UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de un Digital Twin para el proceso de fabricación de una celda de cristal líquido con capa de alineamiento.

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Villanueva Latorre, Pablo

Tutor/a: Bachiller Martin, Maria Carmen

Cotutor/a: Monzó Ferrer, José María

Cotutor/a: Rey Solaz, Beatriz

CURSO ACADÉMICO: 2023/2024

Resumen

El uso de Digital Twins o "gemelos digitales" se ha ido extendiendo para la docencia en la ingeniería. Los Digital Twins son modelos muy detallados de sistemas complejos en realidad virtual. En un entorno virtualizado, los Digital Twins se utilizan para el aprendizaje de procesos complejos, que requieren un alto grado de especialización. Los Digital Twins permiten extender la docencia sin necesidad de tener los modelos reales, que pueden ser muy costosos y dándole al estudiante la posibilidad de equivocarse sin riesgo de estropear, desperdiciar o romper piezas.

En este trabajo de fin de grado se va a crear un Digital Twin para el proceso de fabricación de una celda de cristal líquido con capa de alineamiento. Este proceso es bastante laborioso y tiene una serie de pasos que hay que seguir de forma rigurosa, además los materiales que se ven involucrados son caros y difíciles de conseguir. Por eso que el estudiante pueda aprender el proceso con seguridad en un entorno digital permitirá que cuando se enfrente a él en el laboratorio lo pueda desarrollar de una forma más eficiente.

Para el desarrollo del entorno digital se va a utilizar el paquete de programación Unity, basado en C++ y Python. Se prevé desarrollar una aplicación para uso tanto con gafas Meta como en web.

Palabras Clave: Digital Twin, Unity, realidad virtual, C++, Python, cristal líquido, celda, capa de alineamiento.

Abstract

The use of Digital Twins has been spreading in engineering teaching. Digital Twins are highly detailed models of complex systems in virtual reality. In a virtualized environment, Digital Twins are used for learning complex processes, which require a high degree of specialization. Digital Twins allow teaching to be extended without the need to have real models, which can be very expensive, and allow the student to make mistakes without the risk of damaging, wasting, or breaking parts.

In this final degree project, a Digital Twin will be created for the manufacturing process of a liquid crystal cell with an alignment layer. This process is quite laborious and has a series of steps that must be followed rigorously, and the materials involved are expensive and difficult to obtain. Therefore, the fact that the student can learn the process safely in a digital environment will allow them to develop it more efficiently when faced with it in the laboratory.

For the development of the digital environment, the Unity programming package will be used, based on C++ and Python. It is planned to develop an application for use with both Meta glasses and the web.

Key Words: Digital Twin, Unity, virtual reality, C++, Python, liquid crystal, cell, alignment layer.

Resum

L'ús de Digital Twins o "bessons digitals" s'ha anat estenent per a la docència en l'enginyeria. Els Digital Twins són models molt detallats de sistemes complexos en realitat virtual. En un entorn virtualitzat, els Digital Twins s'utilitzen per a l'aprenentatge de processos complexos, que requereixen un alt grau d'especialització. Els Digital Twins permeten estendre la docència sense necessitat de tindre els models reals, que poden ser molt costosos, i donant a l'estudiant la possibilitat d'equivocar-se sense risc d'espanyar, desaprofitar o trencar peces. En aquest treball de final de grau es crearà un Digital Twin per al procés de fabricació d'una cel·la de cristall líquid amb capa d'alineament. Aquest procés és bastant laboriós i té una sèrie de passos que cal seguir de manera rigorosa, a més els materials que s'hi veuen involucrats són cars i difícils d'aconseguir. Per això, que l'estudiant puga aprendre el procés amb seguretat en un entorn digital permetrà que quan s'hi enfronta al laboratori el puga desenvolupar d'una manera més eficient. Per al desenvolupament de l'entorn digital s'utilitzarà el paquet de programació Unity, basat en C++ i Python. Es preveu desenvolupar una aplicació per a ús tant amb ulleres Meta com en web.

Paraules Clau: Digital Twin, Unity, realitat virtual, C++, Python, cristall líquid, cel·la, capa d'alineament.

RESUMEN EJECUTIVO

La memoria del TFG del GTIST debe desarrollar en el texto los siguientes conceptos, debidamente justificados ydiscutidos, centrados en el ámbito de la REALIDAD VIRTUAL

CONCEPT (ABET)	CONCEPTO (traducción)	¿Cumple? (S/N)	¿Dónde? (páginas)
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad		7-8
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)		8-9
1.3. Setting of goals	1.3. Establecimiento de objetivos		8
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)		10-20
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)		21-22
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos		52-56
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)		57

ÍNDICE

ÍNDICE DE FIGURAS

1.Introducción

El concepto de Digital Twin ha surgido como una herramienta revolucionaria en la industria 4.0 a través de la integración de tecnologías digitales avanzadas para la creación de réplicas virtuales de procesos físicos. La implementación de estos gemelos digitales destaca por ser una solución prometedora para la optimización de procesos, reducir costes y mejorar la calidad de cara al producto final, transformando así la forma en la que se conciben y operan los sistemas industriales.

Un Digital Twin permite replicar un proceso, producto o sistema físico de forma virtual permitiendo así su análisis en tiempo real. Esta innovadora tecnología incorpora datos del mundo real con un modelo digital para proporcionar una representación dinámica del estudio realizado.

En ese proyecto se va a explorar, de forma robusta, el desarrollo de un Digital Twin diseñado para el proceso de fabricación de una celda de cristal líquido con una capa de alineación, un componente esencial para el correcto funcionamiento del dispositivo final. Esta capa de alineación es fundamental al ser determinante de cara a la correcta orientación de las moléculas de cristal líquido, impactando directamente en la calidad y eficiencia de la imagen.

Tradicionalmente, el proceso de fabricación de estas celdas es de alto coste y relativamente complejo, por lo que requiere un control preciso de diversos factores para no cometer fallos que puedan comprometer el resultado final. Es por esto que la introducción de un Digital Twin puede generar un cambio importante en la realización de procesos complejos similares a este, permitiendo la simulación, monitorización y optimización de cada etapa.

Este trabajo de fin de grado se va a centrar en el desarrollo de un Digital Twin para la fabricación de un cristal líquido con una capa de alineación, comenzando desde un estudio del proceso de creación de estos cristales líquidos y de los Digital Twin, para luego continuar con el diseño y modelado a través del programa Unity de las diferentes máquinas y utensilios necesarios para la creación del cristal líquido.

1.1. Objetivos

El principal objetivo de este proyecto es desarrollar un Digital Twin capaz de replicar el proceso de fabricación de una celda de cristal líquido con capa de alineación, realizado en realidad virtual con el uso de las gafas Meta Quest 2, gafas de realidad virtual para realizar infinidad de ideas que a cualquier programador le surja.

Con el uso de la realidad virtual se pretende facilitar el aprendizaje de un trabajo que en la actualidad es de un elevado coste y complejidad, permitiendo posibles errores durante el camino para asi aprender de ellos y evitarlos a la hora de la verdad.

El proyecto se realizará en torno al programa de diseño Unity junto con el paquete de programación de Python a través de Visual Code. Se comenzará con el diseño de los diversos objetos a realizar para su posterior modificación para que realicen una serie de eventos hasta llegar a la creación del cristal líquido.

1.2. Motivación

Me considero una persona a la que desde pequeño le han apasionado los videojuegos y todo lo que les rodea. Desde hace años que disfruto de multitud de videojuegos, generando una pasión como ninguna otra y que ha hecho que, a la hora de elegir un trabajo de final de carrera, la elección fuese relativamente fácil.

El hecho de haber disfrutado durante tanto tiempo de estos ha hecho que empezase a generarse en mí una intriga sobre cómo se diseñaban todos ellos. Esta oportunidad representa para mí un desafío para dar finalizada mi carrera, y qué mejor forma que hacerlo a través de algo que se asimila de la menor manera posible a una de mis pasiones desde pequeño.

Considero que este proyecto me va a permitir expandir mis conocimientos en programación, realidad virtual e inteligencia artificial, tres áreas que cada vez van ganando más popularidad en el mundo laboral.

A todo esto, sumarle el hecho de poder realizar un trabajo sobre una tecnología relativamente nueva como lo es el Digital Twin, una novedad en la industria que a medida que avancen los años va a ser escuchada cada vez más.

En cuanto a motivación a nivel laboral y en el ámbito de la docencia e investigación, este proyecto abre las puertas a infinidad de procesos que mostrar y enseñar a aquel que lo desee y busque adentrarse en el mundo de la realidad virtual, por lo que formar parte de ello, aunque sea de una manera lo más mínima posible, ya es algo que va a quedarse marcado de por vida.

1.3. Estructura

Como ya se ha comentado, el presente trabajo tiene como objetivo el desarrollo de un Digital Twin para el proceso de fabricación de una celda de cristal líquido con una capa de alineación. Para ello, se va a seguir un determinado orden que va a permitir un correcto entendimiento de todas las fases a realizar.

El primer apartado va a incluir dos estudios, el primero sobre los Digital Twin para conocer sus inicios, sus diferentes aplicaciones y así obtener un buen nivel de conocimiento sobre ellos. Una vez completado este, se continuará con el estudio centrado en la creación de los cristales líquidos con capa de alineación, siguiendo el esquema que se va a realizar para el estudio destinado a los Digital Twin.

Una vez completados ambos estudios se dispondrá de un buen nivel de información a nivel teórico que permitirá explicar de una forma correcta dos de los tres bloques que forman este proyecto.

El tercer bloque que conforma el trabajo gira entorno al programa de diseño Unity. Este programa es el encargado de diseñar infinidad de videojuegos, programas, etc. En cuanto a este, se explicará de manera breve sus diversas aplicaciones y funcionamiento para luego poder comentar el proceso que se ha llevado a cabo para el diseño de los productos.

Con esto se cerraría el bloque destinado al estado del arte y se continuaría con el bloque encargado de todo el proceso de creación en realidad virtual de los objetos necesarios para llevar a cabo el Digital Twin.

Este contendrá dos apartados, uno destinado a mostrar el diseño en realidad virtual 3D de una serie de objetos necesarios para crear el cristal líquido con la capa de alineación, y en el que se comentarán los diferentes pasos llevados a cabo en Unity para crearlos. El segundo apartado girará en torno al proceso dinámico del trabajo, es decir, a modificar los objetos ya creados para darles vida, permitirles realizar una serie de tareas en la realidad virtual para simular el proceso de creación de las celdas.

Con esto finalizaría el groso del trabajo, a falta de incluir las diversas conclusiones que se han ido obteniendo durante el proceso y líneas futuras a las que poder adaptar esta nueva tecnología.

2.Estado del arte

¿Cómo puede la innovadora tecnología de los Digital Twin revolucionar el proceso de creación y fabricación de procesos complejos y costosos tales como este? En una época caracterizada por los avances tecnológicos y una creciente demanda de eficiencia y precisión, la llegada al mercado de los gemelos digitales se presenta como una solución prometedora y tremendamente fiable, transformando la manera en que las industrias gestionan y optimizan sus operaciones.

Este capítulo del trabajo de fin de grado tiene como objetivo principal explorar en profundidad los Digital Twins y el proceso de fabricación de cristales líquidos, analizando ambos desde sus inicios hasta la actualidad y sus diferentes aplicaciones.

2.1. Digital Twin

Como ya se ha comentado, los gemelos digitales son una de las tecnologías emergentes en la actual época de la industria 4.0. Este innovador concepto combina modelado, simulación y monitorización, tres habilidades que están cambiando la forma en la que las industrias diseñan, fabrican y operan.

La capacidad de replicar de forma digital un objeto o proceso físico ha abierto nuevas posibilidades para la optimización operativa, reducción de costes y una mejor toma de decisiones.

Este bloque del trabajo tiene como principal objetivo explorar a fondo el concepto de Digital Twin, sus diversas aplicaciones, beneficios y desafíos que se han encontrado sus creadores desde sus inicios hasta la actualidad, así como el impacto que pretende generar de cara al futuro a nivel laboral.

