llegar a un nivel de semejanza física y estética muy elevado.

4.2 Creación del simulador final

Para abrir el modelo creado en Cinema 4D dentro de Unity, movemos el archivo exportado en formato .fbx a la carpeta Assets, dentro del proyecto TFG creado previamente el menú principal de Unity Hub. Una vez hecho esto, ya podemos tener acceso al modelo y a todos sus componentes desde la ventana del explorador del proyecto, como se puede ver en la Figura 44.

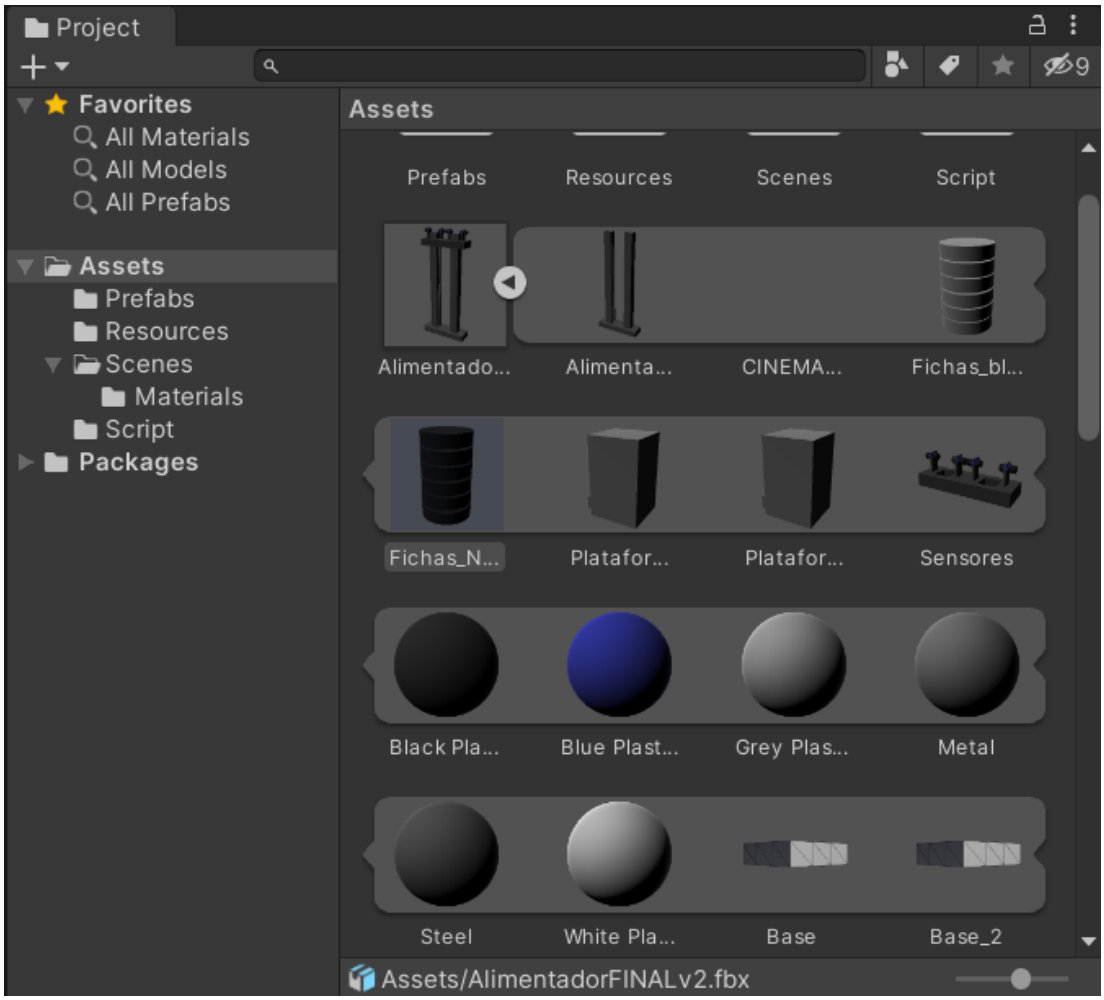

Figura 44: Carpeta Assets de la ventana del explorador

Al añadir el modelo, se puede ver como ya se tienen otros elementos en la escena. Estos son la luz direccional y la cámara principal, que junto con el alimentador de piezas se meten en una nueva escena, de nombre Alimentador. En la Figura 45 se puede ver el modelo en la ventana de escena. Se le ha añadido también un plano que actúa como suelo, aunque ello no influye en nada a la simulación de físicas.

