ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería Civil, de Materiales y Fabricación

Área de Ingeniería de los Procesos de Fabricación

PROYECTO/TRABAJO FIN DE MÁSTER

APLICACIÓN DE TÉCNICAS DE REALIDAD VIRTUAL PARA LA CREACIÓN DE UNA SALA EXPOSITIVA INTERACTIVA EN EL MUSEO DE LA ESCUELA DE INGENIERÍAS INDUSTRIALES DE LA UNIVERSIDAD DE MÁLAGA

Máster en Ingeniería Industrial

Autor: CARMEN MEGÍAS GÓMEZ

Tutor: FRANCISCO JAVIER TRUJILLO VILCHES

Cotutor LORENZO SEVILLA HURTADO

MÁLAGA, Julio de 2.025

RESUMEN

El presente Trabajo Fin de Máster presenta el diseño de un museo virtual interactivo destinado a la conservación y divulgación del patrimonio industrial, dentro de la Escuela de Ingenierías Industriales de la Universidad de Málaga. Mediante el uso del motor gráfico Unity 3D, junto con herramientas como Blender o Visual Studio, se ha creado un entorno inmersivo accesible a través de gafas Meta Quest 2, donde el usuario puede recorrer una sala expositiva y explorar distintas máquinas-herramienta históricas.

Cada máquina ha sido adaptada e integrada en el entorno para permitir su visualización, activación de movimientos mediante botones virtuales y consulta de información técnica e histórica a través de paneles informativos. Todo el proceso, desde la importación de modelos CAD hasta la instalación de la aplicación en las gafas VR, ha sido documentado paso a paso.

Este proyecto surge de la necesidad de encontrar nuevas formas de preservar el patrimonio industrial ya que este conjunto es difícil, o imposible, de exponer de manera conjunta, dado que muchas de las máquinas-herramienta no existen ya físicamente y solo se conserva su patente o documentación gráfica. Las tecnologías de realidad virtual, propias de la Industria 4.0., permiten superar barreras físicas de preservación, ofreciendo entornos simulados de alta fidelidad que facilitan tanto la comprensión visual como la interacción directa con los objetos.

El resultado es una herramienta educativa y divulgativa con gran potencial en contextos formativos. Además, se plantean mejoras futuras para optimizar el museo virtual. En conjunto, se demuestra el valor de la realidad virtual como recurso accesible, escalable y eficaz para la puesta en valor del patrimonio industrial y de la difusión del conocimiento e información.

PALABRAS CLAVE: Realidad virtual, patrimonio industrial, unity 3D, museo virtual, máquinas-herramienta

ABSTRACT

This Final Master Project presents the design of an interactive virtual museum dedicated to the conservation and dissemination of industrial heritage, within the School of Industrial Engineering at the University of Malaga. Using the Unity 3D graphics engine, together with tools such as Blender and Visual Studio, an immersive environment has been created that is accessible through Meta Quest 2 glasses, where users can walk through an exhibition hall and explore different historical machine tools.

Each machine has been adapted and integrated into the environment to enable visualisation, activation of movements using virtual buttons, and consultation of technical and historical information via information panels. The entire process, from importing CAD models to installing the application on the VR glasses, has been documented step by step.

This project stems from the need to find new ways to preserve industrial heritage, as it is difficult, if not impossible, to display this collection as a whole, given that many of the machine tools no longer physically exist and only their patents or graphic documentation remain. Virtual reality technologies, characteristic of Industry 4.0, make it possible to overcome physical barriers to preservation, offering high-fidelity simulated environments that facilitate both visual understanding and direct interaction with the objects.

The result is an educational and informative tool with great potential in training contexts. In addition, future improvements are planned to optimise the virtual museum. Overall, it demonstrates the value of virtual reality as an accessible, scalable and effective resource for promoting industrial heritage and disseminating knowledge and information.

KEY WORDS: Virtual reality, industrial heritage, Unity 3D, virtual museum, machine tool

ÍNDICE GENERAL

Índice

Índice de figuras

Índice de tablas

MEMORIA

1. Introducción

1.1. Antecedentes

La Primera Revolución Industrial (1760-1840) se considera uno de los periodos más transformadores en la historia de la humanidad, caracterizado por un cambio cíclico, irreversible y radical en múltiples ámbitos sociales, económicos y tecnológicos. Este proceso, que se inició en Inglaterra a finales del siglo XVIII, marcó un punto de inflexión entre el pasado agrario y el desarrollo de sociedades industriales, cuyos efectos se prolongan hasta nuestros días (González-Hernández et al., 2021).