2.1.1. Definición y concepto

Un Digital Twin se puede definir como una réplica en formato digital ya sea de un objeto, sistema o proceso físico que usa datos en tiempo real para simular determinados comportamientos durante su ciclo de vida. Esta definición conlleva diversas capas de complejidad, desde el modelado del objeto a la integración de datos en tiempo real y su posterior capacidad para simular aquellos procesos que sean necesarios. A diferencia de un modelo CAD, un Digital Twin está “conectado” a la parte física, permitiéndole reaccionar y adaptarse a cualquier cambio.

En relación con la gestión del ciclo de vida del producto, Michael Grieves propuso este término en 2003 [1]. Para mejorar el diseño, la fabricación y el mantenimiento de un producto específico, se creó un Digital Twin.

El poder de la Inteligencia Artificial (IA), el Internet de las cosas (IoT) y la nube han contribuido a la evolución de este concepto innovador desde su origen.

El avance del Digital Twin se ha acelerado gracias a estas tres. La IA es la tecnología que analiza los datos para su correcto uso y procesamiento; el gemelo permite recopilar datos en tiempo real para su posterior procesamiento; estos datos serán almacenados en la nube.

A diferencia de una simulación estática, los gemelos digitales están en constante comunicación con el objeto o proceso, permitiendo así una continua actualización si fuera necesaria. A esto se le añade que este no solo replica la geometría del objeto, sino que es también capaz de replicar su comportamiento dinámico e interacción.

2.1.2. Arquitectura y componentes

La creación de un Digital Twin empieza con el modelo de un objeto o proceso físico que incluye su geometría, propiedades físicas, parte dinámica y las diferentes interacciones a realizar con otros sistemas.

El modelado físico es una de las bases fundamentales de un Digital Twin, al proporcionar el marco para la simulación y análisis de datos. A medida que se van recopilando datos, el Digital Twin se ajusta para mostrar cualquier posible cambio en el objeto o proceso, representándolos así de forma precisa y actualizada [2], [3].

Dos componentes clave dentro de los Digital Twin son la interoperabilidad y la conectividad. La interoperabilidad es un aspecto crítico debido a que los modelos empleados deben integrarse con otros sistemas y plataformas en una misma organización [4], [5].

La conectividad, por otro lado, permite el acceso remoto a los datos empleados en los gemelos que se encuentran almacenados en la nube. Esto es relativamente importante en aquellas industrias que operan a nivel global, facilitando la colaboración entre los diferentes equipos y mejorando la toma de decisiones ante situaciones críticas.

Los últimos pilares que conforman este bloque son la IA y simulación. La IA es la encargada de analizar grandes volúmenes de datos, identificar patrones y aliviar trabajos costosos en cuanto a tiempo se refiere, permitiendo así tomas de decisiones rápidas y precisas.

La simulación será la encargada de probar escenarios diferentes en el mundo digital sin generar cambios en el proceso u objeto físico. Este proceso es fundamental para una

correcta planificación del mantenimiento, optimización de operaciones y respuestas rápidas ante posibles emergencias.

2.1.3. Tipos de Digital Twin

El actual trabajo está compuesto por 2 tipos de gemelos digitales, siendo estos tipos de Digital Twin los siguientes:

- Digital Twin de producto: Encargado de representar digitalmente un producto específico durante su ciclo de vida, comenzando con el diseño inicial hasta su fabricación final. Este tipo de proceso permite la optimización del diseño del producto, así como la prevención de posibles fallos antes de enviar el producto a su proceso de producción.

  En la industria automotriz, por ejemplo, estos gemelos digitales se emplean para simular el rendimiento de los vehículos ante diferentes condiciones de operación, permitiendo a los ingenieros observar y corregir posibles problemas en el diseño antes de enviarlo a la cadena de producción.

- Digital Twin de producción: Se aplica en el apartado dedicado a la simulación y optimización de los procesos de manufacturación, encargado de modelar las diferentes líneas de producción, operaciones de las máquinas y los flujos de trabajo con el fin de identificar ineficiencias y mejorar la productividad.

  Este tipo de gemelo permite a los gestores de planta monitorear las máquinas en tiempo real y ajustar la producción para cumplir con la demanda esperada, mejorando la eficiencia operativa y reduciendo costes de operación y tiempos de producción e inactividad.

De ambos tipos, el Digital Twin de producto sería el que gira en torno a este proyecto al ser el encargado de construir un proceso real en realidad virtual, con la capacidad de mostrar los diferentes pasos llevados a cabo, siendo relativamente útil a nivel de docencia.

2.1.4. Aplicaciones en la industria

A medida que los Digital Twins han ido desarrollándose, las principales industrias a nivel mundial no han dudado en aplicar sus diversos usos en sus procesos diarios. Destacan 3 industrias.

- Manufactura: Es uno de los sectores donde los gemelos digitales han encontrado un mayor uso. Las empresas lo emplean para mejorar la eficiencia de producción, reducir costes y mejorar la calidad final del producto a través de la simulación de líneas de producción y procesos de manufacturación.

  Siemens, por ejemplo, hace uso de los Digital Twins para simular y optimizar la producción en sus fábricas inteligentes, reduciendo significativamente los tiempos de ciclo y adaptándose a los cambios en la demanda del mercado [6].

- Energía y Servicios Públicos: En ambos sectores, el uso de Digital Twin permite a las empresas gestionar, simular y optimizar procesos de producción principales, situándolos ante diferentes condiciones para observar la respuesta que se obtendría en una situación real.

  General Electric ha sido capaz de desarrollar un Digital Twin para sus turbinas de gas, habilitándoles a los operadores la capacidad de monitorear el rendimiento de estas en tiempo real, prever mantenimientos y ajustar las turbinas para maximizar la eficiencia energética [7].

- Transporte y Logística: En este sector el uso de los Digital Twins gira en torno al diseño de vehículos, gestión de flotas y una mejor eficiencia del transporte. A través de la simulación de rutas y la optimización del uso de combustible, los gemelos digitales son capaces de reducir costes operativos y mejorar la sostenibilidad del apartado logístico.

  DHL ha implementado esta tecnología para optimizar la gestión de su cadena de suministro, permitiéndole simular el flujo de mercancías y ajustar rutas de transporte en tiempo real para reducir retrasos y costes de transporte [8].

2.1.5. Beneficios

Esta tecnología innovadora tiene la capacidad de mejorar la toma de decisiones de las organizaciones, como ya se ha observado. Al brindar una copia digital exacta, las compañías pueden examinar numerosos escenarios y evaluar las diversas decisiones tomadas después de ellos antes de llevarlas a cabo en la realidad, lo que reduce el peligro de errores y los gastos innecesarios.

La optimización de estos es mucho más fácil cuando se tiene un objeto o proceso manual completo en formato digital. La capacidad de realizar todo tipo de pruebas en el entorno digital permite a las empresas identificar ineficiencias y cuellos de botella para después ser capaces de solucionarlos en la realidad.

Los costos de la empresa se ven directamente afectados por ambas situaciones. La habilidad de detectar posibles errores en la cadena de suministro o en la creación y desarrollo del producto permite a los líderes de la empresa tomar decisiones que les permitirán, entre otras cosas, ahorrar costos que no se podrían haber hecho sin la ayuda de los Digital Twins.

2.1.6. Desafíos y Limitaciones

Toda nueva tecnología viene con sus puntos negativos, y los Digital Twins no iban a ser una excepción. A pesar de contar con beneficios relativamente importantes, su implementación presenta ciertos desafíos.

Uno de los principales desafíos es la complejidad y el coste inicial asociado al desarrollo de esta innovación. El proceso de creación de un Digital Twin conlleva una inversión inicial significativa en cuanto a infraestructura, software y personal capacitado se refiere, generando asi una barrera de entrada para cualquier empresa que quiera incorporarlo bajo su equipo.

Este proceso de integración a los sistemas ya existentes de cada organización mundial es uno de los procesos más complejos. Estos sistemas suelen ser relativamente complejos, además de estar compuestos por plataformas y aplicaciones que puede que no sean compatibles con aquellos que el Digital Twin trae de por sí. Sin una correcta integración y entendimiento entre los sistemas de la empresa y del Digital Twin un correcto funcionamiento es impensable.

Otro aspecto importante es la privacidad y seguridad de los datos. Al depender de la recopilación y análisis de numerosas cantidades de datos, es crucial para la empresa asegurarse de que los datos están protegidos de accesos no autorizados y ciberataques.

2.1.7. Tendencias futuras

El futuro de esta tecnología se perfila como una de las más disruptivas en cuanto a transformación digital a nivel industrial se refiere. La creciente demanda por la integración de la inteligencia artificial a nivel empresarial y su correspondiente aprendizaje y entendimiento posiciona a los Digital Twins en la cúspide tecnológica [4]. A través de la combinación IA – Digital Twin, el proceso de replicación y simulación de sistemas físicos está cobrando un nuevo sentido que, tarde o temprano, estará más que asentado en todas las empresas.

La expansión de los Digital Twins hacia ecosistemas más extensos y conectados es otra tendencia que se va a fortalecer gradualmente. Esto se refiere al proceso de unir múltiples Digital Twins de varios procesos y sistemas para crear uno único que pueda optimizar las operaciones a nivel mundial. En sectores como la energía y la manufactura, donde la eficacia depende de la interconexión de sistemas, este caso será particularmente significativo.

Los Digital Twins son una de las piedras angulares de la Industria 4.0 o cuarta revolución industrial, caracterizada por la digitalización masiva y la automatización de procesos industriales. Con esta herramienta las empresas van a ser capaces de crear fábricas inteligentes, con sistemas físicos y digitales integrados y perfectamente entendidos entre sí.

A medida que las empresas vayan adoptando esta tecnología, es más probable que se visualice una aceleración a nivel industrial en el apartado de la transformación digital, abandonando aquellas tareas y bloques más tradicionales hacia un futuro más tecnológico y automatizado.

2.2. Cristal líquido

El cristal líquido es un material esencial en la fabricación de pantallas y dispositivos optoelectrónicos que son los encargados de transformar energía eléctrica en luminosa, donde la incorporación de capas de alineación en estos cristales es fundamental para un correcto rendimiento y calidad en los diferentes dispositivos.

El proceso de fabricación de cristales líquidos con capa de alineación, los materiales utilizados y cómo se pueden utilizar, así como los desafíos que surgen durante el proceso y las tendencias futuras de esta tecnología, serán abordados en este capítulo del trabajo.

El concepto de cristal líquido se descubrió durante el siglo XIX, pero no fue hasta el siglo XX que no se comenzó con su aplicación en dispositivos tecnológicos [9]. La capacidad de estos cristales para cambiar su orientación a través de campos eléctricos ha hecho que se conviertan en piezas indispensables en la industria electrónica, destacando especialmente en la fabricación de pantallas.

Son materiales que muestran propiedades intermedias entre las fases sólida y liquida, característica que habilita la capacidad de manipular sus propiedades ópticas con el uso de campos eléctricos [10]. Por el otro lado, la capa de alineación asegura que las moléculas de cristal líquido se orienten de la manera correcta, orientación necesaria para asegurar el correcto funcionamiento de las pantallas LCD [11].

2.2.1. Proceso de fabricación

El desarrollo de las células de cristal líquido requiere un estudio exhaustivo con el objetivo de obtener estructuras que permitan la caracterización de las propiedades dieléctricas con la mayor precisión posible.

Los materiales empleados para formar las células son: vidrios de 0,410 mm de grosor, separadores Mylar de 0,1 mm y un pegamento óptico Norland UV Sealant 91, pegamento diseñado especialmente para el sellado de dispositivos de cristal líquido. En la figura 1 se puede observar el esquema general de la célula de cristal líquido, siendo los vidrios la parte azul, en gris la capa de alineación, en verde los separadores y en blanco el agujero que contendrá la muestra de cristal líquido [9].

Figura 1. Esquema de la célula de cristal líquido

El proceso de construcción de las células sigue el siguiente orden [9]:

1. Limpieza del vidrio: Ambos vidrios deben estar lo más limpios posibles, por lo que se realizara un proceso de limpieza basado en dos etapas:

    - Primero se frotará el vidrio con un paño bañado en solvente orgánico, alcohol isopropílico o acetona, eliminando así grasas y partículas más grandes. Aunque el vidrio parezca medianamente limpio, siguen existiendo grandes cantidades de partículas microscópicas que podrían dañar la homogeneidad de la capa de alineación y posiblemente la separación entre los vidrios de la célula.