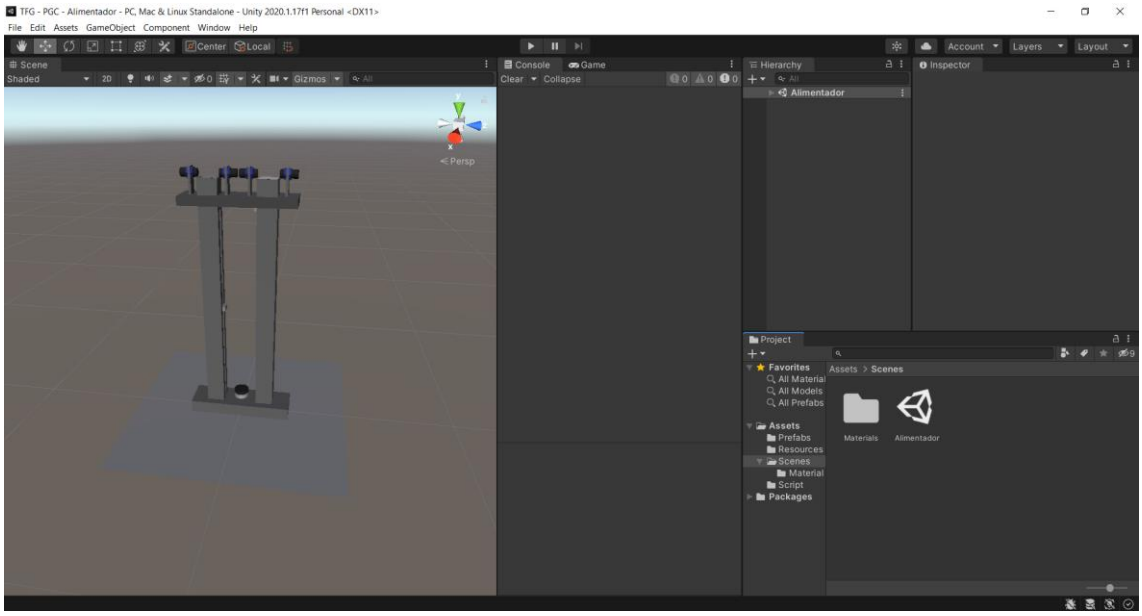

Figura 45: Modelo alimentador en Unity

El paso siguiente ha sido configurar los *colliders* de los elementos que interactúen entre sí. Un *collider* es la superficie física de contacto que rodea a un objeto, de forma que, si a un cubo le aplicas el *collider* de una esfera, el cubo al moverse por un plano estará girando, ya que su superficie de contacto es la de una esfera y no la de un cubo.

Por tanto, se ha aplicado un *collider* tipo *mesh collider*, que detecta toda la superficie del objeto, a toda la estructura del alimentador (Figura 46), es decir, a los prismas y a la base de estos; a las fichas (Figura 47); a la base de los sensores, para que la plataforma no la traspase al llegar hasta arriba; y a ambas plataformas. Sin embargo, para esta última ha surgido un problema, ya que al ser un objeto que se pone en movimiento, no se le puede aplicar un *mesh collider*, a no ser que este sea convexo. Pero si se cambia el *collider* de toda la plataforma a convexo, este no detectará el saliente y la superficie de contacto chocaría con la abertura en el prisma realizada para el saliente. Por ello, la solución que se ha tomado ha sido de configurar cada parte de la plataforma como un *mesh collider* convexo por separado.

Al resto de elementos del modelo no se le han añadido *colliders*, ya que no entran en contacto en ningún momento con otros elementos.