Este contexto posibilitó un proceso acumulativo de avances tecnológicos, surgiendo nuevas máquinas, entre las que tuvieron un papel decisivo la máquina de vapor, Figura 1, o la máquina de bombeo de agua de las minas. Del mismo modo, otro avance importante fue la creación de las fresadoras para perfeccionar el tallado de dientes de engranajes (Chaves Palacios, 2004).

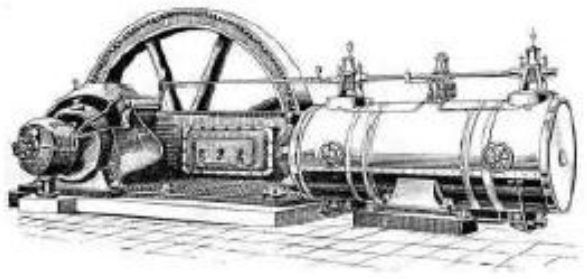


Figura 1: Máquina de vapor. Fuente: González-Hernández et al., 2021

Fue a partir de la Segunda Guerra Mundial cuando se consolidó la conciencia sobre la necesidad de preservar el legado cultural derivado de este proceso de industrialización. Este patrimonio, compuesto por bienes muebles e inmuebles, así como por conocimientos y prácticas asociadas a la producción industrial, comenzó a ser reconocido como un testimonio valioso del pasado (López y Soria Cáceres, 2023 ; Fernández, 2017).

La valorización del patrimonio industrial proporciona una perspectiva única sobre la evolución tecnológica y su impacto en la transformación de las sociedades. Las fábricas, maquinarias, talleres y espacios productivos no solo reflejan una etapa clave del desarrollo económico, sino que también evidencian cómo la industrialización modeló el paisaje urbano y rural (Miranda, s.f.).

En este contexto, una de las líneas más destacadas en la conservación del patrimonio industrial ha sido la rehabilitación y transformación de antiguos espacios industriales en museos, lo que ha supuesto un importante avance en el ámbito de la museografía y la educación patrimonial (González, s.f.).

Actualmente, el discurso museológico se orienta hacia la participación activa del público, incorporando tecnologías emergentes como la realidad aumentada y la realidad virtual. Estas herramientas ofrecen experiencias más inmersivas, accesibles e interactivas, que permiten nuevas formas de comprensión, conservación y divulgación del patrimonio industrial (EVE Museografía, 2023).

En este contexto de revalorización y difusión del patrimonio industrial, destaca la labor que viene desarrollando el grupo de investigación iFAB "Ingeniería de Fabricación" (TEP933), de la Universidad de Málaga, cuya actividad se centra, entre otras líneas, en la recuperación, documentación y puesta en valor del patrimonio industrial vinculado a los procesos de fabricación. Esta línea de trabajo se desarrolla en coordinación con iniciativas de ámbito nacional, a través de los grupos de trabajo "Patrimonio Industrial de Fabricación" (PATRIF) de la Sociedad de Ingeniería de Fabricación (SIF), y "Historia de la Ingeniería Mecánica, Máquinas y Mecanismos" (HIM3) de la Asociación Española de Ingeniería Mecánica (AEIM).

En consecuencia con estas acciones, el presente trabajo se alinea con los objetivos del Proyecto de Innovación Educativa "Desarrollo y empleo de herramientas docentes a través de la creación del Museo Virtual de la Escuela de Ingenierías Industriales de la Universidad de Málaga" (PIE22-94). El entorno virtual desarrollado en este Trabajo Fin de

Máster integrará una selección de máquinas-herramienta históricas, accesibles mediante tecnologías inmersivas como la realidad virtual, permitiendo su consulta interactiva y contribuyendo así a la conservación digital del patrimonio industrial, así como a su difusión con fines educativos y divulgativos.

