Jornadas de Automática

Aproximación Basada en Unity para el Modelado Digital de Sistemas de Automatización

Sánchez, A.[a,*], Illana, S.[a], Casado, P[a], Ruano, I.[a], Estévez, E.[a]

[a] *Dpto. Ingeniería Electrónica y Automática Escuela Politécnica Superior de Jaén, Universidad de Jaén, Campus Las Lagunillas s/n, 23071, Jaén, España.*

To cite this article: Sánchez, A., Illana, S., Casado, P[a], Ruano, I., Estévez, E., 2024. Unity-Based Approach for Digital Modelling of Automation Systems. Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10897

Resumen

La simulación 3D se ha convertido en una herramienta esencial en la automatización industrial moderna. Está alineada con los objetivos de la Industria 4.0, que buscan mejorar la eficiencia y reducir costos en entornos de manufactura complejos. Hoy en día en el mercado hay disponibles entornos de simulación 3D diseñados para abordar los desafíos de replicar con precisión entornos industriales complejos. Lamentablemente, son productos de pago con una costosa curva de aprendizaje, y en los que los usuarios no pueden añadir nuevos componentes que no se encuentren contemplados en sus librerías. Este trabajo, explora cómo utilizar la herramienta Unity, un desarrollo integral utilizado principalmente para la creación de videojuegos y experiencias interactivas en 2D-3D, realidad aumentada y virtual, en el modelado digital de sistemas de automatización. Específicamente, presenta una metodología a seguir con objeto de minimizar la curva de aprendizaje y, adicionalmente, añade dos componentes básicos e indispensables en automatización como son los sensores magnéticos y los cilindros de doble efecto.

Palabras clave: Industria 4.0, Modelado Digital, Automatización Industrial.

Unity-Based Approach for Digital Modelling of Automation Systems

Abstract

3D simulation has become an essential tool in modern industrial automation. It aligns with the goals of Industry 4.0, which aim to improve efficiency and reduce costs in complex manufacturing environments. Nowadays, there are 3D simulation environments available in the market designed to address the challenges of accurately replicating complex industrial settings. Unfortunately, these are paid products with a steep learning curve, and users cannot add new components that are not included in their libraries. This work explores how to use Unity, a comprehensive development tool primarily used for creating video games and interactive experiences in 2D, 3D, augmented reality (AR), and virtual reality (VR), for digital modelling of automation systems. Specifically, it presents a methodology to follow in order to minimize the learning curve and also introduces two basic and essential components in automation: magnetic sensors and double-acting cylinders.

Keywords: Industry 4.0, Digital Modelling, Automatization.

1. Introducción

La simulación 3D en el ámbito de la automatización industrial es una herramienta poderosa para la optimización de procesos y la reducción de costos (Akpan and Offodile, 2024), (de Paula Ferreira et al., 2020), (de Paula Ferreira et al., 2021). En dichos trabajos se remarca cómo la simulación 3D permite el modelado detallado y análisis dinámico de procesos industriales complejos lo cual es fundamental a la hora de identificar y resolver problemas potenciales antes de la implementación real, mejorando la eficiencia operativa. Utilizar simulaciones en 3D reduce la necesidad de prototipos físicos, lo que ahorra tiempo y dinero. Además, facilita la experimentación con diferentes configuraciones de sistemas sin interrumpir la producción real (Reducción de Costos). Asimismo, las simulaciones permiten pruebas y validaciones

*Autor para correspondencia: asgarcia@ujaen.es

rigurosas de nuevos diseños y estrategias de control en un entorno seguro. Esto garantiza que los sistemas funcionen de manera óptima y con mayor confiabilidad una vez implementados (Mejora del Rendimiento). Por último, las plataformas de simulación que proporcionen un entorno interactivo permiten aprender y practicar sin riesgos a los operadores y técnicos (Formación y capacitación).

Todos estos beneficios hacen de la simulación 3D una herramienta esencial en la automatización industrial moderna, alineándose con los objetivos de la Industria 4.0 de mejorar la eficiencia y reducir costos en entornos de manufactura complejos (Dintén et al., 2021), (Folgado et al., 2024).