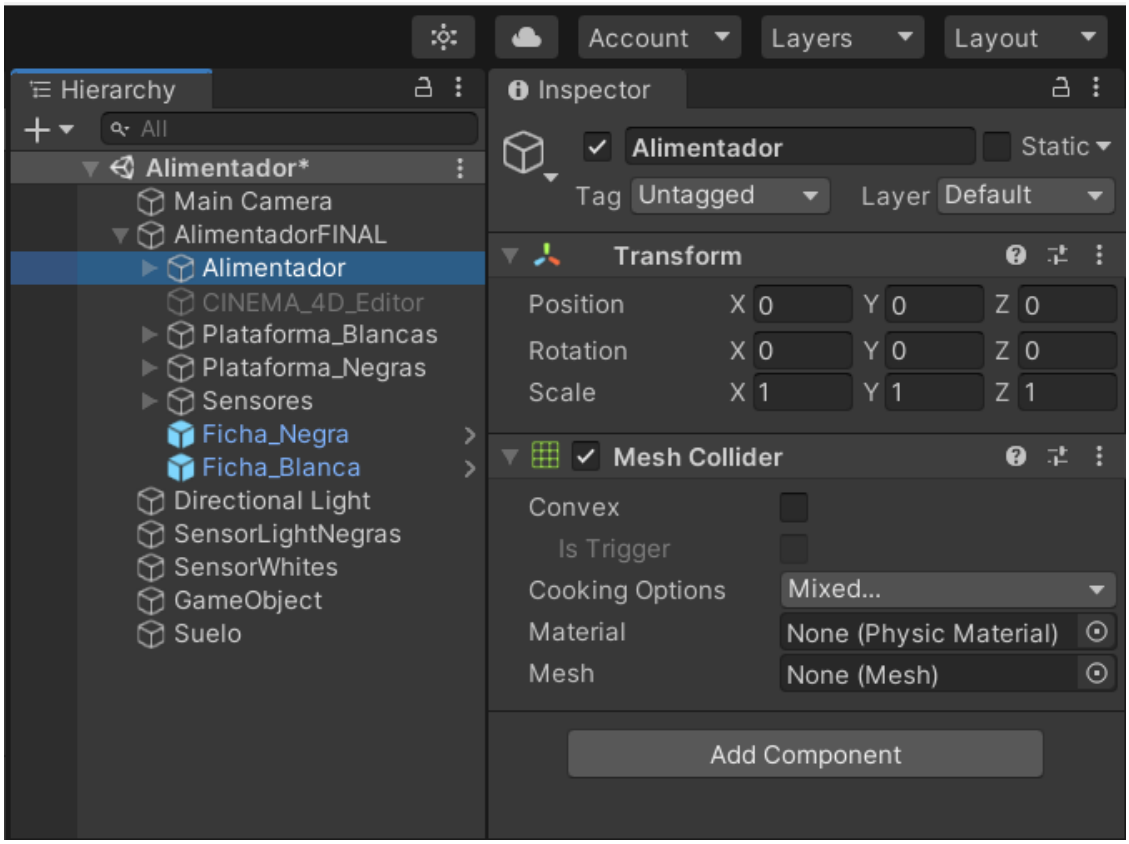

Figura 46: Collider de la estructura del alimentador

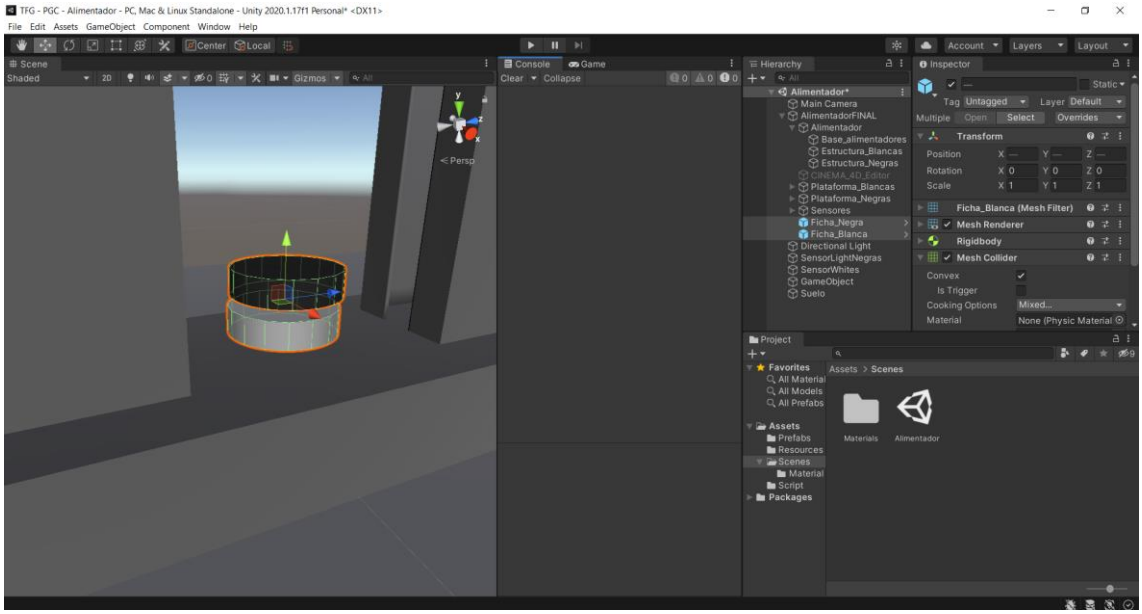

Figura 47: Collider de las fichas negra y blanca

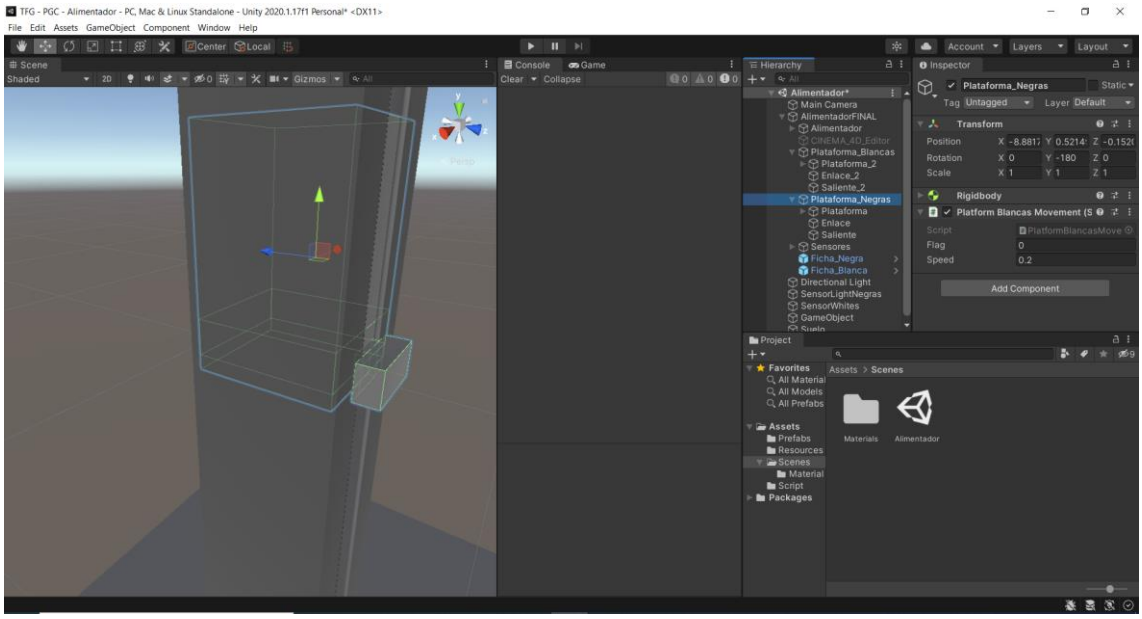

Figura 48: Collider de la plataforma de las fichas negras

A continuación, se han configurado los componentes con movimiento. Estos son las plataformas y las fichas. Para ello, se les ha añadido la propiedad de *Rigidbody*, esto es, de cuerpo rígido, de forma que, cuando se inicie la simulación, si estos objetos no tienen ninguna superficie debajo, caen por gravedad hasta topar con alguna otra superficie. Al añadir el componente *Rigidbody*, se ha pulsado la opción de usar gravedad, ya que viene por defecto deshabilitada, y para las plataformas, también se ha habilitado la opción *Is Kinematic*, esto es, cinemático, la cual permite el movimiento de dicho compente y evita que caiga a plomo por la estructura del alimentador. Con esto, se simula el comportamiento que tendría la plataforma si se hubiese creado el motor y la correa del modelo físico, que hacen que la plataforma se quede suspendida en el aire cuando el motor está en reposo.