1.2. Objetivos

El objetivo principal de este proyecto de fin de máster es el diseño y desarrollo de un entorno virtual interactivo que permita la exposición y valorización del patrimonio industrial, incluyendo las máquinas-herramienta históricas modeladas por otros alumnos en diferentes trabajos fin de estudios de la Escuela de Ingenierías Industriales de la Universidad de Málaga. A través del uso de tecnologías de realidad virtual (VR), se busca ofrecer una herramienta accesible, inmersiva y educativa que contribuya a la conservación, difusión y comprensión de este tipo de patrimonio técnico e industrial.

El presente proyecto parte de la necesidad de encontrar nuevas formas de presentar y preservar el patrimonio industrial, dada la dificultad que presentan muchos de estos elementos, por su tamaño, fragilidad o ubicación, para ser trasladados o expuestos en museos físicos. A ello se le suma el hecho de que, en muchos casos, no se han conservado físicamente los modelos originales, o únicamente se dispone de información documental, lo que limita aún más su accesibilidad y visibilidad. En este sentido, la realidad virtual se presenta como una solución idónea, ya que permite recrear a través de una experiencia inmersiva, tanto el aspecto como el funcionamiento de estas piezas, facilitando su estudio, contemplación y divulgación en contextos académicos, museísticos o formativos.

Dentro del presente trabajo, se detallará el software utilizado y el motivo por el cual ha sido seleccionado para el desarrollo de la aplicación. Asimismo, se describirá en profundidad todo el proceso técnico seguido para la obtención del resultado final, incluyendo el procedimiento de creación e instalación de la aplicación en las propias gafas de realidad virtual. Este enfoque permitirá garantizar la portabilidad y facilidad de uso del entorno virtual, haciendo viable su utilización en distintos contextos educativos y de divulgación.

Así se pretende obtener un museo virtual de máquinas-herramienta, permitiendo la interacción del usuario para observar el movimiento de cada una de ellas a través de las gafas de realidad virtual. Este entorno interactivo será complementado con pósteres informativos para cada máquina, en los que se recogerá la información más relevante sobre sus características, historia y funcionamiento. Esta combinación busca enriquecer la experiencia del usuario, combinando el componente visual e inmersivo con el informativo y didáctico.

El objetivo general que se ha planteado se puede desarrollar en los siguientes objetivos específicos:

- Realizar la conversión y adaptación de modelos tridimensionales de máquinas-herramienta para asegurar su compatibilidad estructural y funcional con el motor gráfico Unity 3D.
- Diseñar y desarrollar un entorno de realidad virtual inmersivo en Unity 3D, que represente de forma realista y operativa el espacio expositivo previsto.
- Implementar un museo virtual interactivo dentro del entorno de Unity 3D, integrando los modelos previamente adaptados.
- Configurar las gafas de realidad virtual, incorporando el archivo ejecutable del museo virtual para garantizar su correcta ejecución e interacción inmersiva.

Además, este proyecto se plantea como una propuesta escalable y metodológicamente replicable, lo que permitirá incorporar nuevos modelos digitales y contenidos en el futuro, garantizando así la continuidad y evolución del museo virtual.

2. Aplicaciones de la industria 4.0. en el patrimonio industrial

2.1. Introducción

En los últimos años ha crecido el interés por la conservación y reutilización del Patrimonio Industrial. Al hablar de Patrimonio Industrial nos estamos refiriendo a todo el patrimonio constituido por los restos de la industrialización. La conservación y puesta en valor de estos bienes va ligada no sólo al interés por recuperar las raíces históricas locales, sino porque es un mecanismo fundamental para conocer nuestra sociedad, ya que el patrimonio está ligado a los ciudadanos, es un signo que les caracteriza históricamente, a través del cual se reconocen en su entorno.

Por tanto, su conservación y protección se hacen necesarias para salvaguardar esa herencia cultural e histórica. Hay que tener en cuenta que la sola preservación del patrimonio industrial puede indicar la presencia de una actividad productiva en el pasado, pero en la mayoría de los casos no puede explicar, por sí sola, su funcionamiento o las relaciones existentes entre las partes. Se hace necesario darle un nuevo uso a estos bienes, de forma que se pueda garantizar su pervivencia en el futuro y pueda ser percibido por la sociedad (Grande Álvarez, 2017).