Hoy en día en el mercado hay disponibles entornos de simulación 3D diseñados para abordar los desafíos de replicar con precisión entornos industriales complejos. Por ejemplo, Siemens presenta NX + SIMIT (Siemens NX y Simit, 2024) como una solución avanzada que destaca en la ingeniería integrada y la simulación de sistemas mecatrónicos. Su integración completa con otros productos de Siemens permite un flujo de trabajo fluido y consistente. Sin embargo, esta integración puede ser menos eficiente con otros fabricantes. La herramienta tiene una curva de aprendizaje alta y requiere una inversión significativa a largo plazo debido a los requisitos de hardware y capacitación. Factory I/O ("Factory I/O – Next-Gen PLC Training," 2024) es una herramienta atractiva que ofrece una amplia gama de escenas y componentes industriales listos para usar, ideal para el diseño y la simulación de escenarios. Es fácil de usar y tiene un costo relativamente bajo (Vargas et al., 2023). Simumatik ("Simumatik | State of the Art Emulation," n.d.) destaca en los siguientes aspectos: facilidad de uso, posibilidad de personalizar y crear modelos propios aunque, como punto negativo, no permite creación de nuevos componentes. La plataforma proporciona acceso a sistemas en tiempo real y recursos basados en la nube las 24 horas del día, lo que la hace ideal para el aprendizaje remoto. Además, Simumatik permite trabajar con otras marcas, como CODESYS, ampliando sus posibilidades de integración.

La limitación fundamental de todas ellas es que están enfocadas en la definición de plantas o líneas producción formadas por módulos de carga y sistemas de transporte (cintas transportadoras). Hasta donde los autores conocen, no ofrecen la posibilidad de incorporar nuevos elementos entre los que se encuentran, por ejemplo, cilindros de doble efecto o sensores magnéticos, elementos básicos y fundamentales en una estación industrial. Además, se tratan de herramientas de pago.

Por otro lado, los motores gráficos, originalmente diseñados para el desarrollo de videojuegos, han experimentado una notable evolución en su estructura y funcionalidad (ITAINNOVA, 2015). Esta evolución ha llevado a este tipo de plataformas a trascender su función inicial, impactando no solo en el entretenimiento puro, sino también en aplicaciones para otros sectores, como la formación y entretenimiento a través de Serious Games o aplicaciones de Realidad Virtual enfocadas al sector industrial. Así, los entornos virtuales se han convertido en herramientas cruciales para la verificación del comportamiento de instalaciones, minimizando riesgos y detectando posibles errores sin afectar a componentes reales. En este contexto, es esencial la simulación precisa de fuerzas en objetos es esencial. Dichas fuerzas que surgen en los objetos físicos de la instalación se realizan a través de lo que se conoce como motor físico i.e. sistema software que implementa una simulación aproximada de ciertos sistemas físicos, incluyendo dinámica de sólidos rígidos, detección de colisiones y dinámica de fluidos en tiempo real.

En el mercado hay diferentes herramientas entre las que destacan: Blender software de código abierto y gratuita utilizada para modelado 3D, animación, renderizado, composición y postproducción, edición de video, creación de aplicaciones 3D interactivas, videojuegos y simulación física (Brito, 2024). Es ampliamente reconocida por su potente conjunto de características y su versatilidad, que la hacen adecuada para una amplia gama de aplicaciones en industrias como la animación, los videojuegos, la realidad virtual y aumentada, y la visualización arquitectónica e industrial. Su naturaleza de código abierto y la amplia comunidad de usuarios y desarrolladores contribuyen continuamente a su mejora y adaptabilidad a nuevas aplicaciones y desafíos, aunque tiene una curva de aprendizaje muy elevada. SketchUp ("SketchUp Blog," 2024) software de modelado 3D ideal para arquitectura y diseño de interiores aunque tiene amplia biblioteca de modelos 3D no permite la definición de nuevos componentes por lo que adaptarla al sector de automatización es complejo. Unity es una plataforma de desarrollo integral utilizada principalmente para la creación de videojuegos y experiencias interactivas en 2D, 3D, realidad aumentada (AR) y realidad virtual (VR) (Coutinho, 2022). Unity es conocida por su flexibilidad, facilidad de uso y su capacidad para exportar proyectos a múltiples plataformas, incluyendo consolas, PC, dispositivos móviles y web. La combinación de su potente motor gráfico y su entorno de desarrollo intuitivo lo convierte en una herramienta popular entre desarrolladores y diseñadores de todo el mundo. Además, tiene una versión gratuita y permite la definición de nuevos componentes.

Este trabajo se centra en analizar la utilidad de Unity en automatización. Para ello, se propone una metodología para poder definir un modelo digital de automatización en Unity. Además, se han añadido dos conceptos fundamentales e imprescindibles en todo modelo/sombra y gemelo Digital en automatización como son: los sensores magnéticos y los cilindros de doble efecto. También se incluye una demostración de su uso a través de un modelo digital donde se manipula un vástago de doble efecto. Este trabajo finaliza con unas conclusiones y trabajos futuros.