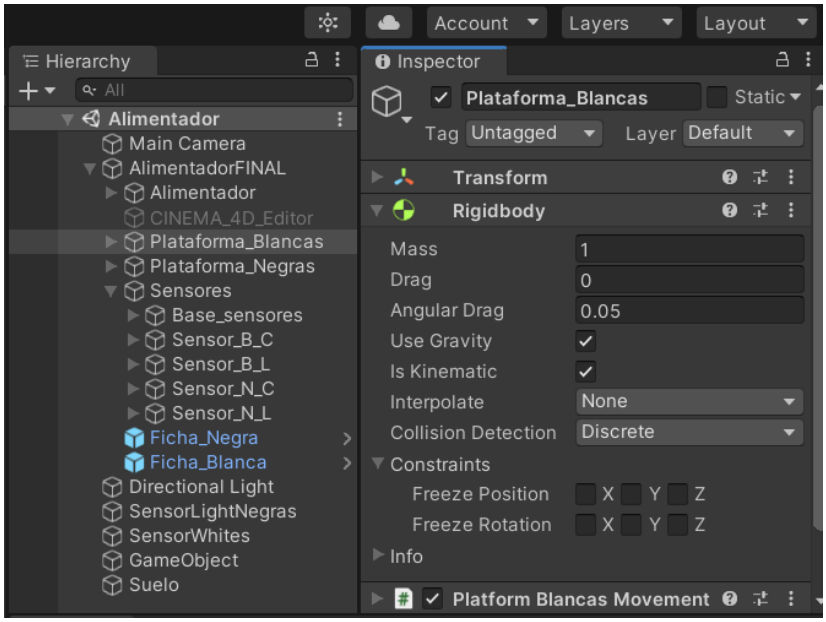

Figura 49: Plataforma de fichas blancas como cuerpo rígido con gravedad y cinemático

Una vez configurados todos los elementos físicos, se han configurado los sensores. Para ello, se ha creado un pequeño cilindro en la salida de cada uno de los alimentadores, a la altura de ambos pares de sensores, se les ha añadido un *mesh collider* convexo, con la opción *Is Trigger*. Esta opción hace invisible al objeto, pero este sigue detectando colisiones, de forma que en cuanto una ficha entre en contacto con el cilindro, este lo detectará. En la Figura 50 se mustran ambos sensores,

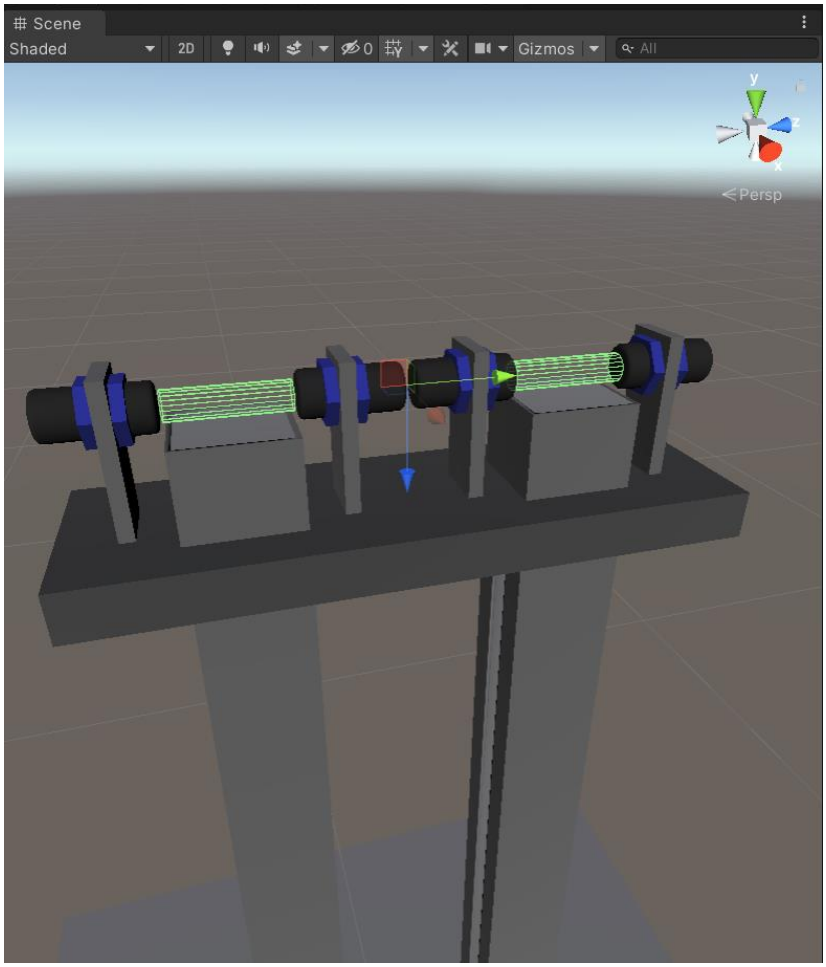

Figura 50: Colliders de los sensores capacitivos

A continuación, se mustra también el código empleado para la detección de una ficha por parte de los sensores (uno para cada sensor, aunque solo se muestra el de las fichas blancas):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SensorWhite : MonoBehaviour
{
    public bool flag;

    void OnTriggerStay(Collider other)
    {
        flag = true;
        if (Input.GetKey(KeyCode.KeypadMinus))
            Destroy(other.gameObject);
    }

    void OnTriggerExit(Collider other)
    {
        flag = false;
        Debug.Log("No hay pieza");
    }