    - Una vez finalizado el primer paso, se comienza una segunda fase de lavado en un tanque de ultrasonido donde los vidrios se sumergen en una solución detergente especial para la limpieza ultrasónica, dejándolos en el taque durante 15 minutos. Tras esto, se secan y almacenan los vidrios en recipientes herméticos para evitar su contaminación.

2. Superficie de alineación: Para orientar de forma correcta estas moléculas se crea una superficie de alineación en un lado de cada vidrio de la célula, anclándola a través de tres pasos:

    - Deposición de la poliamida: Realizada a través de un centrifugado, donde el vidrio se deposita en una centrifugadora (Spin Coater) y se cubre con poliamida para su posterior centrifugado durante 10 segundos aproximadamente para obtener una superficie de unos 100 nm aproximadamente.

    - Curado del polímero: Una vez finalizada la deposición se curan las muestras a alta temperatura para conseguir la polimerización de la poliamida. Este curado se realiza con el uso de una placa calefactora en dos procesos, un primero a 80ºC durante unos 10 minutos y posteriormente una segunda fase a 210ºC durante 45 minutos.

    - Surcos microscópicos: Este proceso se realiza mediante un frotado en terciopelo de seda natural para evitar residuos en los vidrios. En este, la

parte del cristal cubierta con la poliamida se frota sobre el terciopelo en una única forma y dirección, ya que es lo que determinara la inclinación inicial de las moléculas.

3. Montaje de las células: Para construir la célula se adhieren 4 separadores Mylar en cada una de las cuatro esquinas con el pegamento Norland, permitiendo así una correcta separación entre ambos cristales. Los cristales se pegarán el uno con el otro como se ha podido ver en la Figura 1 para permitir el posterior llenado, aplicando presión para consolidar la estructura y minimizar el grosor extra añadido por el pegamento. Una vez pegado, el curado se hará en un horno ultravioleta.

4. Llenado de la célula: Una vez secada, con el uso de una pipeta se procede al llenado de la cavidad a través del lecho que se ha dejado libre, extendiéndose así por toda la célula. Una vez la célula ha sido llenada se procederá a su sellado con pegamento óptico para evitar derrames futuros. Al usar vidrios transparentes existe la posibilidad de verificar la correcta alineación de las moléculas, asegurándose de que se ha realizado de forma correcta todo el proceso de fabricación.

Este proceso de fabricación es crucial para un correcto funcionamiento del cristal líquido, es por ello que requiere un meticuloso plan que se ha de seguir al pie de la letra, evitando posibles errores que puedan comprometer el resultado final. Es por esto que, a través del uso de los Digital Twins, se va a llevar a cabo una réplica virtual del proceso completo, desde la inicial limpieza de los vidrios hasta el sellado e introducción del cristal líquido.

2.2.2. Aplicaciones del cristal líquido con capa de alineación

Los cristales líquidos con capa de alineación son componentes clave en una gran variedad de aplicaciones tecnológicas debido a su capacidad para controlar la orientación de las moléculas del cristal líquido y las propiedades ópticas correspondientes. En cuanto a las aplicaciones más destacables de esta tecnología:

- Pantallas de cristal líquido (LCD): Son la aplicación más común de esta tecnología, donde la calidad de la imagen de estas depende principalmente de la precisión de la alineación de las moléculas. La capa de alineación es la encargada de guiar la orientación de las moléculas de cristal líquido para asegurar efectos como contrastes óptimos o colores vibrantes. Empresas como Samsung o LG ya

están implementando estas técnicas innovadoras para obtener una mejor resolución en sus pantallas LCD de última generación [12], [13].

- Dispositivos optoelectrónicos: En estos, la capacidad para ajustar la orientación de las moléculas permite cambiar las propiedades de la luz a través de ellos, característica crucial en aplicaciones que necesitan de un control riguroso de polarización o intensidad de la luz. La empresa Sharp Corporation ha aprovechado esta tecnología para fabricación moduladores de luz, permitiéndoles desarrollar productos altamente precisos para esta industria [14].

La aplicación del cristal líquido gira en torno a la correcta alineación de las moléculas de este tal y como se ha podido comentar en los dos casos anteriores donde el ajuste de estas conlleva una serie de funciones diferentes para cumplir con una serie de objetivos.

2.2.3. Beneficios

El uso de estas celdas ofrece beneficios significativos que impactan en el ámbito de la docencia tanto a nivel de comprensión teórica como en el desarrollo de habilidades prácticas. Estas celdas ofrecen una comprensión visual que permiten visualizar como la orientación de las moléculas de cristal líquido tienen un efecto importante en sus propiedades ópticas y eléctricas.

Al participar en el proceso de fabricación de principio a fin permite a los estudiantes adquirir innumerables habilidades, desde el manejo de materiales avanzados a controles de procesos de medición como la microscopia.

Como último aspecto a destacar, el estudio de celdas de cristal líquido fomenta la innovación y pensamiento crítico a través de los diferentes desafíos que uno se puede encontrar durante todo el proceso.

2.2.4. Desafíos y limitaciones

Para asegurar la viabilidad y la calidad de los dispositivos finales, se deben abordar numerosos desafíos económicos y técnicos en la fabricación de cristal líquido. La precisión en la alineación es uno de los mayores desafíos al ser fundamental conseguir que las moléculas de cristal líquido se orienten correctamente, ya que cualquier desviación puede provocar defectos visibles en la imagen o un rendimiento insuficiente del dispositivo. En dispositivos de alta resolución, donde la precisión es fundamental para

mantener la calidad de la imagen, este es un problema especialmente importante. Samsung ha tenido que superar este desafío al desarrollar pantallas de última generación utilizando tecnologías avanzadas de alineación que garantizan una precisión excepcional en la orientación.

La compatibilidad de los materiales es otro gran desafío. No todos los materiales de alineación funcionan correctamente con todos los tipos de cristales, lo que puede resultar en la creación de nuevos polímeros o técnicas que permitan estas compatibilidades. Para garantizar que el cristal líquido conserve sus características dieléctricas y ópticas sin verse afectado por la capa de alineación, la compatibilidad es fundamental. Sharp Corporation, se ha dedicado a la investigación y el desarrollo de materiales que maximicen la compatibilidad, lo que permite un rendimiento estable y duradero en sus dispositivos optoelectrónicos.

Por último, el elevado coste de producción. Al ser un proceso de fabricación con técnicas relativamente avanzadas, el coste elevado puede complicar el proceso de adopción de esta tecnología a diferentes puntos donde se pueda obtener una buena rentabilidad y rendimiento.

2.2.5. Tendencias futuras

El cristal líquido cuenta con un futuro prometedor gracias a diversas áreas de investigación y desarrollo que están abriendo nuevas puertas. Una de estas es el desarrollo de nuevos materiales. La investigación en polímeros avanzados y materiales nanoestructurados está cogiendo cada vez más fuerza, con el objetivo de mejorar la compatibilidad, precisión y reducción de costes.

Además de la introducción de estos nuevos materiales, se pretende encontrar nuevas aplicaciones en campos diferentes a los actuales como por ejemplo la realidad virtual y las pantallas flexibles, ambas tecnologías emergentes con un gran potencial de crecimiento.

A medida que las demandas de calidad, sostenibilidad y eficiencia energética vayan aumentando, el cristal líquido ira cogiendo cada vez más voz en torno al mundo tecnológico del desarrollo de dispositivos electrónicos, jugando un papel clave en cuanto a competencia e innovación en la industria electrónica se refiere.

2.3. Herramientas tecnológicas para el desarrollo del proyecto

El proceso de desarrollo para ser capaces de simular de inicio a fin la fabricación de un cristal líquido con capa de alineación necesita una cuidadosa selección de herramientas de software con la habilidad de modelar y simular cada fase del proceso.

La plataforma elegida para llevar a cabo todo el diseño es Unity. Unity es una aplicación de desarrollo de juegos que ha sido adaptada para suplir diferentes necesidades en el sector industrial, incluyendo la creación de los Digital Twins.

Esta plataforma destaca por su habilidad de visualización 3D en tiempo real, convirtiéndola en la herramienta idónea para el modelado del proceso completo de fabricación, comenzando con el diseño uno a uno de los diferentes objetos a emplear hasta la posterior simulación en realidad virtual de manera interactiva.

Unity cuenta con la capacidad de integrar modelos físicos completos asi como datos en tiempo real, lo que le permite crear un entorno virtual donde visualizar y manipular el proceso de fabricación. Este proceso de fabricación apoyará en el lenguaje de codificación C# para llevar a cabo la parte dinámica del trabajo que concluirá con la realización del proceso completo a través de las gafas de realidad virtual Meta Quest, una de las últimas novedades en el mercado de la realidad virtual.

Con este último bloque concluye el capítulo destinado a explicar tanto los Digital Twins como el proceso de fabricación de cristal líquido con capa de alineación que se va a replicar en la tecnología explicada.

Este capítulo ha tenido como objetivo ofrecer una visión de dos procesos no muy comunes actualmente, pero que están cogiendo cada vez más voz en la industria tecnológica principalmente además de otros sectores donde la aplicación del Digital Twin ofrece innumerables beneficios para suplir problemas de gran complejidad.

3.Desarrollo del Proyecto

Este tercer capítulo del proyecto va dirigido a los dos procesos que ya se han podido comentar previamente, el diseño de los objetos y su posterior dinamismo para poder completar el proceso de fabricación en realidad virtual.

Se va a dividir esta parte del trabajo en dos subcapítulos para poder explicar de la manera más clara posible ambos procesos, al tratarse de una herramienta no muy común en el ambiente de trabajo sobre el que se trabaja y aun sin ser complicada de usar, es necesario una buena guía que permita seguir el proceso al pie de la letra.

3.1. Creación y diseño de los objetos

A continuación, se va a mostrar el paso a paso para crear todos y cada uno de ellos para luego enseñar tanto el objeto en la realidad como su copia en 3D.

Unity cuenta con un menú dedicado a la creación de objetos en 3D con diferentes formas que permiten su modificación para poder replicar de la manera que uno quiera su respectivo diseño. Con este menú comienza el proceso de diseño para cada objeto, donde cada uno de los objetos se construirá a partir de diferentes formas para replicar su aspecto real.

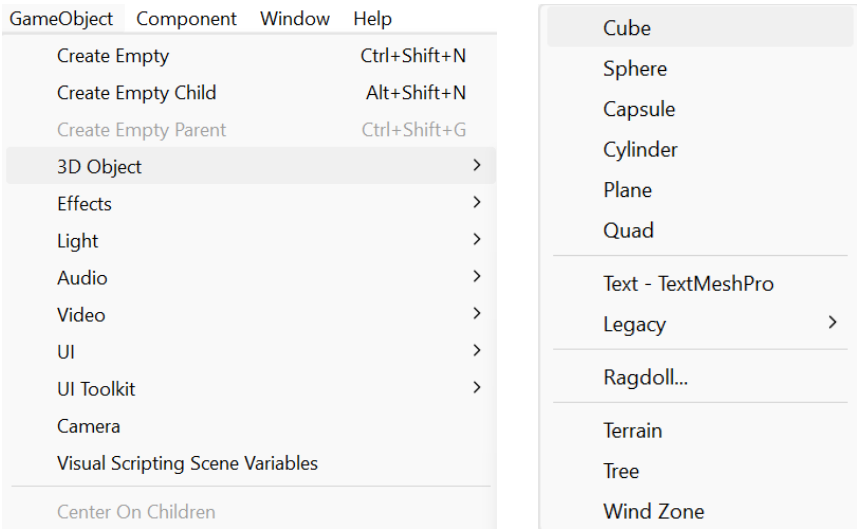

Figura 2. Menú Game Object

Se procede a crear un Game Object vacío que será el encargado de contener los diferentes componentes que construyen el objeto. Una vez creado, se le irán incorporando las figuras que se han usado para la fabricación de este, convirtiéndolo en el Game Object "padre", lo que le va a permitir al programa mover, girar o modificar todo el objeto en conjunto sin que los diferentes objetos que lo componen sufran modificaciones por separado.

Una vez asentado el Game Object "padre" se procederá a comprobar que todo está en su correcta posición, sin huecos entre los diferentes objetos que puedan comprometer el resultado final.