2. Metodología para definir Modelos Digitales con Unity

La creación de un modelo digital en Unity implica varias etapas, desde la conceptualización hasta la implementación en la plataforma. Este apartado identifica los pasos a seguir a la hora de definir un modelo digital en Unity. Para ello se tomará como ejemplo ilustrativo un elemento básico y a la vez indispensable en automatización como es el LED.

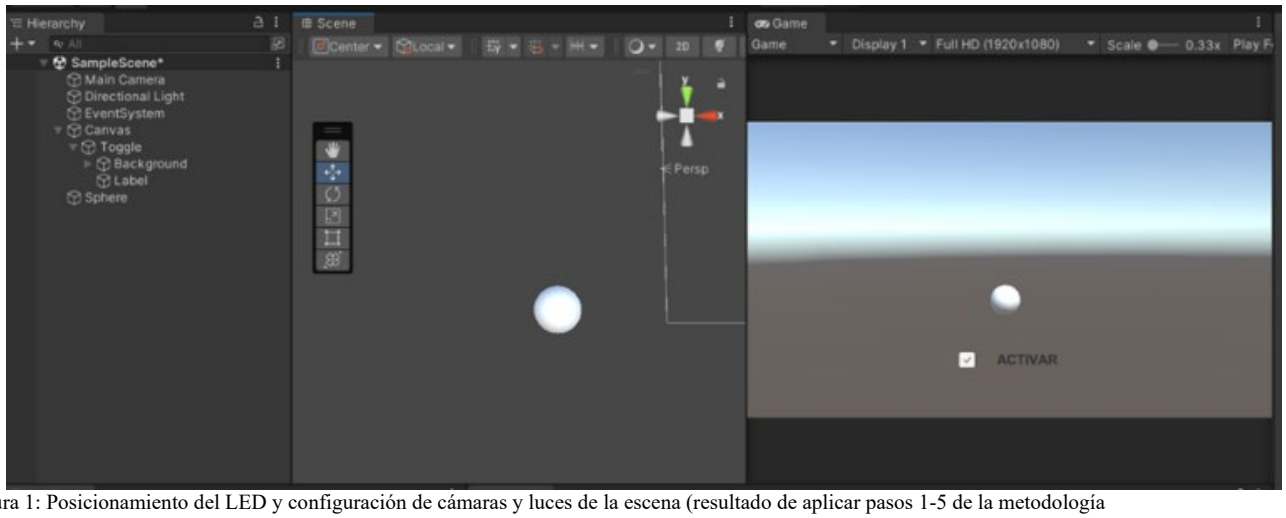

Figura 1: Posicionamiento del LED y configuración de cámaras y luces de la escena (resultado de aplicar pasos 1-5 de la metodología

Paso 1. Conceptualización del modelo digital. Consiste en Identificar el propósito y los requisitos del modelo digital y definir las características clave y la funcionalidad requerida. En el caso del LED es un activo que puede cambiar de estado de manera instantánea en respuesta a ciertas condiciones o acciones predefinidas.

Paso 2. Creación de activos: diseñar o adquirir los activos necesarios para el modelo digital, como modelos 3D, texturas, y sonidos si fuese necesario (véase apartado 2 del presente trabajo). Este paso finaliza al importar los activos a Unity y organizarlos en sus carpetas correspondientes. Tomando como ejemplo el LED, es posible generar un modelo 3D utilizando un software de modelado como Blender, Maya o 3ds Max, y luego importarlo a Unity. Otra opción es acceder a la Asset Store de Unity, la cual ofrece una amplia variedad de activos pre-creados, si bien, esta última alternativa implica un coste. Para este ejemplo, la opción más directa consiste en utilizar las primitivas 3D básicas, como esferas o cilindros, para dar forma al LED y, posteriormente, ajustar sus propiedades como color, tamaño y otros atributos para que se asemejen a un LED.

Paso 3. Implementación de elementos de interfaz de usuario (UI). Diseñar y agregar elementos de UI, utilizando botones, toggles, sliders, etc... Configurar la funcionalidad de estos elementos para interactuar con el modelo digital. Para el caso del LED, se ha creado de un Toggle que será el responsable de activarlo y desactivarlo.

Paso 4. Definición de la escena. Creación de una nueva escena sobre la que se definirá el modelo digital. Tras crear una nueva escena en Unity ya se dispone los activos en la escena de acuerdo con el diseño previamente establecido (pasos 2 y 3). Para el caso específico del LED, además de la esfera, como se explicó anteriormente, también se ha creado un Toggle que será el responsable de activarlo y desactivarlo.