}
```

Una vez exlicado todo lo anterior, se detalla ahora el funcionamiento del alimentador, que se ha concebido de la forma más simple posible. Se han creado códigos en C# para el control manual del alimentador, de manera este se puede controlar mediante el teclado del ordenador.

Para simular la que un brazo robótico o un operario ponen una determinada ficha en la entrada del alimentador para almacenarla o se llevan una determinada ficha que se quiere extraer, se han creado, por un lado, dos programas (uno para cada alimentador) que crean clones de las fichas en la posición de entrada y salidad de los alimentadores cuando se pulsa una tecla determinada; y otros dos programas que eliman dicho clon cuando este se encuentra en el sensor y se pulsa una determinada tecla. Se muestra a continuación el código de poner fichas, ya que la sentencia necesaria para eliminar el clon creado se ha programado en el sensor

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PonerFichaBlanca : MonoBehaviour
{
    public Transform pos;
    public Rigidbody NuevaFicha;
    public bool flagpieza;

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.KeypadPlus))
        {
            Instantiate(NuevaFicha,pos.position,pos.rotation);
            flagpieza = true;
        }
    }
}
```

Las acciones que hace el alimentador son muy sencillas. Cada alimentador puede almacenar un tipo de ficha (negra o blanca) o extraerlas. El almacenamiento de una pieza consiste en que, si se encuentra una pieza ocupando el sensor de piezas, la plataforma desciende hasta que se deje de detectar dicha pieza, con lo que se consigue el almacenaje de la pieza. La extracción es el proceso contrario, si no se detecta ninguna pieza, la plataforma ascenderá hasta que el sensor detecte la siguiente pieza, momento en el que se detendrá.

Como ya se ha explicado antes, gracias a las acciones de poner o quitar pieza, estas van apareciendo o desapareciendo de la escena, simulando que un brazo robótico o un operario las pone o las quita. A continuación, se muestran los códigos de ambas operaciones (solo para las blancas):

- Almacenar pieza:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AlmacenarPiezaBlanca : MonoBehaviour
{
    public Sensor SensorAlmacenamiento;
    public PlatformBlancasMovement PlatB;
    public bool flagboton;
    public bool flag;

    void Start()
    {
        SensorAlmacenamiento = GameObject.Find("SensorWhites").GetComponent<Sensor>();
        PlatB =
GameObject.Find("Plataforma_Blancas").GetComponent<PlatformBlancasMovement>();
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Keypad7))
            flagboton = true;
        if(flagboton)
        {
            Debug.Log("Almacenando pieza");
            flag = SensorAlmacenamiento.flag;

            if (flag == true)
                PlatB.flag = -1;
            if (flag == false)
            {
                PlatB.flag = 0;
                flagboton = false;
                Debug.Log("Pieza almacenada");
            }
        }
    }
}
```

- Extraer piezas:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ExtraerFichaBlanca : MonoBehaviour
{
    public Sensor SensorExtraccion;
    public PlatformBlancasMovement PlatB;
    public bool flagboton;
    public bool flag;

    void Start()
    {
        SensorExtraccion = GameObject.Find("SensorWhites").GetComponent<Sensor>();
        PlatB =
GameObject.Find("Plataforma_Blancas").GetComponent<PlatformBlancasMovement>();
```

```
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Keypad9))
            flagboton = true;
        if (flagboton)
        {
            Debug.Log("Extrayendo pieza");
            flag = SensorExtraccion.flag;

            if (flag == false)
                PlatB.flag = 1;
            if (flag == true)
            {
                PlatB.flag = 0;
                flagboton = false;
                Debug.Log("Pieza extraida");
            }
        }
    }
}
```

Para el movimiento de las plataformas, las cuales también se pueden mover manualmente pulsando determinadas teclas, se han empleado el siguiente código (se muestra solo el de la plataforma blanca):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlatformBlancasMovement : MonoBehaviour
{
    private Rigidbody Platform;
    public int flag = 0;
    public float Speed = 0.2f;

    void Start()
    {
        Platform = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        if (Input.GetKey(KeyCode.UpArrow))
            flag = 1;
        if (Input.GetKey(KeyCode.DownArrow))
            flag = -1;
        if (Input.GetKey(KeyCode.Space))
            flag = 0;

        if (flag == 1)
            Platform.MovePosition(Platform.position + transform.up * Speed *
Time.deltaTime);

        if (flag == -1)
            Platform.MovePosition(Platform.position - transform.up * Speed *
Time.deltaTime);
    }