El paradigma de la cuarta revolución industrial, también denominada Industria 4.0, surge como respuesta ante la transformación hacia un mercado globalizado y con una alta competitividad. En un contexto en el que el cliente exige cada vez una mayor calidad y personalización de los productos, la Industria 4.0 plantea el uso de herramientas y tecnologías que aumenten la eficiencia y la capacidad de adaptación de los procesos productivos (Orive et al., 2021).

En este sentido, las tecnologías emergentes asociadas a la Industria 4.0 representan una oportunidad única para revitalizar el Patrimonio Industrial. Técnicas como la fotogrametría digital, la realidad aumentada, la realidad virtual y los gemelos digitales permiten

no solo documentar con alta precisión los elementos patrimoniales, sino también facilitar su comprensión, difusión y conservación activa. Como destacan Alexandre et al., 2017, aunque el enfoque inicial de estas tecnologías se dirigía al ámbito industrial, su aplicación es transversal y proyectable al patrimonio artesanal e industrial, ofreciendo oportunidades para agilizar procesos de documentación, personalizar la experiencia del visitante y poner en valor los bienes culturales de manera sostenible. La integración de estas herramientas no supone una alteración del bien patrimonial, sino una estrategia para fortalecer su accesibilidad, su interpretación y su viabilidad futura.

2.2. Herramientas de la industria 4.0.

Como se ha mencionado en el apartado anterior, la aplicación de las tecnologías emergentes asociadas a la Industria 4.0 ha supuesto una transformación profunda en numerosos ámbitos, entre ellos el de la conservación, difusión y valorización del patrimonio industrial. Gracias al desarrollo de herramientas como la fotogrametría digital, la realidad aumentada, la realidad virtual o los gemelos digitales, hoy es posible no solo preservar estos bienes de manera más precisa, sino también acercarlos a la sociedad de una forma innovadora y didáctica. A continuación, se expone en detalle cómo cada una de estas tecnologías contribuye a mejorar el conocimiento, la conservación activa y la difusión del patrimonio industrial.

2.2.1. Fotogrametría

En la actualidad, la digitalización ha supuesto una oportunidad para la resolución de los problemas que presenta la fotografía convencional, ya que una fotografía digital se puede almacenar en menor espacio y se puede transmitir con mayor facilidad, sin que ello suponga un coste. Sin embargo, la generación de imágenes obtenidas por estas técnicas, como se muestra en la Figura 2, es una representación en dos dimensiones (2D), requiriéndose numerosas imágenes para poder recopilar toda la información que aporta un bien. En este sentido, la generación de modelos en tres dimensiones (3D) permite su registro de forma completa en un único elemento digital (Trujillo et al., 2022).

Para la generación de un modelo digital, tradicionalmente ha sido necesaria la toma de medidas para poder trasladarlas en el dimensionamiento del elemento a través de un software de Diseño Asistido por Ordenador (Computer Aided Design, CAD). Rojas-Sola et al., 2013 realizó el modelo tridimensional de una prensa de aceite a partir de una medida directa del elemento, o bien, aplicando la misma metodología generó el modelo 3D de una antigua cosechadora. Sin embargo, esta metodología puede representar una elevada complejidad en bienes que tengan numerosos elementos con elevado nivel de detalle (Martín Béjar et al., 2023).

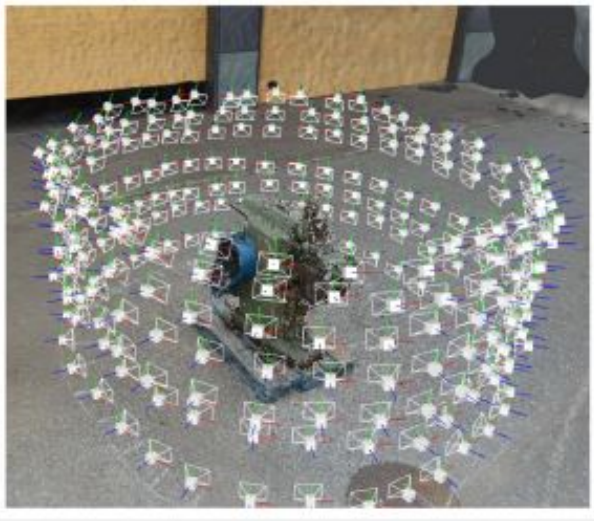


Figura 2: Fotogrametria. Fuente: Botella Ortega, 2021

En el ámbito del patrimonio industrial, se ha utilizado esta técnica para la generación de modelos de diferentes bienes industriales, en los que, a partir de un vehículo aéreo no tripulado, se ha obtenido un conjunto de imágenes aéreas del bien para su posterior tratamiento y generación del modelo digital (Martín-Béjar et al., 2020).