Paso 5. Configuración de cámaras y luces. Ajustar la posición, rotación y configuración de las cámaras para obtener la perspectiva deseada del modelo. Se aconseja, al manipular el Canvas y sus elementos, tener la ventana del juego abierta en paralelo a la de la escena, dado que será necesario posicionar la Cámara Principal enfocando la escena que se desea proyectar, mientras que, en el Canvas, se ajustan las ubicaciones de los elementos mediante la ventana del juego. La Figura 1 presenta el resultado de los pasos 1-5 de la metodología en la definición del modelo digital de un LED.

Paso 6. Implementación de interactividad. Agregar scripts y lógica de juego para permitir la interacción del usuario con el modelo digital. Para el caso del LED, se ha desarrollado un Script llamado 'ConexiónLED, para controlar la funcionalidad del LED, que adquirirá un tono rojo claro cuando esté inactivo y rojo fuerte cuando se active. La Figura 2 muestra dicho script que pasa a describirse a continuación.

```
public class ConexiónLED : MonoBehaviour
{
    public Toggle toggle;
    public GameObject esfera;

    private Color colorRojoFuerte = new Color(1f, 0f, 0f, 1f);
    private Color colorRojoClaro = new Color(1f, 0.5f, 0.5f, 1f);

    void Start(){

    }

    public void Update(){
        if (toggle.isOn){
            esferaRenderer.material.color = colorRojoFuerte;
        }else{
            esferaRenderer.material.color = colorRojoClaro;
        }
}
```

Figura 2: Script Conexión LED

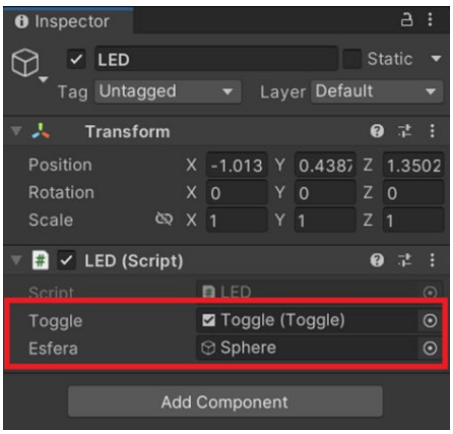

Figura 3: Componentes externos del script LED

En primer lugar, aparece una definición de un conjunto de variables que posteriormente se utilizarán en el script. En concreto las variables públicas se establecerán desde el panel inspector de Unity (véase Figura 3). La variable 'toggle' y 'esfera' representan el interruptor utilizado para cambiar el color y el objeto esférico en sí, respectivamente. La variable 'esferaRenderer' se utilizará para acceder al componente 'esfera' y será el responsable de las propiedades visuales del objeto entre las que se encuentra el color. Por otro lado, hay dos variables privadas para representar los colores: 'colorRojoFuerte' un tono de rojo intenso, mientras 'colorRojoClaro', se inicializa con un tono de rojo más suave. En cuanto a los métodos, este script se compone de los dos comentados anteriormente. El método Start() en este caso no requiere codificación ya que todas las variables de interés se inicializan desde la vista Inspector. El método Update() evalúa el estado del toggle para determinar qué color debe tener la esfera. Si el toggle está activado (toggle.isOn es verdadero), se asignará el color rojo fuerte al material de la esfera. Si el toggle está desactivado, se asignará el color rojo claro.

Una vez definido el Script es necesario asignarlo a un objeto e inicializar las variables públicas. La Figura 3 presenta la vista Inspector de un objeto llamado LED con dicho script asociado. En este caso, se han establecido los objetos Toggle y Sphere (definidos) a las variables 'toggle' y 'esfera' del Script respectivamente.

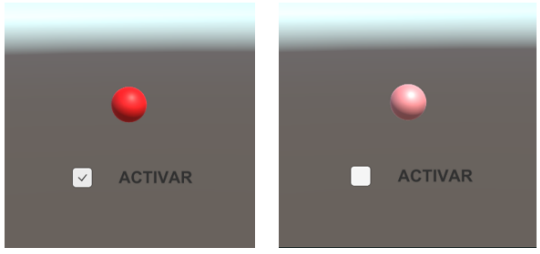

Figura 4: LED en funcionamiento. (a) Activado y (b) Desactivado

Paso 7. Pruebas y ajustes. Probar el modelo digital en el editor de Unity para verificar su funcionamiento y apariencia. Realizar ajustes según sea necesario para corregir posibles errores. La Figura 4 presenta las pruebas realizadas para el LED.

Paso 8. Compilación y distribución. Compilar el modelo digital para la plataforma de destino, ya sea PC, móvil o web.