Para esta comprobación, se irán seleccionando cada uno de los objetos "hijos" y a través de las herramientas que incorpora Unity (Move Tool, Rotation Tool, Scale Tool) se ajustarán para obtener visualmente un único objeto. Una vez realizada esta comprobación se dispondrá del objeto en sí y se podrá guardar de manera conjunta en la carpeta "Assets" en lo que se llama un "Prefab", lo que le permitirá hacer uso de cada objeto por separado en diferentes escenas y diferentes usos.

Para aplicar diferentes colores, texturas y materiales a cada uno de los componentes que forman los objetos se crearán diferentes aspectos como los anteriores dentro de las carpetas que Unity incluye y que se irán incorporando a los objetos que se vayan usando. Dentro de la carpeta de "Assets" se creará una nueva carpeta destinada a incorporarlos y así poder asignarlos a sus respectivos componentes. En la Figura 3 se puede observar las diferentes carpetas que componen la carpeta inicial, además de un ejemplo de como se almacena y muestra uno de los colores usados en el trabajo.

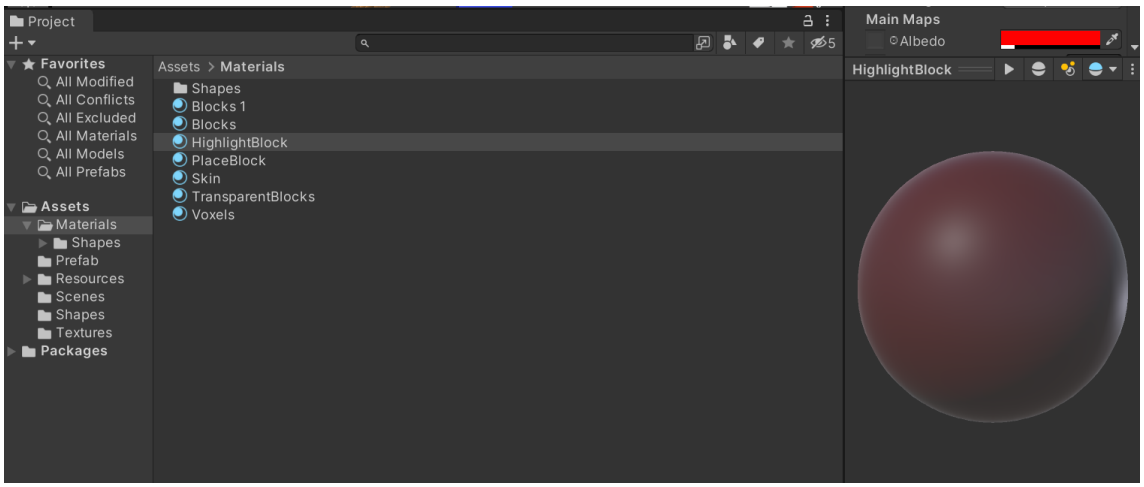

Figura 3. Carpeta "Assets" y Material aplicado en el proceso

Una vez completado el diseño y asignación de color del objeto se dispondrá del diseño final y que se guardará en la carpeta "Prefabs", donde se encuentran todos los objetos acabados y listos para su posterior uso.

A continuación, se van a mostrar diferentes objetos que se han replicado en la realidad virtual para su posterior uso en la parte dinámica del trabajo.

Figura 4. Cristales

Figura 5. Cristales en realidad virtual

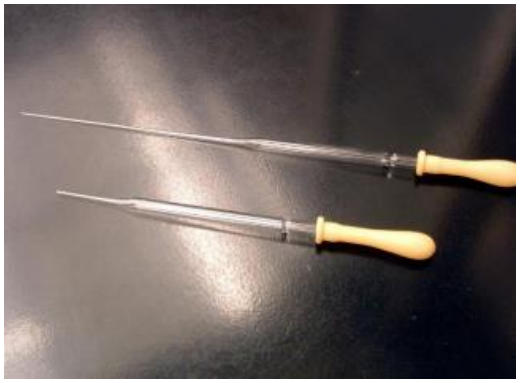

Figura 6. Pipeta

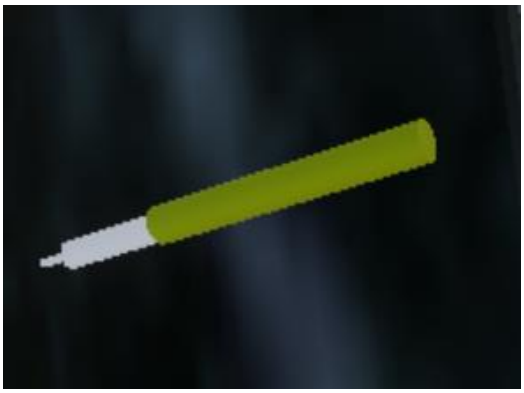

Figura 7. Pipeta en realidad virtual

Figura 8. Pegamento Óptico

Figura 9. Pegamento Óptico en realidad virtual

Estas son unas de las herramientas que se han creado para replicar el proceso en realidad virtual. Con esto concluye este capítulo destinado a mostrar los pasos que se han seguido para crear estos objetos, objetos a los que ahora se les dará vida para poder completar la fabricación del cristal líquido en la realidad virtual.

3.2. Máquina de estados

A continuación, se adjunta una máquina de estados para mostrar el proceso paso a paso de forma más visual y entendible para el usuario.

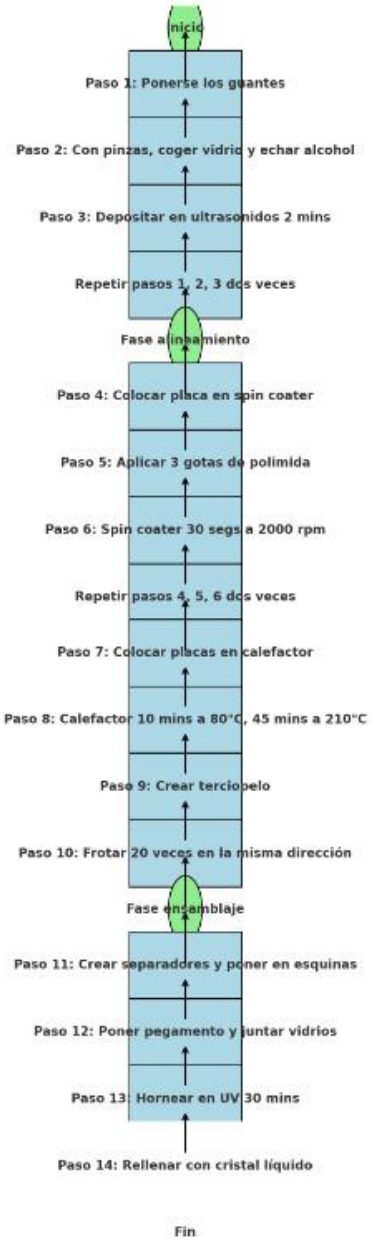

Figura 10. Máquina de estados

3.3. Proceso dinámico del trabajo

En esta segunda parte del capítulo 3 se va a mostrar y comentar el código utilizado para poder completar todo el proceso. Unity cuenta con código diseñado y estructurado al diseño de videojuegos, entendible y accesible para cualquiera con un nivel básico de programación.

Antes de adentrarse en los diferentes códigos que se han usado, se han incorporado 3 paquetes de programación para poder generar un "puente" entre Unity y lo que son las gafas Meta Quest 2 de Meta. Estos paquetes son los siguientes:

- XR Plugin Management: Herramienta de Unity que permite manejar la realidad virtual, facilitando la configuración y gestión esta. El término XR hace referencia a lo que es la realidad extendida "extended reality", sistema que engloba la realidad virtual y aumentada. Con este paquete se facilita la configuración de ambos entornos, permitiendo un fácil apoyo y soporte para los soportes de realidad virtual, en este caso las Meta Quest 2.

- Meta XR All-in-One SDK: Paquete desarrollado por el propio Meta para el desarrollo de software en aplicaciones de realidad virtual en dispositivos propios de Meta. El SDK de este proporciona todas las herramientas y bibliotecas necesarias para la creación y posterior prueba y despliegue de aquellas aplicaciones a emplear en estos dispositivos.

- Auto Hands VR: Paquete encargado de ofrecer físicas realistas para una interacción manual en este tipo de entorno, permitiendo que las manos virtuales puedan interactuar con los diversos objetos creados en 3D, simulando de una manera precisa como se haría el mismo proceso en la realidad.

La combinación de estos 3 paquetes es esencial para el correcto desarrollo del proceso en la realidad virtual, permitiendo así el correcto uso del proceso diseñado.

Para realizar el puente Unity – gafas de VR se han seguido los pasos siguientes:

- Una vez instalado el paquete XR Plugin Management, asegurarse de que el soporte para las gafas de realidad virtual esté habilitado, esencial para que Unity sea capaz de comunicarse de forma correcta con las Meta Quest 2.

- Cambiar la plataforma de tu proyecto a Android (software de Meta).

- Instalar el paquete Oculus Integration que proporciona herramientas y scripts específicos para el desarrollo en dispositivos Oculus. Una vez instalado, se configura el proyecto para asegurarse de que todo está listo para usarse en la realidad virtual.

Colocación de guantes:

A continuación, se va a mostrar el código empleado desde inicio a fin de todo el proceso de fabricación, comenzando con la colocación de los guantes para poder comenzar con las herramientas.

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

namespace Autohand
{
    // Esta clase controla la funcionalidad de ponerse guantes en un
juego de VR.
    public class GlovesWear : MonoBehaviour
    {
        // Sección en el inspector para configurar las referencias de las
manos antiguas (antes de ponerse los guantes)
        [Header("Old Hands")]
        public Hand Left;  // Referencia a la mano izquierda sin guante
        public Hand Right; // Referencia a la mano derecha sin guante

        // Sección en el inspector para configurar el controlador del
jugador
        [Header("Player Controller")]
        public AutoHandPlayer player; // Referencia al controlador del
jugador, que gestiona las manos en VR

        // Sección en el inspector para configurar las referencias de las
manos con guantes
        [Header("Gloves")]
        public Hand DLeft;  // Referencia a la mano izquierda con guante
        public Hand DRight; // Referencia a la mano derecha con guante

        // Sección en el inspector para configurar la referencia al
Tooltip (indicador visual) para ponerse los guantes
        [Header("Tool Tip For Wearing Gloves")]
```

```
        public GameObject Canvas; // Referencia al objeto Canvas que
muestra un mensaje de ayuda sobre cómo ponerse los guantes

        // Sección en el inspector para configurar los efectos de sonido
al ponerse los guantes
        [Header("SoundEffect")]
        public AudioSource source; // Fuente de audio que reproducirá el
sonido al ponerse los guantes
        public AudioClip clip;     // Clip de audio que se reproducirá

        // Método Start, se ejecuta cuando la escena inicia
        void Start()
        {
            // Añade un componente AudioSource al GameObject que contiene
este script
            source = gameObject.AddComponent<AudioSource>();

            // Asigna el clip de audio que se reproducirá al componente
AudioSource
            source.clip = clip;
        }

        // Método para ponerse el guante izquierdo
        public void WearLeft()
        {
            // Desactiva la mano izquierda antigua (sin guante)
            Left.gameObject.SetActive(false);

            // Activa la mano izquierda con guante
            DLeft.gameObject.SetActive(true);

            // Actualiza la referencia en el controlador del jugador para
que ahora use la mano con guante
            player.handLeft = DLeft;

            // Reproduce el sonido asignado
            source.Play();
        }

        // Método para ponerse el guante derecho
        public void WearRight()
        {
            // Desactiva la mano derecha antigua (sin guante)
            Right.gameObject.SetActive(false);

            // Activa la mano derecha con guante
            DRight.gameObject.SetActive(true);
```

```
            // Actualiza la referencia en el controlador del jugador para
que ahora use la mano con guante
            player.handLeft = DRight;

            // Reproduce el sonido asignado
            source.Play();

            // Desactiva el Tooltip (indicador visual) una vez que se ha
puesto el guante derecho
            Canvas.SetActive(false);
        }

        // Método para eliminar un objeto de guante
        public void delete(GameObject Gloves)
        {
            // Destruye el GameObject pasado como parámetro
            Destroy(Gloves.gameObject);
        }

        // Método Update, se llama una vez por frame, aunque aquí no se
está usando
        void Update()
        {
            // No se hace nada en este método en el estado actual del
código
        }
    }
}
```

Este script permite al usuario ponerse los guantes para proceder a realizar el proceso de fabricación. En cuanto a puntos a destacar:

- "GlovesWear": clase encargada de la gestión de la transición de las manos del jugador de un estado sin guantes a un estado con guantes.

- Variables públicas "Left", "Right", "DLeft", "DRight": Referencias a los objetos en la escena que representan a las manos del usuario antes y después de ponerse los guantes.

- "WearLeft" y "WearRight": Se acuden a estas cuando el usuario se coloca el guante izquierdo y derecho, desactivando lo que es la mano antigua y activando una nueva ya con ambos guantes puestos.

Pinzas:

El siguiente script gira entorno a las pinzas y el proceso de sujetar con ellas durante todo el proceso de fabricación, siendo estas el modo de transporte de los dos cristales a lo largo de la cadena.