}
```

Para añadir estos códigos al funcionamiento del modelo, se han añadido desde la ventana de proyecto, en la carpeta de *Scrip*. Como se puede ver en la Figura 51, todos los *scrips* creados, exceptuando los de los sensores y los de las plataformas se han añadido a *AlimentadorFINAL*. Algunos de los *scripts* que aparecen en la Figura

51 no han sido utilizados, ya que se han ido probando varias combinaciones hasta dar con el funcionamiento óptimo

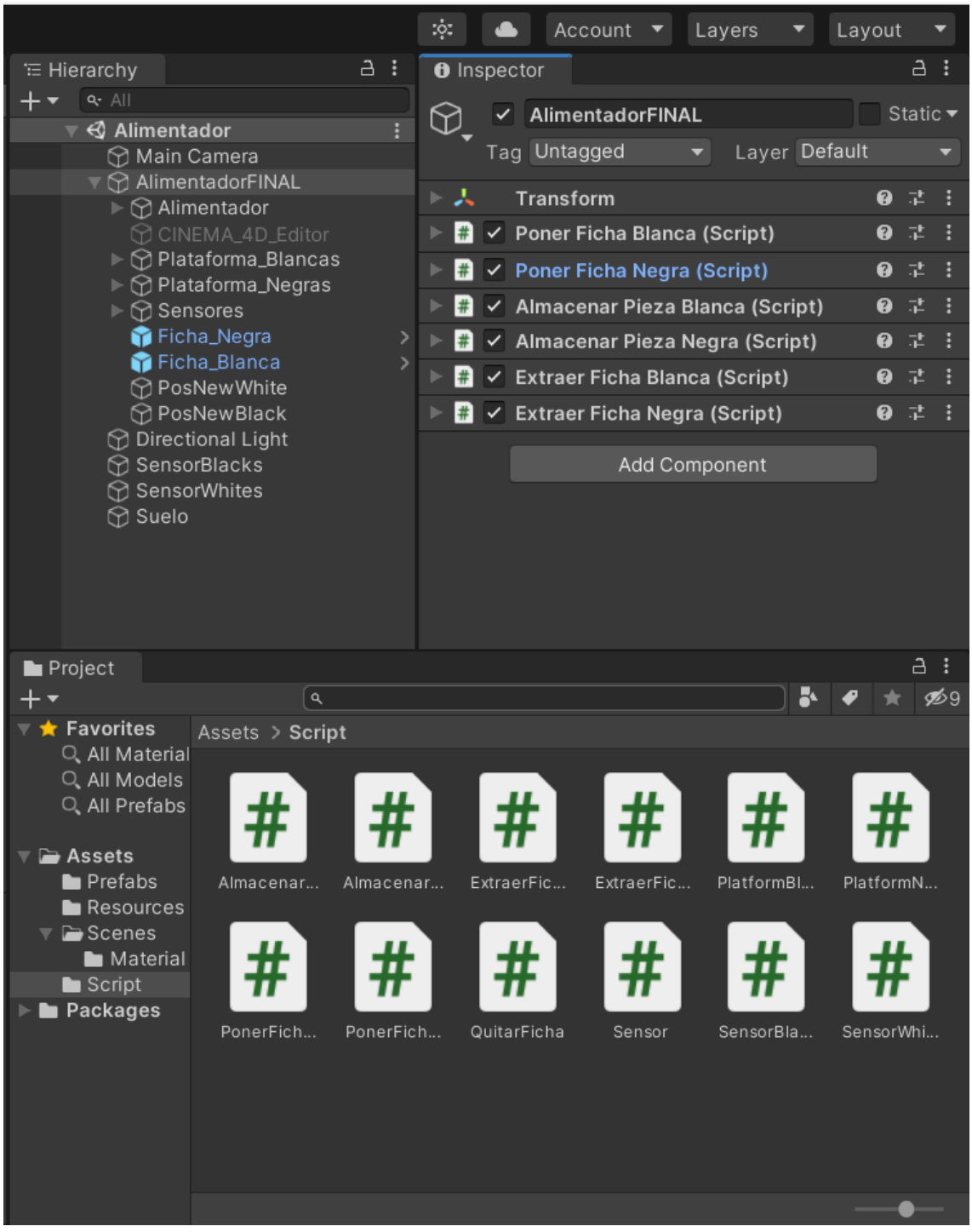

Figura 51: *Scripts* del proyecto añadidos al alimentador

Cabe comentar que en varios de los códigos se han programado mensajes a la ventana de consola para que el programa vaya indicando las acciones que va realizando. Por último, se pueden ver varias imagenes de la simulación del proceso.

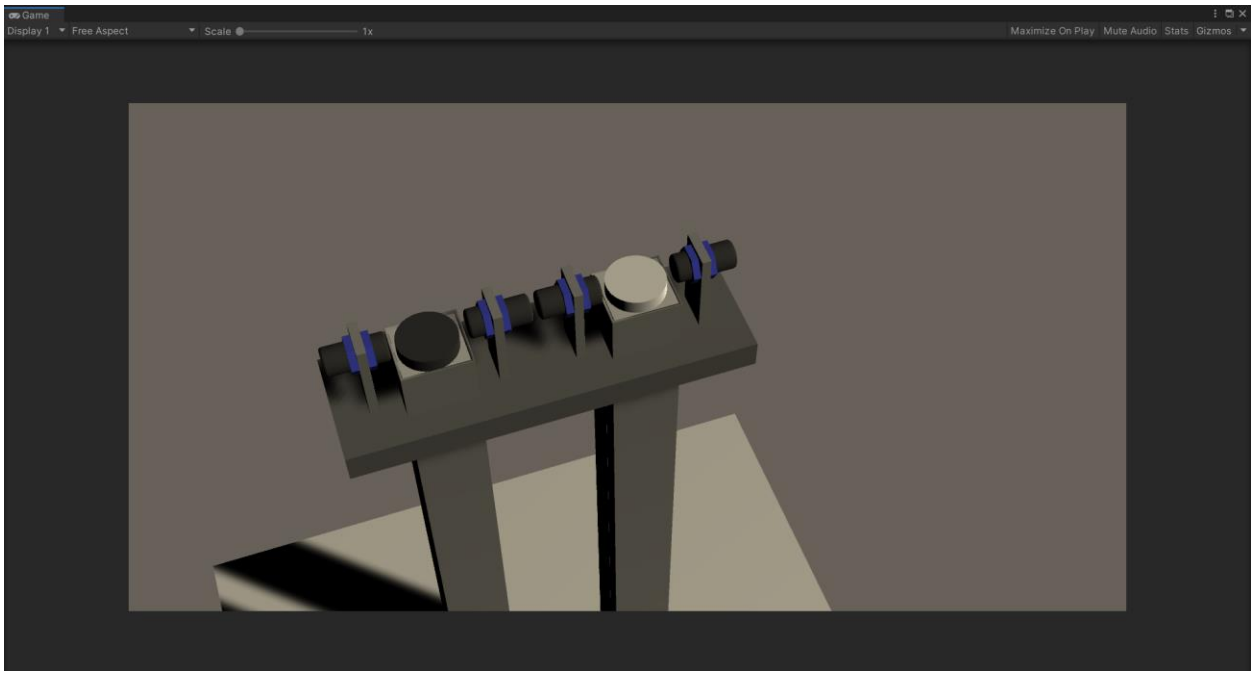

Figura 52: Vista superior de la simulación del alimentador de piezas

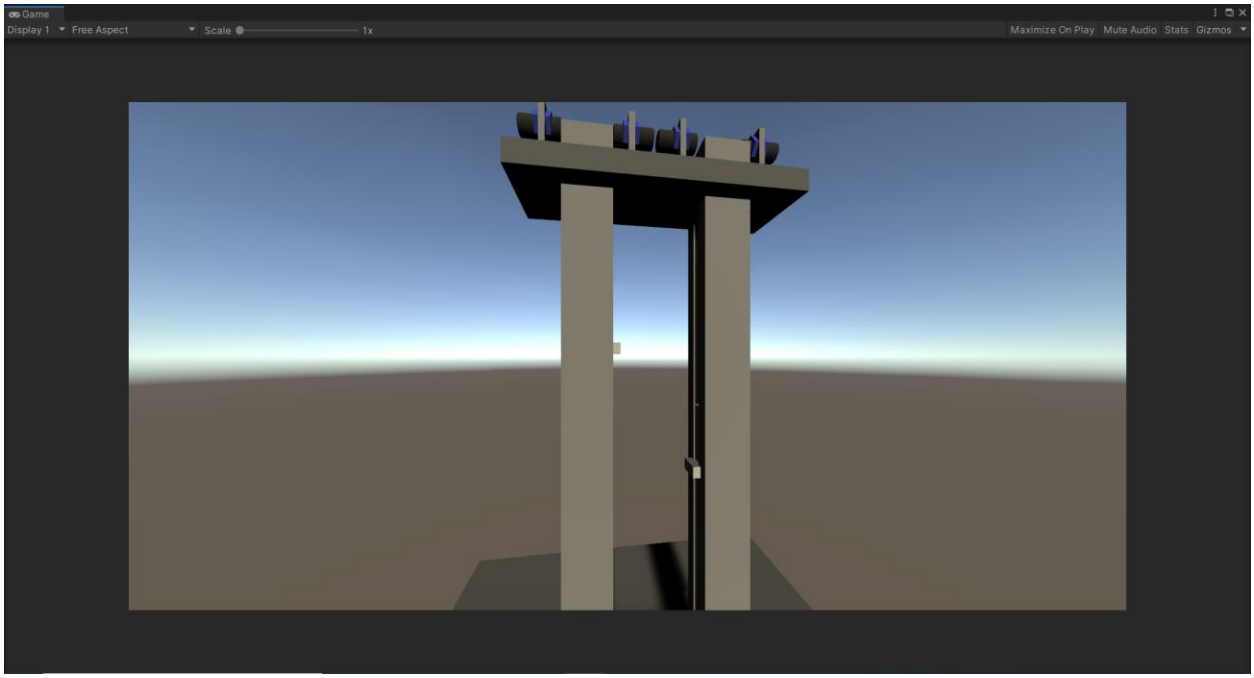

Figura 53: Vista frontal de la simulación del alimentador de piezas

Figura 54: Vista alejada de la simulación del alimentador de piezas

5 CONCLUSIÓN Y TRABAJOS FUTUROS

Este proyecto ha consistido en una pequeña introducción sobre los Gemelos Digitales. Se ha estudiado el desarrollo de este concepto a lo largo de los últimos años y sus diversas aplicaciones. Se han evaluado herramientas industriales de diversos desarrolladores y para poder programar de una forma totalmente flexible y a bajo nivel, se ha optado por usar el software de diseño de videojuegos Unity 3D. Por último, se ha diseñado un pequeño modelo 3D que simula un alimentador de piezas de una célula de fabricación flexible.

Con este modelo, solo se tendría la parte virtual del Gemelo Digital del alimentador, además del propio alimentador físico que se ha modelado. Se proponen, como trabajos futuros, los siguientes pasos para completar el Gemelo Digital: obtener los datos del modelo físico; generar una serie de servicios como puede ser optimizar el gasto de una operación concreta: almacenar o extraer una o varias piezas, de forma que dichas operaciones se hagan de la manera más económica posible, energéticamente hablando; y, por último, crear las conexiones entre el modelo físico y el modelo digital con un PLCs para poder así obtener los datos precisos en tiempo real en el modelo virtual.