3. Manipulación Vástago Doble Efecto en Unity.

Este apartado tiene como finalidad seguir la metodología propuesta en el apartado dos a la hora de implementar un ejemplo básico de automatización como es la manipulación de un cilindro de doble efecto.

Paso 1. Conceptualización del modelo digital. Para este caso son necesarios dos finales de carrera del cilindro y el cilindro.

Paso 2 – 5. Creación de activos y configuración de la escena. Para este caso de uso se implementarán dos sensores magnéticos que representarán los finales de carrera del cilindro y el propio cilindro.

El procedimiento para su funcionamiento constará de dos cubos, los cuales representarán los dos sensores del cilindro, junto con el vástago del cilindro (formado por un eje y una base).

La parte visual de sensor se ha seguido el mismo procedimiento que el comentado en el apartado anterior. Para plasmar la parte magnética, se ha hecho uso del elemento cubo, al que se le ha añadido un Rigibody y Box Collider configurado como Trigger. La Figura 5 presenta las vistas con la caracterización de los elementos que forman un sensor magnético.

La Figura 6 presenta la caracterización del cilindro. Un objeto compuesto a su vez por una base y un eje (móvil) que desempeña el papel de pistón. La parte visual, fichero OBJ se dichas partes se ha obtenido de (traceparts, 2024).