En resumen, la fotogrametría es una técnica de medición y modelado tridimensional que se basa en la obtención de imágenes fotográficas desde distintos ángulos para reconstruir digitalmente un objeto o espacio físico. Mediante el uso de software especializado, como Meshroom o ReCap Pro, estas imágenes se procesan para generar una nube de puntos que representa con alta precisión la geometría del objeto capturado. Esta técnica

permite obtener modelos digitales detallados, útiles para aplicaciones como la documentación, restauración, reproducción física mediante impresión 3D o su integración en entornos de realidad virtual y aumentada (Martín Béjar et al., 2023).

2.2.2. Gemelos digitales

Diferentes autores y organizaciones, como la plataforma alemana para la industria 4.0, definen a los gemelos digitales como una representación digital de las características y del comportamiento o funcionalidad de una entidad física, normalmente en planta (por ejemplo, un brazo robot), como se muestra en la Figura 3. Esta definición está muy vinculada al concepto de sistemas ciber-físicos, que constan de partes físicas (activos en planta), y una parte virtual (su gemelo digital) indicando el estado de la parte física, permitiendo su interacción con otros activos en planta, intercambiando información (Orive et al., 2021).

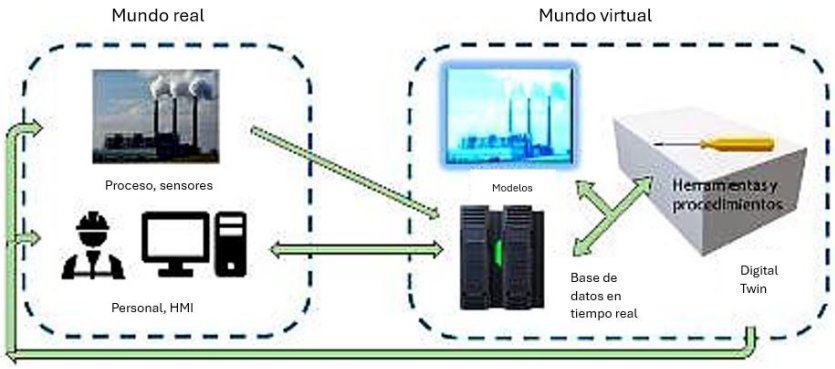


Figura 3: Interacción entre el mundo físico y componentes de un gemelo digital en el mundo virtual . Fuente: Prada et al., 2022

Los gemelos digitales en la actualidad se utilizan en una gran variedad de campos, como por ejemplo, urbanismo, construcción naval, aeroespacial, etc. En el ámbito de un entorno industrial, se define un gemelo digital, en el marco de industria 4.0., como una base de datos con información coherente y en línea enviada desde el proceso, un conjunto de elementos de un entorno capaces de ”hablarse” entre sí a través de algún tipo de comunicación, y un conjunto de herramientas para diversas tareas como podría ser optimización o análisis de datos (Prada et al., 2022).

En los últimos años, la aplicación de gemelos digitales ha comenzado a extenderse más allá del entorno puramente industrial, encontrando un campo de aplicación especialmente prometedor en la conservación y gestión del patrimonio industrial. Los gemelos digitales no solo replican el estado físico actual de un bien patrimonial, sino que también pueden integrar datos históricos, ambientales y de uso para apoyar la toma de decisiones en conservación preventiva (Arias et al., 2022; Culturalab, 2022).

Además, proyectos recientes en entornos industriales históricos han demostrado que la digitalización mediante gemelos digitales puede mejorar significativamente la eficiencia en la planificación de intervenciones, reducir costos operativos y preservar la memoria técnica de infraestructuras que han sido clave en la evolución industrial de distintas regiones. Esta evolución tecnológica representa una sinergia entre la industria 4.0 y el patrimonio, permitiendo no solo la protección de elementos materiales, sino también la difusión educativa y cultural de dichos espacios a través de plataformas digitales interactivas (Prada et al., 2021, 2022).