Como posible trabajo futuro, se propone hacer un modelo digital de cada una de las estaciones de trabajo que componen la célula de fabricación flexible del laboratorio del Departamento de Ingeniería de Sistemas y Automática de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla. Una vez creados todos los modelos virtuales de la célula, se podrían ejecutar en paralelo cada una de las simulaciones y obtener diferentes datos de componentes, como perdidas de presión en los pistones, perdida por rozamiento en las cintas transportadoras, etc. Dichos datos obtenidos gracias al modelado físico y matemático, aplicados al modelo virtual, pueden servir para modificar malfuncionamientos o pequeños fallos en el funcionamiento de la célula.

REFERENCIAS

[1] L. D. Xu, E. L. Xu y L. Li, «Industry 4.0: state of the art and future trends,» *International Journal of Production Research,* vol. 56:8, pp. 2941-2962, 2018.

[2] Q. L. Qi y F. Tao, «Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison,» *IEEE Access,* vol. 6, pp. 3585-3593, Enero 2018.

[3] B. Barricelli, E. Casiraghi y D. Fogli, «A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications,» *IEEE Access,* 14 Noviembre 2019.

[4] E. J. Tuegel, A. R. Ingraffea, T. G. Eason y S. M. Spottswood, «Reengineering aircraft structural life prediction using a digital twin,» *Int. J. Aerosp. Eng,* vol. 2011, pp. 1687-5966, Agosto 2011.

[5] E. Glaessgen y D. Stargel, «The digital twin paradigm for future NASA and U.S. Air Force vehicles,» de *Proc. 53rd AIAA/ASME/ASCE/AHS/ASC Struct. Struct. Dyn. Mater. Conf.*, Honolulu, Hawaii, 2012.

[6] M. Grieves, «Digital twin: Manufacturing excellence through virtual factory replication,» *White paper,* 2014.

[7] F. Tao, H. Zhang, A. Liu y A. Y. C. Nee, «Digital Twin in Industry: State-of-the-Art.,» *IEEE Transactions on Industrial Informatics.,* vol. 15, nº 4, pp. 2405-2415, Abril 2019.

[8] Q. Qi, F. Tao, T. Hu, N. Anwer, A. Liu, Y. Wei, L. Wang y A. Nee, «Enabling technologies and tools for digital twin,» *Journal of Manufacturing Systems,* vol. 58, pp. 3-21, 2021.

[9] R. Rosen, G. V. Wichert, G. Lo y K. D. Bettenhausen, «About the Importance of autonomy and digital twins for the future of manufacturing,» *IFAC PapersOnLine,* vol. 48, nº 3, pp. 567-572, 2015.

[10] T. Gabor, L. Belzner, M. Kiermeier, M. T. Beck y A. Neitz, «A simulation-based architecture for smart cyber-physical systems,» de *IEEE International Conference on Autonomic Computing (ICAC)*, Wurzburg, Germany, 2016.