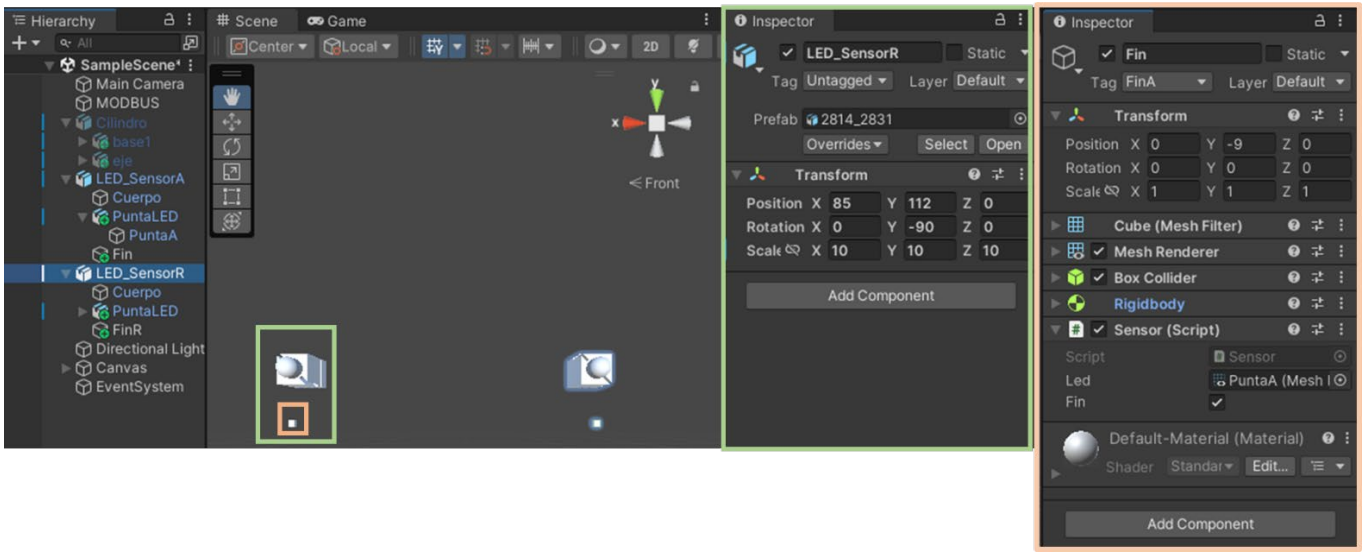

Figura 5: Creación de sensores magnéticos

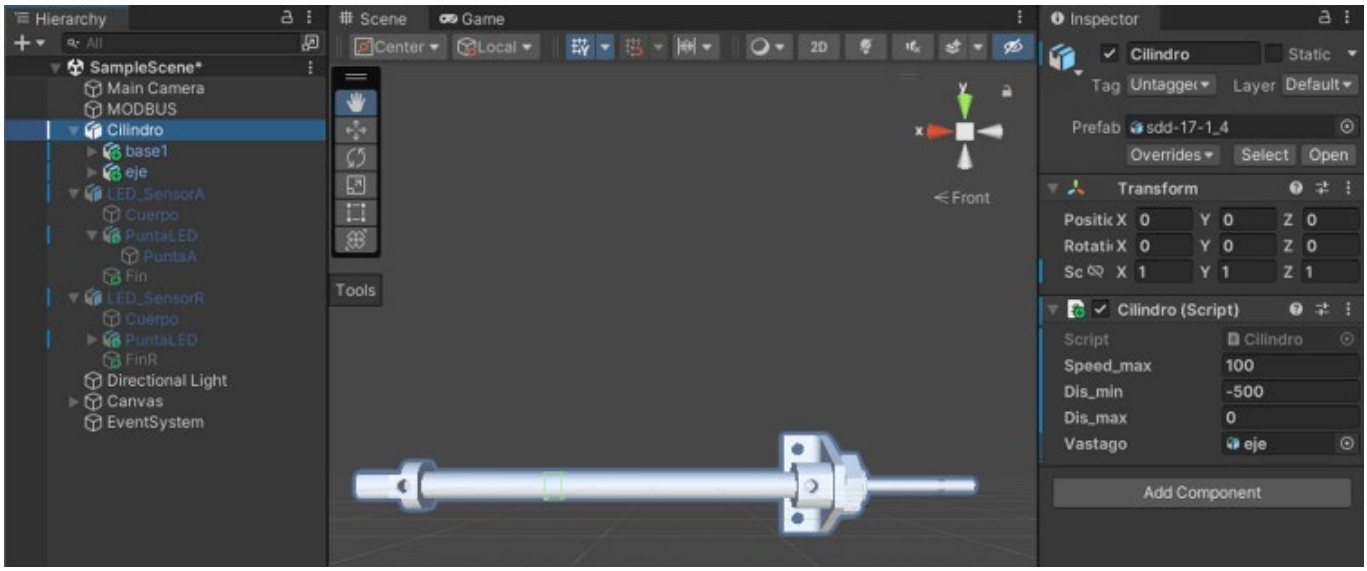

Figura 6: Definición del objeto cilindro

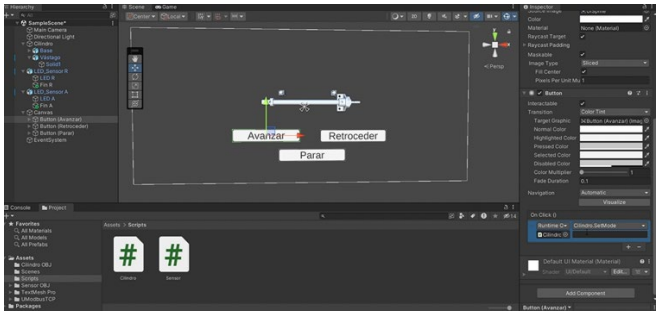

Figura 7: elementos Canvas: botones que permitirán al usuario ordenar expandir, contraer o parar el vástago del cilindro.

Se han añadido tres botones que permitirán interactuar al usuario con el movimiento del cilindro: Avanzar, Retroceder y Parar (véase Figura 7).

Paso 6. Implementación de interactividad. Este paso presenta la codificación de los scripts del sensor magnético y el cilindro. El Script Sensor se encargará de modificar el LED dependiendo de los Colliders (véase Figura 8). Este script se asociará a los dos objetos que ejercen de Trigger para el sensor. Para este script, simplemente se necesitarán declarar las variables que definen los colores necesarios, la variable privada de tipo Renderer para acceder al LED y por último una variable booleana llamada 'Fin' que marcará el estado del LED. Es importante mencionar que '[HideInInspector]' se emplea en el código con el fin de evitar que dichas variables aparezcan en la ventana de Inspector. Este Script consta de tres métodos. En el método *Update()*, solo se necesita que esté constantemente actualizando el color del LED según la variable booleana 'Fin'. Si 'Fin' es true, el color del material se establece en 'colorRojoFuerte'; de lo contrario, se establece en 'colorRojoFlojo'. El método *OnTriggerExit()* se llama cuando otro objeto sale del área de colisión asociada con este objeto. Dentro de este método, la variable booleana 'Fin' se establece en false, lo que indica que el objeto ya no está en el área de colisión. Por último, el método *OnTriggerEnter()* se llama cuando otro objeto entra en el área de colisión asociada con este objeto. Dentro de este método, la variable booleana 'Fin' se establece en true, indicando que el objeto está ahora dentro del área de colisión.