2.2.3. Realidad Aumentada

En 1990, Thomas Caudell y David Mizel, dos ingenieros de la compañía Boeing, diseñan y crean un prototipo consistente en unas gafas transparentes que combinaban la detección de elementos del mundo real y el posicionamiento de la cabeza, fue lo que se llamó Realidad Aumentada (RA) por primera vez en 1992. Aunque esta tecnología ya se había usado antes de ese año, en la Universidad de Columbia, que utilizaba un HMD para presentar el manual de usuario de forma tridimensional sobre las impresoras para ayudar en los procesos de mantenimiento. (Navarro et al., 2018).

Estas aplicaciones demostraban el potencial de la realidad aumentada para mejorar la interacción con dispositivos en una variedad de contextos prácticos, lo cual ha avanzado y llega hasta nuevos días.

A partir de esas primeras aplicaciones pioneras, la Realidad Aumentada (RA) ha evolucionado hasta convertirse en una herramienta poderosa dentro de diversos ámbitos, entre ellos la gestión del patrimonio histórico arquitectónico. Como se señala en (Bolaños, 2019), la RA permite comunicar de manera efectiva la información propia de los objetos arquitectónicos patrimoniales con el usuario, mediante iconografía tridimensional. Esta tecnología no solo facilita la interpretación del entorno construido, sino que también actúa como un recurso didáctico, útil tanto para los equipos técnicos encargados de los procesos de restauración como para fines educativos en escuelas y universidades.

En este contexto, la implementación de aplicaciones móviles con Realidad Aumentada permite al visitante de un sitio histórico experimentar reconstrucciones virtuales del patrimonio, superpuestas en tiempo real sobre el espacio físico. De este modo, se favorece tanto la conservación como la difusión del patrimonio cultural, al permitir que los usuarios visualicen los edificios en su forma original o en diferentes etapas históricas. El contenido generado mediante modelos 3D y localización puede ser un insumo clave en los procesos de intervención arquitectónica, así como una herramienta para involucrar a la comunidad en la valoración de su propio entorno urbano. (Bolaños, 2019).

Además de su aplicación en el ámbito patrimonial, la Realidad Aumentada ha demostrado ser una herramienta clave en sectores industriales, particularmente en los procesos de mantenimiento y formación técnica. Gracias a la superposición de información digital en el entorno real —como advertencias, instrucciones, modelos 3D o historiales de reparación—, los operarios pueden recibir asistencia guiada en tiempo real, lo que reduce significativamente los errores y el tiempo de inactividad durante averías. Esta tecnología, que se puede observar en la Figura 4, no solo mejora la eficiencia en el diagnóstico y reparación de equipos, sino que también facilita el aprendizaje (Tolsan, 2022).

Figura 4: Aplicación móvil para RA . Fuente: Tolsan, 2022

2.2.4. Realidad Virtual

Según Escartín, 2023, la realidad virtual es una simulación de un ambiente tridimensional generada por ordenadores, en el que el usuario es capaz tanto de ver como de manipular los contenidos de ese ambiente. Con el propósito de alcanzar una sensación de realidad creíble, los ordenadores deben ser capaces de calcular y visualizar la información sensorial lo suficientemente rápido para engañar a los sentidos del participante.

Este concepto, aunque se comenzó a llamar así en los años 80, tiene más tiempo del que en principio nos pudiésemos imaginar, pues las primeras investigaciones en este campo se hicieron en los años 50 y las primeras máquinas de realidad virtual estuvieron disponibles a principios de los 60.

Las primeras máquinas estaban basadas en la estereoscopía o visión estereoscópica, que consiste en disponer de dos imágenes desde dos puntos de vista distintos, como en la Figura 5, cada una de las cuales se correspondería al punto de vista de uno de los ojos y que al mostrarle a cada ojo su correspondiente imagen por separado, el cerebro las ordena creando la sensación de tridimensionalidad (Navarro et al., 2018).

Figura 5: Estereoscopía en vistas 3D . Fuente: Navarro et al., 2018

Partiendo de esta técnica, aparecieron las primeras máquinas de realidad virtual. En 1957, se inventó la primera, aunque la patente es de 1962, y esta se llamó Sensorama (6), que permitía disfrutar de experiencias inmersivas de cine en 3D mediante la estereoscópica en color, acompañadas de sonido, vibraciones, emisión de aire... (Navarro et al., 2018).