```
using UnityEngine;
using Autohand;
using TMPro;
using System.Collections;

public class TweezerGrab : MonoBehaviour
{
    // Variables públicas para referencias en el inspector
    public GameObject currentCrystal; // Referencia al cristal actual que
se está agarrando
    public Transform grabPoint; // Punto de agarre en las pinzas donde se
sostendrá el cristal
    public TextMeshProUGUI time; // Componente TextMeshProUGUI para
mostrar el temporizador

    private Coroutine timerCoroutine; // Referencia a la Coroutine del
temporizador
    public bool check = false; // Indicador para verificar si el
temporizador está en funcionamiento

    [Header("Timer Canvas")]
    public GameObject Canvas; // Referencia al objeto Canvas que muestra
el temporizador

    void Start()
    {
        // Asegura que la escala de tiempo sea normal al iniciar
        Time.timeScale = 1f;
    }

    // Método que se ejecuta cuando otro objeto colisiona con el
GameObject que contiene este script
    private void OnCollisionEnter(Collision other)
    {
        // Verifica si el objeto que colisionó tiene la etiqueta
"Crystal" y si no hay un cristal ya agarrado
        if (other.gameObject.CompareTag("Crystal") && currentCrystal ==
null)
        {
            // Asigna el objeto colisionado como el cristal actual
```

```
            currentCrystal = other.gameObject;

            if (currentCrystal != null)
            {
                // Establece el cristal como hijo del punto de agarre en
las pinzas
                currentCrystal.transform.SetParent(grabPoint);
                currentCrystal.transform.localPosition = Vector3.zero; //
Coloca el cristal en la posición exacta del punto de agarre
                currentCrystal.transform.localRotation =
Quaternion.identity; // Resetea la rotación del cristal

                // Desactiva la física para mantener el cristal fijo en
las pinzas
                Rigidbody crystalRb =
currentCrystal.GetComponent<Rigidbody>();
                Grabbable grabber =
currentCrystal.GetComponent<Grabbable>();

                if (crystalRb != null)
                {
                    // Elimina el Rigidbody y el componente Grabbable
para que el cristal no se mueva ni pueda ser agarrado por otros medios
                    Destroy(crystalRb);
                    Destroy(grabber);
                    Canvas.SetActive(true); // Activa el Canvas para
mostrar el temporizador
                }

                // Inicia el temporizador si no está ya en funcionamiento
                if (timerCoroutine != null && !check)
                {
                    StopCoroutine(timerCoroutine);
                }
                timerCoroutine = StartCoroutine(StartTimer()); // Inicia
la Coroutine del temporizador
            }
        }
    }

    // Coroutine que maneja el temporizador y las acciones tras
completarse
    private IEnumerator StartTimer()
    {
        check = true; // Marca que el temporizador está en funcionamiento

        float elapsedTime = 0f;
        float timerDuration = 30f; // Duración del temporizador en
segundos
```

```
        while (elapsedTime < timerDuration)
        {
            elapsedTime += Time.deltaTime; // Incrementa el tiempo
transcurrido
            time.text = "Wait " + $"{(timerDuration - elapsedTime):F1}" +
" Seconds"; // Actualiza la UI del temporizador con un decimal
            yield return null; // Espera al siguiente frame
        }

        // Una vez completado el temporizador, libera el cristal y cambia
su etiqueta
        if (currentCrystal != null)
        {
            // Libera el cristal eliminando su relación de parentesco con
las pinzas
            currentCrystal.transform.SetParent(null);
            yield return null; // Espera un frame antes de agregar de
nuevo el Rigidbody

            // Añade de nuevo un Rigidbody para restaurar la física del
cristal
            Rigidbody crystalRb =
currentCrystal.gameObject.AddComponent<Rigidbody>();

            // Configura las propiedades del Rigidbody para evitar
comportamientos extraños
            crystalRb.mass = 1f; // Ajusta la masa según sea necesario
            crystalRb.drag = 0.5f; // Ajusta la fricción lineal
            crystalRb.angularDrag = 0.5f; // Ajusta la fricción angular

            // Opcionalmente, congela la posición en Y para estabilizar
el cristal momentáneamente
            crystalRb.constraints = RigidbodyConstraints.FreezePositionY;

            // Vuelve a agregar el componente Grabbable para que el
cristal pueda ser agarrado nuevamente
            currentCrystal.gameObject.AddComponent<Grabbable>();

            // Espera un breve momento para que el cristal se estabilice
            yield return new WaitForSeconds(0.1f);

            // Descongela las restricciones de posición
            crystalRb.constraints = RigidbodyConstraints.None;

            // Cambia la etiqueta del cristal a "DriedCrystal"
            currentCrystal.gameObject.tag = "DriedCrystal";

            // Desactiva el Canvas y reinicia el check
```

```
            Canvas.SetActive(false);
            check = false;

            // Libera la referencia al cristal actual
            currentCrystal = null;
        }

        // Detiene la Coroutine del temporizador
        timerCoroutine = null;
    }
}
```

En cuanto a variables de tipo público se refiere, destacar la variable “currentCrystal” que es la encargada de referenciar que el cristal está siendo sujetado por las pinzas.

Para comprobar que esta “colisión” entre pinzas y cristal se usa el método “OnCollisionEnter”, que se ejecuta cuando el GameObject con este script en su Prefab colisiona con otro objeto. En el caso de que el objeto en colisión tenga la etiqueta “Crystal" y no haya ninguno de los dos cristales agarrados por las pinzas (currentCrystal == null) se comienza pues con el proceso de agarre.

Alcohol isopropílico:

Una vez realizado este proceso de agarre se procede a limpiar los dos cristales con alcohol isopropílico, como se ha comentado en el capítulo dedicado a explicar el proceso paso a paso. El script que hace referencia a este es el siguiente.

```
using LiquidVolumeFX;
using UnityEngine;

public class PourAlcohol : MonoBehaviour
{
    // Variables públicas que se configuran en el Inspector de Unity
    public ParticleSystem alcoholLeakParticles;  // Sistema de partículas
que simula el escape de agua, asignado en el Inspector
    public float pourAngleThreshold = 30f;     // Umbral de ángulo para
determinar cuándo el líquido comienza a derramarse
    public LiquidVolume volume;                // Referencia al
componente LiquidVolume para controlar el nivel de líquido
    public float drainRate = 0.05f;            // Tasa a la que disminuye
el nivel de líquido por segundo

    void Update()
    {
```

```
        // Obtiene la rotación actual de la botella en el eje X en
espacio local
        float currentRotationX = transform.localEulerAngles.x;

        // Normaliza el ángulo de rotación para que esté entre -180 y 180
grados
        if (currentRotationX > 180)
        {
            currentRotationX -= 360;
        }

        // Si la botella está lo suficientemente inclinada, disminuye el
nivel de líquido y activa las partículas
        if (currentRotationX <= -pourAngleThreshold && volume.level > 0)
        {
            // Si las partículas no están reproduciéndose, inicia el
efecto de fuga de agua
            if (!alcoholLeakParticles.isPlaying)
            {
                alcoholLeakParticles.Play();  // Inicia el sistema de
partículas
            }

            // Disminuye gradualmente el nivel de líquido según la tasa
de drenaje y el tiempo transcurrido
            volume.level -= drainRate * Time.deltaTime;
            // Asegura que el nivel de líquido no baje por debajo de 0
            volume.level = Mathf.Max(volume.level, 0);
        }
        else
        {
            // Si la botella no está lo suficientemente inclinada,
detiene el efecto de partículas
            if (alcoholLeakParticles.isPlaying)
            {
                alcoholLeakParticles.Stop();  // Detiene el sistema de
partículas
            }
        }
    }
}
```

La variable “alcoholLeakParticles” simula el proceso del alcohol saliendo de la botella, proceso que se activa una vez se coge la botella y se inclina a un determinado ángulo definido por la variable “pourAngleThreshold”.

Además de estas dos, se hace uso de la variable “volume” para asegurarse de que hay liquido dentro de la botella. La combinación del angulo de tirada y el líquido de la botella permitirá tirar el alcohol sobre los cristales.

Depósito ultrasonido:

El proceso de limpieza finaliza con los cristales en el depósito de ultrasonido para pulir aquellas bacterias y residuos que el alcohol no ha quitado.

```
using UnityEngine;
using TMPro;
using System.Collections;

public class UltrasoundDeposit : MonoBehaviour
{
    // Referencia al componente TextMeshProUGUI que mostrará el tiempo
restante para el depósito
    public TextMeshProUGUI depositTime;

    // Bandera para verificar si el proceso de limpieza (depósito) está
en curso
    private bool isCleaning = false;

    // Duración del depósito en segundos (2 minutos en este caso)
    public float depositDuration = 120f;

    // Método que se llama cuando otro objeto entra en el trigger de este
GameObject
    private void OnTriggerEnter(Collider other)
    {
        // Verifica si el objeto que entra tiene la etiqueta
"DriedCrystal"
        if (other.CompareTag("DriedCrystal"))
        {
            // Inicia el proceso de limpieza si no está ya en curso
            StartCleaning();
        }
    }

    // Método para iniciar el proceso de limpieza
    private void StartCleaning()
    {
        // Solo inicia el proceso si no hay otro en curso
        if (!isCleaning)
        {
            isCleaning = true; // Marca que la limpieza está en curso
```

```
            StartCoroutine(StartDepositTimer()); // Inicia la Coroutine
del temporizador de depósito
        }
    }

    // Coroutine que maneja el temporizador del proceso de depósito
    private IEnumerator StartDepositTimer()
    {
        float elapsedTime = 0f; // Tiempo transcurrido

        // Bucle que se ejecuta hasta que el tiempo transcurrido iguale
la duración del depósito
        while (elapsedTime < depositDuration)
        {
            elapsedTime += Time.deltaTime; // Incrementa el tiempo
transcurrido
            depositTime.text = $"{(depositDuration - elapsedTime):F1}" +
"  Seconds Left"; // Actualiza la UI del temporizador
            yield return null; // Espera hasta el siguiente frame
        }

        // Una vez completado el temporizador
        depositTime.text = "Deposit complete"; // Muestra que el depósito
está completo en la UI
        isCleaning = false; // Resetea la bandera para permitir futuros
procesos de limpieza
    }
}
```

Las variables públicas “depositTime” y “depositDuration” son las encargadas de mostrar el tiempo restante de los cristales dentro del depósito y la duración total de este, siendo esta de 120 segundos.

La variable privada “isCleaning”, que actúa como bandera, es la encargada de verificar que el proceso del depósito se ha iniciado. Para comprobar que no hay otro método en marcha se hace uso del método “StartCleaning” para asegurarse de que este proceso, controlado por “isCleaning” es el único en curso.

Una vez finalizado el proceso de limpieza por completo en las dos placas de cristal se avanza a la fase de alineamiento empezando por depositar ambas placas en el Spin Coater donde se realiza el proceso de centrifugado una vez se le añaden las gotas de poliamida a ambos.

Poliamida:

El siguiente código es el que se ocupa del proceso de coger la pipeta con la que se extrae el líquido de la poliamida para poner las respectivas gotas sobre los cristales.

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

public class Pippet : MonoBehaviour
{
    // Variables públicas para controlar el estado de la pipeta y las
partículas
    public bool isGrabbed;              // Indica si la pipeta está
siendo agarrada
    public ParticleSystem Droplets;     // Referencia al sistema de
partículas que simula las gotas

    // Método para marcar la pipeta como agarrada
    public void grabbed()
    {
        isGrabbed = true;  // Cambia el estado a "agarrado"
    }

    // Método para marcar la pipeta como liberada
    public void Released()
    {
        isGrabbed = false;  // Cambia el estado a "liberado"
    }

    // Método Start, se llama al iniciar la escena
    void Start()
    {
        Droplets.Stop();  // Asegura que el sistema de partículas esté
detenido al inicio
    }

    // Método para salir de la aplicación (probablemente para pruebas o
como botón de salida)
    public void Exit()
    {
        Application.Quit();  // Cierra la aplicación
    }

    // Método Update, se llama una vez por frame
    void Update()
    {
        // Verifica si la pipeta está agarrada y si se presiona alguno de
los gatillos (primario o secundario)
```

```
        if(isGrabbed &&
(OVRInput.GetDown(OVRInput.Button.PrimaryIndexTrigger) ||
OVRInput.GetDown(OVRInput.Button.SecondaryIndexTrigger)))
        {
            Droplets.Play();  // Inicia el sistema de partículas para
simular gotas
        }
        // Verifica si se ha soltado alguno de los gatillos
        else if(OVRInput.GetUp(OVRInput.Button.PrimaryIndexTrigger) ||
OVRInput.GetUp(OVRInput.Button.SecondaryIndexTrigger))
        {
            Droplets.Stop();  // Detiene el sistema de partículas
        }
    }
}
```

Como puntos a destacar, las variables públicas “isGrabbed” para indicar que se la pipeta se está cogiendo por el usuario, siendo “true” cuando esta esté en uso. La otra variable que destacar “Droplets” para simular las gotas que salen de la pipeta.