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

Script de Unity (2 referencias de recurso) | 1 referencia
public class Sensor : MonoBehaviour{
    public Renderer led;
    [HideInInspector]public Color colorRojoFuerte = new Color(2.0f, 0.0f, 0.0f);
    [HideInInspector]public Color colorRojoFlojo = new Color(0.4f, 0.0f, 0.0f);
    public bool Fin = false;
    Mensaje de Unity | 1 referencia
    public void Update(){
        led.material.color = Fin ? colorRojoFuerte : colorRojoFlojo;
    }
    Mensaje de Unity | 0 referencias
    private void OnTriggerExit(Collider other){
        Fin = false;
    }
    Mensaje de Unity | 0 referencias
    private void OnTriggerEnter(Collider other){
        Fin = true;
    }
}
```

Figura 8: Script Sensor.

```
public class Cilindro : MonoBehaviour
{
    public float speed_max = 500.5f;
    [HideInInspector] public float speed_actual;
    [HideInInspector] public Vector3 new_pos;
    private Vector3 startingPosition;
    public float dis_min = -100.0f;
    public float dis_max = 100.2f;
    public GameObject vastago;
    Mensaje de Unity | 0 referencias
    void Start(){
        startingPosition = vastago.transform.position;
    }
    Mensaje de Unity | 1 referencia
    public void Update(){
        new_pos = vastago.transform.position + (Vector3.right * speed_actual * Time.deltaTime);
        float clampedx = Mathf.Clamp(new_pos.x,  dis_min,  dis_max);
        new_pos.x = clampedx;
        vastago.transform.position = new_pos;
        if (dis_max < new_pos.x | dis_min > new_pos.x ){
            speed_actual = 0;
        }
    }
    3 referencias
    public void SetMode(string mode){
        switch (mode){
            case "Avanzar":
                speed_actual = speed_max;
                break;

            case "Retroceder":
                speed_actual = -speed_max;
                break;

            case "Parar":
                speed_actual = 0;
                break;