La trayectoria inicial de la realidad virtual no contemplaba la interacción del usuario con el entorno virtual, limitándose a la representación de imágenes en 3D. No obstante, pocos años después, Ivan Sutherland sentó las bases teóricas para lo que sería un avance revolucionario: un sistema que no solo sumergía al usuario en un entorno virtual generado por computadora, sino que también permitía interactuar con él. En los años siguientes, diversas innovaciones en dispositivos de realidad virtual marcaron un periodo de intensa experimentación y desarrollo (Navarro et al., 2018).

Figura 6: Sensorama. Fuente: Navarro et al., 2018

Aunque esta tecnología ha estado tradicionalmente vinculada al ámbito de los videojuegos, su aplicación se ha extendido progresivamente a otros campos.En particular, la realidad virtual se ha consolidado como una de las soluciones más eficaces para la enseñanza demostrativa mediante la reconstrucción virtual de entornos industriales. La recreación de escenarios inmersivos para el usuario constituye,así, una metodología altamente eficaz en contextos educativos (López et al., 2023).

En esta línea, autores como Otero y Flores (Otero & Flores, 2011) destacan el potencial de la realidad virtual como herramienta pedagógica, especialmente en contextos de educación informal como museos y espacios públicos. Gracias a su capacidad inmersiva e interactiva, la realidad virtual permite sustituir el clásico “no tocar” por un “toque, por favor”, facilitando el aprendizaje mediante la exploración activa y la interacción directa. Esta tecnología genera una sensación de presencia que implica al usuario emocional y cognitivamente, convirtiéndolo en protagonista del proceso de aprendizaje.

Asimismo, los sistemas de realidad virtual ofrecen experiencias únicas que no podrían reproducirse en el mundo físico, como viajar al interior de una molécula o explorar el fondo marino en un vehículo simulado. Estos entornos virtuales permiten representar conceptos abstractos, escalar visualmente estructuras invisibles al ojo humano, o simular fenómenos complejos en tiempo real. Por ello, Otero y Flores (Otero & Flores, 2011) subrayan la relevancia de la realidad virtual como medio de comunicación de contenidos y como instrumento para fomentar el aprendizaje constructivista, al permitir que el estudiante construya el conocimiento a partir de la experiencia directa.

En este contexto se enmarca el presente trabajo, que pretende contribuir a la preservación y difusión del patrimonio industrial de la Escuela de Ingenierías Industriales de la Universidad de Málaga mediante la creación de una sala expositiva interactiva basada en tecnología de realidad virtual. Este tipo de iniciativas, además de su valor académico, promueven una mayor concienciación sobre la importancia del patrimonio industrial como parte esencial de la historia tecnológica y social (González, s.f.).

3. Aplicaciones informáticas para crear realidad virtual

En la actualidad existen numerosas aplicaciones de software que son adecuadas para la creación de realidad virtual. A continuación se muestran algunas de estas, presentando posibles ventajas e inconvenientes sobre su uso.

3.1. Unreal Engine

Unreal Engine, cuya interfaz se muestra en la Figura 7, es un motor gráfico de desarrollo avanzado creado por Epic Games, ampliamente utilizado en la industria del videojuego, pero también adoptado en sectores como la arquitectura, la medicina o la ingeniería. Su evolución lo ha convertido en una herramienta versátil para la creación de entornos virtuales inmersivos gracias a su capacidad para generar gráficos en tiempo real con un alto nivel de realismo, así como por su compatibilidad con tecnologías como la realidad virtual o la realidad aumentada.

Entre sus principales ventajas destaca su potente motor de renderizado que permite simular espacios con gran fidelidad, así como su sistema de física y simulación, ideal para proyectos que requieran una alta precisión en la representación de comportamientos reales, como ocurre en la creación de realidad virtual.

Figura 7: Interfaz Unreal Engine. Fuente: Epic Games, 2024

No obstante, Unreal Engine presenta también algunas limitaciones. La principal desventaja es su curva de aprendizaje muy profunda, ya que, aunque el motor ofrece potentes herramientas, estas requieren mayores conocimientos técnicos y experiencia previa en desarrollo 3D, por lo que puede resultar complejo para estudiantes o desarrolladores que se inician en el campo de la programación de proyectos visuales e inmersivos.