[11] G. Knapp, T. Mukherjee, J. Zuback, H. Wei, T. Palmer, A. De y T. DebRoy, «Building blocks for a digital twin of additive manufacturing,» *Acta Materialia,* vol. 135, pp. 390-399, Agosto 2017.

[12] A. M. Lund, «Digital wind farm system». U. S. Patente US 2016/0333855 A1, 17 Noviembre 2016.

[13] A. M. Lund, «Digital twin interface for operating wind farms». U. S. Patente US 9,995,278 B2, 12 Junio 2018.

[14] L. Wang y A. M. Canedo, «Human programming interfaces for machinehuman interfaces». U. S. Patente 15/284,571, 20 Abril 2017.

[15] R. Johnson, «Method for creating a digital twin of a room». Europa Patente 16186640.5, 7 Marzo 2018.

[16] «DENiM project,» [En línea]. Available: https://cordis.europa.eu/project/id/958339.

[17] J. Gómez Jiménez, «Contribuciones al desarrollo de plataforma estandarizada para la simulación de sistemas de automatización,» Sevilla, 2020.

[18] J. Gómez Jiménez, «Modelado y control de quadrotors en la plataforma UNITY 3D,» Sevilla, 2020.

[19] «Siemens Digital Industries Software: Tecnomatix,» [En línea]. Available: https://www.plm.automation.siemens.com/global/es/products/tecnomatix/.

[20] «Siemens Plant Simulation,» [En línea]. Available: https://www.plm.automation.siemens.com/store/es-es/trial/plant-simulation.html.

[21] «Siemens Digital Industries Software: NX,» [En línea]. Available: https://www.plm.automation.siemens.com/global/es/products/nx/.

[22] «Visual Components,» [En línea]. Available: https://www.visualcomponents.com/.

[23] «Visual Components: Archivos CAD soportados,» [En línea]. Available: https://www.visualcomponents.com/supported-cad-files/.

[24] «Download Unity,» [En línea]. Available: https://unity3d.com/get-unity/download. [Último acceso: 05 07 2021].

[25] «Unity Documentation: Mesh Renderer,» [En línea]. Available: https://docs.unity3d.com/Manual/class-MeshRenderer.html. [Último acceso: 05 07 2021].

[26] «Maxon: Cinema 4D,» [En línea]. Available: https://www.maxon.net/es/cinema-4d. [Último acceso: 06 07 2021].

[27] Autor, «Este es el ejemplo de una cita,» *Tesis Doctoral,* vol. 2, nº 13, 2012.

[28] O. Autor, «Otra cita distinta,» *revista,* p. 12, 2001.

[29] Avantek, «Webinar sobre MCD,» [En línea]. Available: https://www.youtube.com/watch?v=TjRMEnKAUQ0&ab_channel=AvantekS.L..

[30] «Unity Store: Planes y precios,» [En línea]. Available: https://store.unity.com/es#plans-individual.

[31] «Academy Visual Components,» [En línea]. Available: https://academy.visualcomponents.com/.