            default:
                speed_actual = 0;
                break;
        }
    }
}
```

Figura 9: Script comportamiento del cilindro

Por otro lado, para se ha creado el script 'Cilindro' que será el responsable de gestionar el movimiento del vástago dependiendo de unos botones de activación. Este script se

asociará al objeto donde se encuentre el cilindro entero. La Figura 9 presenta la estructura que seguirá dicho script. Para la declaración de variables, hay una variable que recoge la velocidad máxima con la que se moverá el vástago y una distancia mínima para tener el movimiento del vástago limitado en ambas direcciones. Se define como GameObject el vástago al que se le vinculará el movimiento. Este script tiene 3 métodos: los clásicos Start() y Update() y setMode().

El método Start() se ejecuta una sóla vez e inicializa la posición inicial del vástago. El método Update() actualiza la posición actual del vástago. Dicho método contempla que no se expanda más de su distancia máxima ni tampoco menos de su distancia mínima.

Finalmente, el método SetMode toma un parámetro de entrada, una cadena de caracteres llamada ‘mode’. Esta variable representa los estados en los que puede estar el vástago que son tres: Expandiéndose, contrayéndose o parado. Por ello, se utiliza una estructura de control switch para determinar qué acción tomar en función del valor del parámetro ‘mode’. Si ‘mode’ es igual a ‘Avanzar’, la variable ‘speed_actual’ se establece en el valor máximo de velocidad. Si ‘mode’ es igual a ‘Retroceder’, ‘speed_actual’ se establece en el valor negativo del máximo de velocidad. Si ‘mode’ es igual a ‘Parar’, ‘speed_actual’ se establece en 0. En caso de que ‘mode’ no coincida con ninguno de estos casos, ‘speed_actual’ también se establece en 0.

Una vez definido los scripts para finalizar este punto es necesario agregar a los botones de UI la función correspondiente según se detalla en la sección ‘Button’ del propio elemento. Por ejemplo, la en la parte inferior derecha de la Figura 7, se ha asociado el botón Avanzar con el objeto ‘Cilindro’ y se ha seleccionado la función SetMode de su script donde se le pasa como parámetro ‘Avanzar’. Siguiendo esta misma filosofía se asociarían los botones ‘Retroceder’ y ‘Parar’.

Paso 7. Pruebas y ajustes: En el siguiente enlace está disponible un video que muestra dicho caso de uso al completo (GRAV, 2024).

4. Conclusiones

El modelado 3D es imprescindible para poder llevar a cabo los objetivos de la industria 4.0. Este trabajo ha analizado cómo hacer uso de la herramienta Unity para modelar sistemas de automatización industrial. Se ha presentado una metodología con las pautas a seguir ante cualquier modelado y, además, se ha presentado cómo añadir dos objetos básicos pero esenciales en todos sistemas de automatización industrial y que, a priori, Unity no incluye, como son el sensor magnético y el cilindro de doble efecto. Dicha metodología se ha probado en el ámbito docente, concretamente con TFG de diferentes alumnos donde han realizado modelos digitales de estaciones disponibles en el laboratorio. El hecho de disponer de dichas pautas y elementos les ha permitido centrarse exclusivamente en las particularidades de sus estaciones. Por otro lado, es importante remarcar que todo lo presentado es extensible con mínimo esfuerzo para poder realizar sombras y gemelos digitales donde existe una comunicación unidireccional y bidireccional respectivamente con la planta física. Esto se debe a que Unity ofrece la posibilidad de trabajar con buses industriales como Modbus y también con protocolos IoT como OPC-UA. Por ello, en trabajos futuros se extenderá la metodología propuesta con objeto de poder definir no sólo modelos digitales, sino también sombras y gemelos digitales donde es imprescindible la interacción con el mundo físico de forma unidireccional y bidireccional respectivamente.

Agradecimientos

Este trabajo ha sido realizado parcialmente gracias al apoyo Ministerio de Ciencia, Innovación y Universidades a través del Programa "Salvador de Madariaga" 2019, PRX22/00371 y el proyecto PID2019-110291RB-I00.

Referencias

Akpan, I.J., Offodile, O.F., 2024. The Role of Virtual Reality Simulation in Manufacturing in Industry 4.0. Systems 12, 26. https://doi.org/10.3390/systems12010026

Brito, A., 2024. Blender 4.0: Precise Modeling for Architecture, Engineering, and 3D Printing.

Coutinho, C., 2022. Unity® Virtual Reality Development with VRTK4: A No-Coding Approach to Developing Immersive VR Experiences, Games, & Apps. Berkeley, CA.

de Paula Ferreira, W., Armellini, F., De Santa-Eulalia, L.A., 2020. Simulation in industry 4.0: A state-of-the-art review. Comput. Ind. Eng. 149, 106868. https://doi.org/10.1016/j.cie.2020.106868

de Paula Ferreira, W., Armellini, F., de Santa-Eulalia, L.A., Rebolledo, C., 2021. Modelling and Simulation in Industry 4.0, in: Dingli, A., Haddod, F., Klüver, C. (Eds.), Artificial Intelligence in Industry 4.0: A Collection of Innovative Research Case-Studies That Are Reworking the Way We Look at Industry 4.0 Thanks to Artificial Intelligence. Springer International Publishing, Cham, pp. 57–72. https://doi.org/10.1007/978-3-030-61045-6_5

Dintén, R., Martínez, P.L., Zorrilla, M., 2021. Arquitectura de referencia para el diseño y desarrollo de aplicaciones para la Industria 4.0. Rev. Iberoam. Automática E Informática Ind. 18, 300–311. https://doi.org/10.4995/riai.2021.14532

Factory I/O – Next-Gen PLC Training [WWW Document], 2024. URL https://factoryio.com/ (accessed 5.31.24).

Folgado, F.J., Calderón, D., González, I., Calderón, A.J., 2024. Review of Industry 4.0 from the Perspective of Automation and Supervision Systems: Definitions, Architectures and Recent Trends. Electronics 13, 782. https://doi.org/10.3390/electronics13040782

GRAV, 2024. Modelo Digital - YouTube [WWW Document]. URL https://www.youtube.com/watch?v=oQtQ6Ma9ixM (accessed 6.4.24).

ITAINNOVA, 2015. Análisis: Motores gráficos y su aplicación en la industria.

Plataforma de desarrollo en tiempo real de Unity [WWW Document], 2023. . Unity. URL https://unity.com/ (accessed 5.31.24).

Siemens NX y Simit, 2024. Virtual commissioning of machines with S7-PLCSIM Advanced, SIMIT and NX MCD - ID: 109758943 - Industry Support Siemens [WWW Document]. URL https://support.industry.siemens.com/cs/document/109758943/virtual-commissioning-of-machines-with-s7-plcsim-advanced-simit-and-nx-mcd?dti=0&lc=en-TN (accessed 5.31.24).

Simumatik | State of the Art Emulation [WWW Document], n.d. . Simumatik. URL https://simumatik.com/ (accessed 5.31.24).

SketchUp Blog [WWW Document], 2024. URL https://blog.sketchup.com (accessed 5.31.24).

traceparts, 2024. Biblioteca de archivos 3D [WWW Document]. URL https://www.traceparts.com/es/ (accessed 5.31.24).

Vargas, H., Heradio, R., Donoso, M., Farias, G., 2023. Teaching automation with Factory I/O under a competency-based curriculum. Multimed. Tools Appl. 82, 19221–19246. https://doi.org/10.1007/s11042-022-14047-9