Otro punto que destacar es la referencia “OVRInput”, encargada de descargar el líquido de la pipeta una vez se tenga está en la mano y se presione uno de los gatillos que se usan con las Meta Quest.

Spin Coater:

A continuación, se muestra el código que gira en torno al Spin Coater y su correspondiente configuración.

```
using UnityEngine;
using TMPro;
using UnityEngine.UI;
using System.Collections;

public class SpinCoater : MonoBehaviour
{
    // Referencias públicas para componentes en el Inspector de Unity
    public TextMeshProUGUI timerText;  // Referencia al componente
TextMeshProUGUI para mostrar el temporizador
    public AudioSource timerSound;     // Referencia al componente
AudioSource para reproducir un sonido al finalizar el temporizador
    public float countdownTime = 30f;  // Duración del temporizador en
segundos

    private bool isTimerRunning = false;  // Bandera para verificar si el
temporizador está en marcha
```

```
    void Start()
    {
        // Opcionalmente, se puede iniciar el temporizador al comenzar el
juego
        // StartTimer();
    }

    private void Update()
    {
        // Inicia el temporizador cuando se presiona la tecla Espacio
        if(Input.GetKeyDown(KeyCode.Space))
        {
            StartTimer();
        }
    }

    public void StartTimer()
    {
        // Inicia el temporizador solo si no hay otro temporizador
corriendo
        if (!isTimerRunning)
        {
            StartCoroutine(TimerCoroutine());
        }
    }

    private IEnumerator TimerCoroutine()
    {
        isTimerRunning = true;  // Marca que el temporizador está
corriendo
        float remainingTime = countdownTime;  // Inicializa el tiempo
restante con la duración total

        while (remainingTime > 0)
        {
            // Actualiza el texto del temporizador en la UI con un
decimal
            timerText.text = remainingTime.ToString("F1") + " sec";

            // Espera 1 segundo
            yield return new WaitForSeconds(1f);

            // Disminuye el tiempo restante
            remainingTime--;
        }

        // Actualiza el texto del temporizador a 0 cuando finaliza
        timerText.text = "0.0s";
```

```
        // Reproduce el sonido para indicar que el temporizador ha
terminado
        timerSound.Play();

        isTimerRunning = false;  // Resetea la bandera para permitir
futuros temporizadores
    }
}
```

Como tal, destacar la Coroutine "TimerCoroutine" que engloba una serie de variables como las siguientes:

- "isTimerRunning = true": para indicar que el temporizador está en marcha.

- Bucle "while (remainingTime > 0)": Bucle que se ejecuta hasta que la variable "remainingTime" llega a cero.

Placa calefactora:

Una vez acabado el proceso en el Spin Coater se saca el cristal con las pinzas para dejarlo sobre la placa calefactora. El código encargado de configurar la placa es el siguiente.

```
using UnityEngine;
using TMPro;  // Importa el namespace TextMeshPro para utilizar
componentes de texto avanzados
using System.Collections;  // Importa el namespace para utilizar
Coroutines

public class HeatingPlate : MonoBehaviour
{
    // Referencias públicas a componentes TextMeshProUGUI para mostrar el
temporizador y la temperatura en la UI
    public TextMeshProUGUI timerText;       // Referencia al componente
TextMeshProUGUI que muestra el temporizador
    public TextMeshProUGUI temperatureText; // Referencia al componente
TextMeshProUGUI que muestra la temperatura

    // Configuraciones para las dos fases de calentamiento
    public float firstTemperature = 80f;    // Primera temperatura
objetivo (80 grados Celsius)
```

```
    public float firstDuration = 10f;       // Duración de la primera
fase (10 minutos)
    public float secondTemperature = 210f;  // Segunda temperatura
objetivo (210 grados Celsius)
    public float secondDuration = 45f;      // Duración de la segunda
fase (45 minutos)

    // Sonido que se reproducirá cuando el proceso de calentamiento se
complete
    public AudioSource heatingCompleteSound; // Referencia al AudioSource
para reproducir un sonido al finalizar el calentamiento

    // Factor de escala de tiempo para acelerar el proceso de
calentamiento (2x significa que 1 segundo en tiempo real equivale a 2
minutos en el juego)
    public float timeScale = 2f;  // Escala de tiempo para acelerar el
proceso (60x significa que 1 segundo en tiempo real equivale a 1 minuto
en el juego)

    // Variable privada para saber si el proceso de calentamiento está en
curso
    private bool isHeating = false;

    // Método Update, se llama una vez por frame
    private void Update()
    {
        // Si se presiona la tecla 'V', se inicia el proceso de
calentamiento
        if (Input.GetKeyDown(KeyCode.V))
        {
            StartHeatingProcess();
        }
    }

    // Método para iniciar el proceso de calentamiento
    public void StartHeatingProcess()
    {
        // Si no se está calentando, comienza la Coroutine del proceso de
calentamiento
        if (!isHeating)
        {
            StartCoroutine(HeatingCoroutine());
        }
    }

    // Coroutine que gestiona las dos fases de calentamiento
    private IEnumerator HeatingCoroutine()
    {
        isHeating = true;  // Marca que el calentamiento ha comenzado
```

```
        // Primera fase de calentamiento
        yield return StartCoroutine(HeatForDuration(firstTemperature,
firstDuration));

        // Segunda fase de calentamiento
        yield return StartCoroutine(HeatForDuration(secondTemperature,
secondDuration));

        // Proceso de calentamiento completado
        heatingCompleteSound.Play();  // Reproduce el sonido de
finalización
        timerText.text = "Heating Complete!";  // Muestra en la UI que el
calentamiento ha finalizado
        temperatureText.text = "0°C";  // Resetea la temperatura mostrada
a 0°C
        isHeating = false;  // Marca que el proceso de calentamiento ha
terminado
    }

    // Coroutine que calienta la placa a una temperatura durante una
duración específica
    private IEnumerator HeatForDuration(float temperature, float
duration)
    {
        temperatureText.text = "Temperature Reaching: " +
temperature.ToString() + "°C";  // Muestra la temperatura objetivo en la
UI

        float remainingTime = duration * 60;  // Convierte la duración de
minutos a segundos

        // Bucle que cuenta hacia atrás el tiempo restante
        while (remainingTime > 0)
        {
            timerText.text = "Time Left: " +
FormatTime(remainingTime);  // Actualiza el texto del temporizador en la
UI

            yield return new WaitForSeconds(1f / timeScale);  // Espera
un segundo escalado por timeScale (aceleración del tiempo)
            remainingTime -= timeScale;  // Reduce el tiempo restante en
función de timeScale
        }
    }

    // Método para formatear el tiempo en minutos y segundos (mm:ss)
    private string FormatTime(float seconds)
    {
```

```
        int minutes = Mathf.FloorToInt(seconds / 60F);  // Convierte los
segundos a minutos
        int secs = Mathf.FloorToInt(seconds % 60F);     // Obtiene los
segundos restantes después de calcular los minutos
        return string.Format("{0:0}:{1:00}", minutes, secs);  // Formatea
el tiempo como "mm:ss"
    }
}
```

En este bloque destacar la importación de los NameSpaces “using TMPro” que permite mostrar texto en la interfaz del usuario y “using System.Collections” para realizar tareas que se tienen que ejecutar en varios espacios o “frames”.

En cuanto a variables de carácter público tener en cuentas las siguientes:

- “timerText” y “temperatureText”: Ambas harán referencia a componentes del interfaz, encargadas de mostrar tiempo restante y temperatura durante el proceso de calentamiento.

- “firstTemperature”, “firstDuration” y “secondTemperature”: Definen lo que son las dos temperaturas de calentamiento de las placas de cristal, además de la temperatura objetivo y la duración de cada fase.

De variables privadas destacar “isHeating”, una bandera o “flag” que muestra que el proceso de calentamiento ha comenzado, evitando que el proceso se inicie varias veces de forma simultánea.

Un aspecto que destacar de este bloque de código son las “Coroutine”, métodos especiales que permiten ejecutar el código a lo largo del tiempo y no solo en un frame, siendo en este caso necesario para la espera que se hace antes de depositar los cristales en la placa calefactora.

- “HeatingCoroutine”: encargada de gestionar el calentamiento de la placa en la primera y segunda fase. Una vez finalizados ambos procesos actualiza la interfaz del usuario (UI) para indicar que se ha acabado con este.

- “HeatForDuration”: Esta “Coroutine” se encarga de dirigir cada una de las fases de manera separada, mostrando las respectivas temperaturas y el tiempo restante de cada fase actualizando asi el temporizador en la interfaz.

Terciopelo:

Una vez acabado se les pasa el terciopelo a los cristales para crear los surcos necesarios. Importante realizar el frotado en una misma dirección y sentido para evitar problemas en el futuro.

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

public class SandedCrystal : MonoBehaviour
{
    // Variables públicas que se configuran en el Inspector de Unity
    public GameObject Base;  // Referencia al objeto base que se activará
    public bool isBase;      // Bandera para identificar si este objeto
es la base

    // Método que se llama automáticamente cuando este objeto colisiona
con otro
    private void OnCollisionEnter(Collision other)
    {
        // Verifica si el objeto colisionado tiene la etiqueta
"SandedCrystal" y si este objeto es la base
        if(other.gameObject.tag == "SandedCrystal" && isBase)
        {
            // Destruye el otro objeto con la etiqueta "SandedCrystal"
            Destroy(other.gameObject);

            // Activa el objeto base
            Base.SetActive(true);
        }
    }
}
```

Resaltar el método “OnCollisionEnter”, método que se llama de forma automática por Unity cuando el objeto en cuestión colisiona con otro objeto. Para asegurarse de que los objetos que colisionan entre si son los cristales y el terciopelo se hace uso de un condicional “other.gameObject.tag == "SandedCrystal”” para verificar que el objeto con el que se ha chocado es el correcto y asi proceder con el paso de forma correcta.

Colisiones:

Este Script recoge las colisiones que ocurren durante todo el proceso como la que se usa en el código previo y a las que Unity accede de forma automática cuando suceden.

```
using UnityEngine;

public class ParticleCollisionDetector : MonoBehaviour
```

```
{
    // Referencia pública a un objeto Canvas que se utilizará para
mostrar un mensaje o UI cuando ocurra la colisión
    public GameObject Canvas;

    // Este método se llama automáticamente cuando un sistema de
partículas colisiona con este GameObject
    private void OnParticleCollision(GameObject other)
    {
        // Verifica si el objeto que colisionó tiene la etiqueta
"SandedCrystal"
        if(other.gameObject.tag == "SandedCrystal")
        {
            // Imprime un mensaje en la consola cuando las partículas
colisionan con un objeto que tiene la etiqueta "SandedCrystal"
            Debug.Log("Particles collided with " + other.name);

            // Activa el objeto Canvas, que podría mostrar alguna
información o UI en la pantalla
            Canvas.SetActive(true);
        }
    }
}
```

Separadores:

Una vez finalizada esta parte, comienza la parte final dedicada al ensamblaje de los dos cristales. El siguiente bloque de código hace referencia al paso de pegar los separadores de los cristales usando el pegamento óptico.

```
using LiquidVolumeFX;
using UnityEngine;

public class PourGlue : MonoBehaviour
{
    // Variables públicas configurables desde el Inspector
    public ParticleSystem glueLeakParticles;  // Sistema de partículas
que simula el derrame de pegamento
    public float pourAngleThreshold = 30f;     // Umbral de ángulo para
comenzar a verter el pegamento
    public float drainRate = 0.05f;            // Tasa a la que disminuye
el nivel del líquido (no se usa en este código)

    void Update()
```

```
    {
        // Obtiene la rotación actual de la botella en el eje X en
espacio local
        float currentRotationX = transform.localEulerAngles.x;

        // Normaliza el ángulo de rotación para que esté entre -180 y 180
grados
        if (currentRotationX > 180)
        {
            currentRotationX -= 360;
        }

        // Si la botella está inclinada lo suficiente, se reproducen las
partículas del derrame de pegamento
        if (currentRotationX <= -pourAngleThreshold)
        {
            // Si las partículas no están ya en reproducción, las inicia
            if (!glueLeakParticles.isPlaying)
            {
                glueLeakParticles.Play();  // Inicia el sistema de
partículas
            }
        }
        else
        {
            // Si la botella no está suficientemente inclinada, se
detienen las partículas
            if (glueLeakParticles.isPlaying)
            {
                glueLeakParticles.Stop();  // Detiene el sistema de
partículas
            }
        }
    }
}
```

En este código sucede lo mismo que con el proceso de limpiado de cristales con el alcohol, donde "glueLeakParticles" simula el derrame pegamento sobre los separadores. La botella de pegamento óptico está sometida al mismo proceso que la de alcohol, se establece un ángulo determinado para que el pegamento caiga y se usa "drainRate" para controlar el líquido restante.

Pegado:

El próximo código es el encargado de replicar el proceso de pegado de los separadores una vez se ha puesto el pegamento en cada uno de los cuatro que se ponen en las esquinas del cristal.

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

public class SpacerAttachment : MonoBehaviour
{
    public List<Transform> attachmentPoints = new List<Transform>();  //
List of attachment points on the crystal

    private void OnCollisionEnter(Collision other)
    {
        if (other.gameObject.CompareTag("Spacer"))  // Ensure the object
has the tag "Spacer"
        {
            Grabbable spacerGrabbable =
other.gameObject.GetComponent<Grabbable>();
            Rigidbody spacerRb =
other.gameObject.GetComponent<Rigidbody>();

            if (spacerGrabbable != null && spacerRb != null)
            {
                // Find the closest attachment point to snap the spacer
to
                Transform closestPoint =
FindClosestAttachmentPoint(other.transform);

                if (closestPoint != null)
                {
                    // Attach the spacer to the closest attachment point
                    other.transform.position = closestPoint.position;
                    other.transform.rotation = closestPoint.rotation;
                    other.transform.SetParent(closestPoint);

                    Destroy(spacerRb);
                    Destroy(spacerGrabbable);

                    // Remove the used attachment point from the list
                    attachmentPoints.Remove(closestPoint);
                }
            }
        }
    }

    private Transform FindClosestAttachmentPoint(Transform spacer)
    {
```

```
        Transform closestPoint = null;
        float minDistance = Mathf.Infinity;

        foreach (Transform point in attachmentPoints)
        {
            float distance = Vector3.Distance(spacer.position,
point.position);
            if (distance < minDistance)
            {
                minDistance = distance;
                closestPoint = point;
            }
        }

        return closestPoint;
    }
}
```

La variable pública “attachmentPoints” genera una lista de puntos donde situar los separadores, puntos que se configuran a través del Inspector de Unity. El método “FindClosestAttachmentPoint” se encarga de recorrer estos hasta encontrar el más cercano al separador y asi situarlo de forma correcta.

Horno UV:

Una vez se han pegado los 4 separadores al cristal se dispone ya de una estructura con los dos cristales juntos separados por estos 4. Tras esto, se introduce esta en el horno UV para el curado final.

```
using UnityEngine;
using TMPro;
using System.Collections;

public class UVOven : MonoBehaviour
{
    // Variables públicas configurables desde el Inspector
    public TextMeshProUGUI timerText;  // Referencia al componente
TextMeshProUGUI para mostrar el temporizador
    public float timeInMinutes = 30f;  // Tiempo en minutos para
"cocinar" la célula
    public float timeScale = 30f;  // Multiplicador de velocidad para el
lapso de tiempo
    public AudioClip finishedSound;  // Sonido que se reproduce cuando el
tiempo se agota
```

```
    private AudioSource audioSource;  // Fuente de audio para reproducir
sonidos

    private void Start()
    {
        // Obtiene el componente AudioSource adjunto a este objeto
        audioSource = GetComponent<AudioSource>();
    }

    private void Update()
    {
        // Inicia el proceso de horneado UV cuando se presiona la tecla
'B'
        if (Input.GetKeyDown(KeyCode.B))
        {
            StartUVOvenProcess();
        }
    }

    // Método para iniciar el proceso del horno UV
    public void StartUVOvenProcess()
    {
        StartCoroutine(BakeCell());  // Inicia la Coroutine para
"cocinar" la célula
        audioSource.Play();  // Reproduce el sonido asociado al inicio
del proceso
    }

    // Coroutine que maneja el proceso de "cocinado"
    private IEnumerator BakeCell()
    {
        // Calcula el tiempo total en segundos según el multiplicador de
tiempo
        float timeInSeconds = timeInMinutes * 60f / timeScale;
        float elapsedTime = 0f;

        // Bucle que se ejecuta mientras el tiempo transcurrido es menor
que el tiempo total
        while (elapsedTime < timeInSeconds)
        {
            elapsedTime += Time.deltaTime;  // Incrementa el tiempo
transcurrido
            float remainingTime = timeInSeconds - elapsedTime;  //
Calcula el tiempo restante
            UpdateTimerDisplay(remainingTime);  // Actualiza la pantalla
del temporizador
            yield return null;  // Espera hasta el siguiente frame
        }
```

```
        // Asegura que el temporizador muestre 00:00 al finalizar
        UpdateTimerDisplay(0f);
        PlayFinishedSound();  // Reproduce el sonido de finalización
    }

    // Método para actualizar la pantalla del temporizador
    private void UpdateTimerDisplay(float remainingTime)
    {
        int minutes = Mathf.FloorToInt(remainingTime / 60f);  // Calcula
los minutos restantes
        int seconds = Mathf.FloorToInt(remainingTime % 60f);  // Calcula
los segundos restantes
        // Actualiza el texto del temporizador en formato mm:ss
        timerText.text = string.Format("{0:00}:{1:00}", minutes,
seconds);
    }

    // Método para reproducir el sonido cuando el proceso termina
    private void PlayFinishedSound()
    {
        if (finishedSound != null && audioSource != null)
        {
            // Reproduce el clip de sonido de finalización
            audioSource.PlayOneShot(finishedSound);
        }
    }
}
```

Proceso similar al realizado con las otras máquinas que se han usado en el proceso, lo que ha permitido reusar el código empleado en esas y aplicarlo en el horno UV con alguna modificación.

Una vez finalizado el curado se le añadiría lo que es el cristal líquido y se dispondría del objeto final, cristal líquido con una capa de alineación.

Con esto concluye todo el trabajo de programación y dinamismo y que daría por finalizado el trabajo de fin de grado. Como ya se ha comentado, aunque parezca un código extenso y complejo realmente son variables e instancias simples que el propio Unity integra y que permite realizar un puente a las gafas de realidad virtual de forma momentánea.

4.Resultados

En este capítulo se van a mostrar diferentes capturas de pantalla para mostrar como queda cada parte del proceso de fabricación.

1. Puesta de guantes:

Figura 11. Paso 1 fabricación

2. Limpiado de cristales con alcohol:

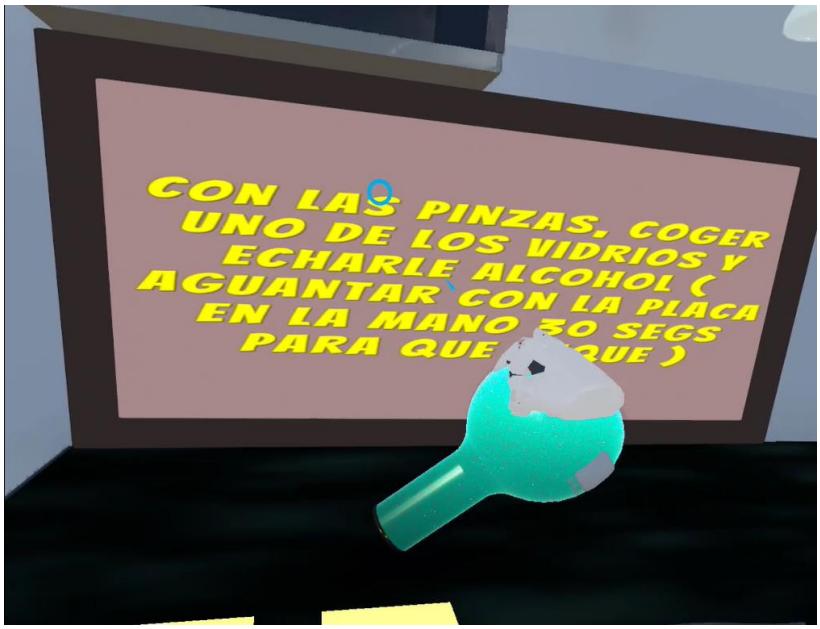

Figura 12. Paso 2 fabricación

3. Depositado en ultrasonido:

Figura 13. Paso 3 fabricación

4. Polimida en los critales y depositado en Spin Coater:

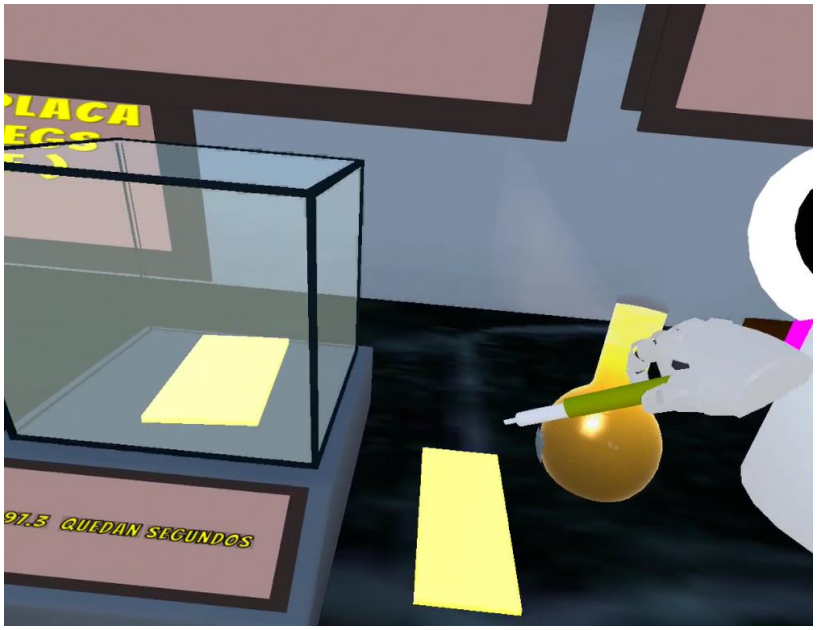

Figura 14. Paso 4 fabricación

Figura 15. Paso 4 fabricación

5. Introducción en calefactor:

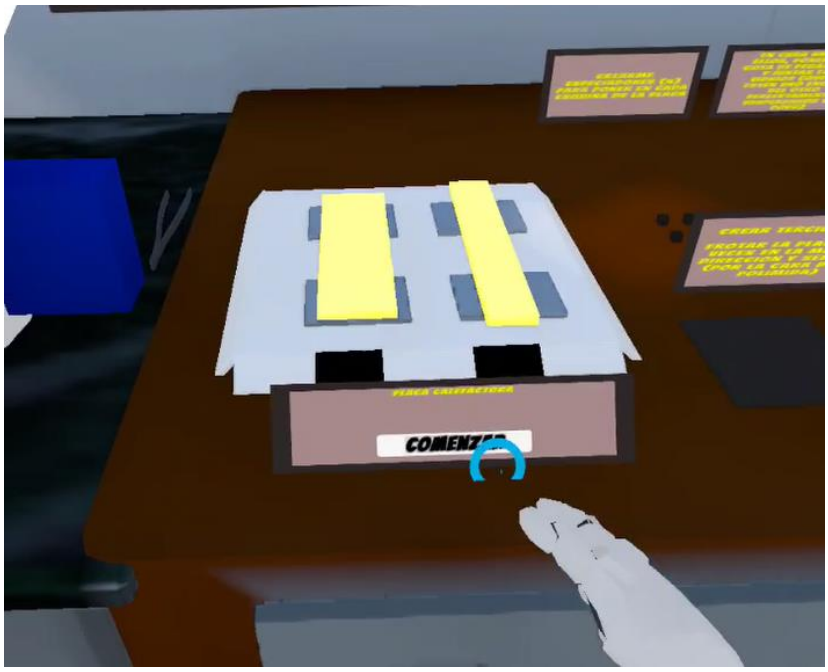

Figura 16. Paso 5 fabricación

6. Frotado con terciopelo:

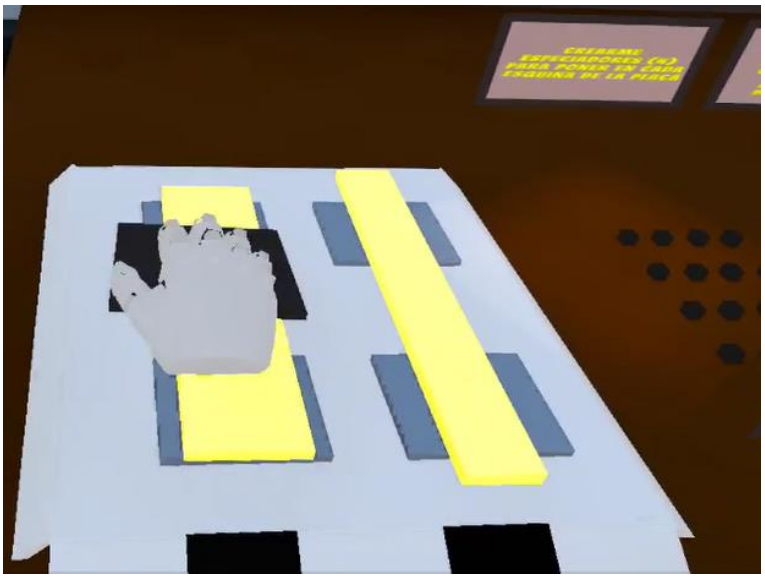

Figura 17. Paso 6 fabricación

7. Pegado de separadores:

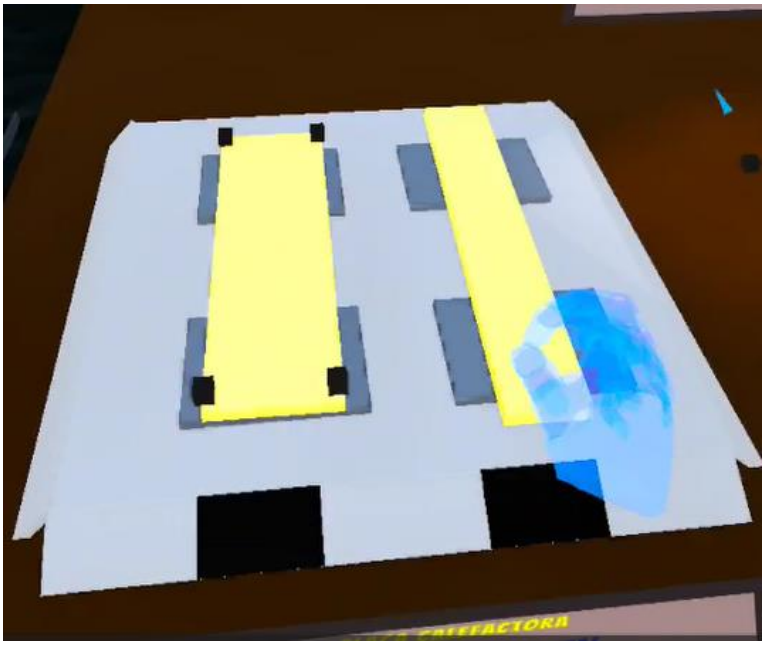

Figura 18. Paso 7 fabricación

8. Introducción en horno UV:

Figura 19. Paso 8 fabricación

9. Introducción del cristal líquido:

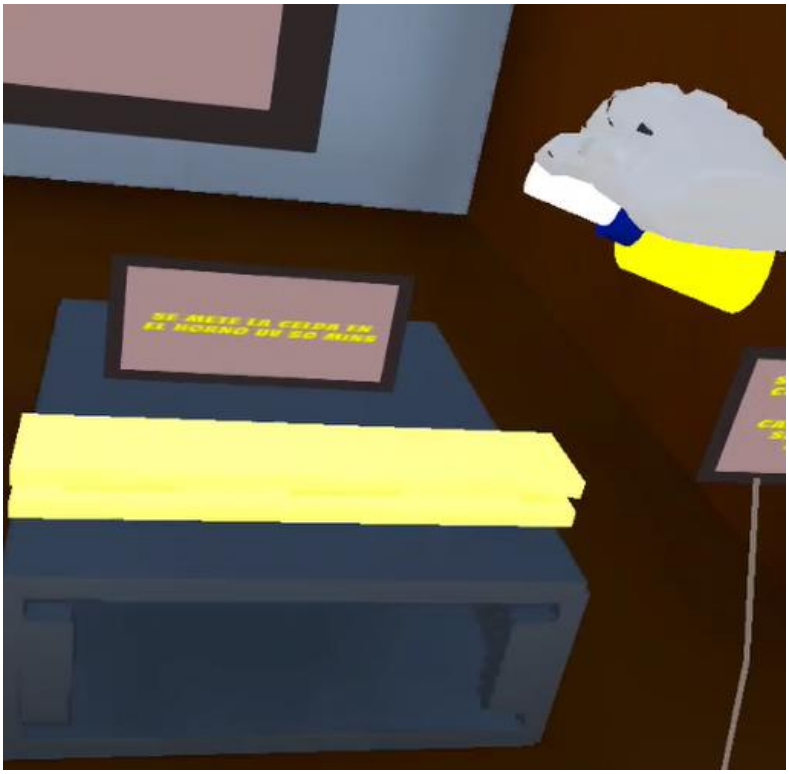

Figura 20. Paso 9 fabricación

4. Conclusiones

Con este trabajo se ha podido demostrar la importancia de las tecnologías que nos rodean en nuestro dia a dia y como su evolución constante consigue facilitar la vida a todas las personas que tienen capacidad de acceder a ellas. Como ya se ha comentado a lo largo de este proyecto, esta innovadora tecnología que ha ido cogiendo cada vez más voz y fuerza a nivel industrial cuenta con innumerables aplicaciones para cualquier tipo de problema que surja, además de contar con una capacidad de mejora y evolución enorme gracias a toda la adaptabilidad que tiene en innumerables procesos.

Las celdas de cristal liquido con capa de alineación forman parte de una amplia gama de productos electrónicos que dependen de una buena calidad y rendimiento de estas celdas, notable especialmente en el apartado destinado a la correcta alineación de las moléculas de cristal líquido.

En el contexto de este trabajo, el desarrollo en realidad virtual de todo este proceso cuenta con una importancia muy grande al estar compuesto por diferentes fases que deben ser realizadas con precisión para asegurar un producto de calidad excelente y asegurar asi su funcionalidad y durabilidad.

A nivel empresarial, ya son muchas las empresas las que están adoptando esta tecnología para obtener soluciones y mejorar sus productos de una manera innovadora, además de conseguir un alivio a nivel de costes empleados en productos y personal.

Es por esto que, el desarrollo y uso de estos gemelos digitales representan un claro avance a nivel industrial y empresarial. El poseer una herramienta con la capacidad de simulación, monitorización y optimización a tiempo real ofrece a las empresas una clara ventaja respecto a sus principales competidores, convirtiéndoles en los principales referentes de su industria.

5.ANEXO

5.1. Vídeo Proceso de Fabricación en Digital Twin

https://drive.google.com/file/d/1jFQlVd753IyZiPIwG4ulFuANgghk9MVH/view?usp=drive_link

5.2.ODS

En relación con el trabajo de fin de grado, se pueden establecer diversas relaciones con los objetivos de desarrollo sostenible.

- ODS 8, Trabajo Decente y Crecimiento Económico: Una correcta implementación y uso de los Digital Twins en los diferentes procesos que realiza una empresa puede aumentar la eficiencia y productividad en todo el proceso completo de fabricación.

- ODS 9, Industria, Innovación e Infraestructura: A través del Digital Twin se consigue optimizar el proceso de fabricación, permitiendo asi un uso más eficiente de los recursos y dirigiéndose hacia una práctica más sostenible.

- ODS 17, Alianzas para Lograr los Objetivos: El desarrollo e implementación de un Digital Twin para realizar todo el proceso de fabricación del cristal liquido involucra diferentes organizaciones e instituciones a nivel tecnológico para promover esta nueva innovadora tecnología y su incorporación en las diferentes empresas que requieren de su uso.

6. Bibliografía

[1] M. Grieves, Digital Twin: Manufacturing Excellence through Virtual Factory, 2014.

[2] H. Z. A. L. y. A. Y. C. N. F. Tao, «Digital Twin in Industry: State-of-the-Art,» *IEEE Transactions on Industrial Informatics,* pp. 2405-2415, 2019.

[3] L. F. y. M. M. E. Negro, «A Review of the Roles of Digital Twin in CPS-based Production Systems,» *Procedia Manufacturing,* pp. 939-948, 2017.

[4] Z. F. C. D. y. C. B. A. Fuller, «Digital Twin: Enabling Technologies, Challenges and Open Research,» *IEEE Access,* vol. 8, pp. 108952-108971, 2020.

[5] M. K. G. T. J. H. y. W. S. W. Kritzinger, «Digital Twin in Manufacturing: A Categorical Literature Review and Classification,» *IFAC-PapersOnLine,* vol. 51, pp. 1016-1022, 2018.

[6] Siemens, «Siemens Digital Industries: Leading the Digital Transformation,» https://new.siemens.com/global/en/company/stories/industry/siemens-digital-industries-leading-the-digital-transformation.html, 2020.

[7] G. Electric, «How GE Uses Digital Twins to Make Its Jet Engines More Efficient,» https://www.ge.com/reports/how-ge-uses-digital-twins-to-make-its-jet-engines-more-efficient/, 2018.

[8] DHL, «DHL Embraces Digital Twin Technology to Enhance Supply Chain Management,» https://www.dhl.com/global-en/home/press/press-archive/2018/dhl-embraces-digital-twin-technology-to-enhance-supply-chain-management.html, 2018.

[9] S. W. Singer, Liquid Crystal Display Manufacturing: Processes and Materials, Nueva York, 2018.

[10] Y. W. y. K. Y. M. Kimura, «Alignment Technologies for Liquid Crystal Displays,» *ournal of the Society for Information Display,* vol. 28, nº 3, pp. 201-213, 2020.

[11] J. L. y. H. Yoshida, «Advanced Polymer Alignment Layers for Liquid Crystal Displays,» *Handbook of Liquid Crystals,* vol. 3, nº 2, pp. 150-175, 2019.

[12] S. Display, «New Horizons in Liquid Crystal Display Technology,» https://news.samsung.com/global/new-horizons-in-liquid-crystal-display-technology, 2019.

[13] L. Display, «Innovative Alignment Techniques for High-Resolution LCDs,» LG Display Press Releases, 2020.

[14] S. Corporation, «Advanced Liquid Crystal Display Manufacturing Technologies,» Sharp Electronics, 2018.