Además, Unreal suele requerir mayores recursos computacionales, lo que implica un hardware más potente para trabajar con fluidez en grandes proyectos, sobre todo cuando se manejan escenas gráficas de alta densidad o simulaciones en tiempo real.

En conclusión, Unreal es una herramienta de gran potencial para el diseño de experiencias inmersivas; su elección debe ponderarse en función del objetivo del proyecto o trabajo que se quiera realizar. Mientras que su potencia gráfica y versatilidad lo convierten en un referente para proyectos profesionales, sus exigencias técnicas suponen un gran reto adicional que no cualquier persona está capacitada para superarlo (Epic Games, 2024).

3.2. EyeCad VR

EyeCad VR es una aplicación especializada en la creación de experiencias de realidad virtual orientadas al sector de la arquitectura, el diseño interior y la visualización urbanística, como se muestra en la Figura 8. Fue desarrollada por la empresa italiana Digital Atom SRL y permite generar recorridos inmersivos a partir de modelos 3D importados desde plataformas como Revit, ArchiCAD o Rhino. Su enfoque está claramente dirigido a profesionales del diseño que necesitan mostrar espacios arquitectónicos de forma intuitiva, rápida y realista, sin necesidad de conocimientos de programación.

Figura 8: Interfaz EyeCad VR. Fuente: Autodesk App Store, s.f.

Entre sus principales ventajas se encuentra la capacidad de realizar renderizados en tiempo real, la facilidad de navegación en entornos virtuales, la compatibilidad con visores como Oculus Rift o HTC Vive, y una interfaz de usuario sencilla, que permite a arquitectos y diseñadores generar presentaciones interactivas sin necesidad de involucrarse en procesos complejos. También ofrece una app móvil (Eyecad 360) para la visualización remota de proyectos o trabajos realizados.

Sin embargo, EyeCad VR presenta limitaciones importantes frente a motores de desarrollo más avanzados. En primer lugar, su enfoque está muy acotado al campo de la visualización arquitectónica, por lo que no permite una personalización profunda del comportamiento de objetos, la implementación de interacciones complejas ni la integración de sistemas externos. Además, al tratarse de una plataforma cerrada, el usuario queda sujeto a las herramientas predefinidas que ofrece el entorno, lo que limita el desarrollo de aplicaciones interactivas más ambiciosas o adaptadas a objetos educativos específicos.

Otro aspecto relevante es que EyeCad VR no está diseñado como un motor de programación, sino como una herramienta visual, limitando, de nuevo, su uso a campos amplios (Autodesk App Store, s.f.; Digital Atom SRL, s.f.).

3.3. Unity 3D

Unity 3D es un motor gráfico 3D tanto para PC como para Mac que se usa para crear y desarrollar juegos, aplicaciones interactivas, visualizaciones y animaciones. en 3D. La principal ventaja que tiene el uso de Unity 3D es que es accesible por cualquier persona, ya que tiene versiones gratuitas y profesionales. Aunque la versión más completa es la profesional, con la versión gratuita puede ser más que suficiente, sin la necesidad de obtener ningún tipo de licencia (Cardona & Herrera, 2014).

Además del motor gráfico 3D, que aporta muchas funcionalidades, como simular leyes físicas (gravedad, colisiones, animaciones, sonidos...), Unity tiene un editor virtual, combinado con programación en lenguaje C#. El proceso de trabajo básico en Unity 3D consiste en crear objetos desde una barra de herramientas gráficas y, posteriormente, asignarles comportamientos desarrollados en C# que definen cómo se van a comportar los objetos (González-Hernández et al., 2021).

La interfaz principal de Unity 3D, mostrada en la Figura 9, está compuesta por varias ventanas fundamentales como la "Scene", donde se construyen y editan los niveles del juego, y la "Game", que permite visualizar el resultado tal como lo verá el usuario final. A la derecha se encuentra la jerarquía de objetos, que muestra todos los elementos activos en la escena y su estructura, mientras que en la parte inferior está el panel de "Project", donde están almacenados todos los recursos importados o creados dentro del proyecto (scripts, modelos 3D, sonidos,etc...) (Miñarro, 2016).

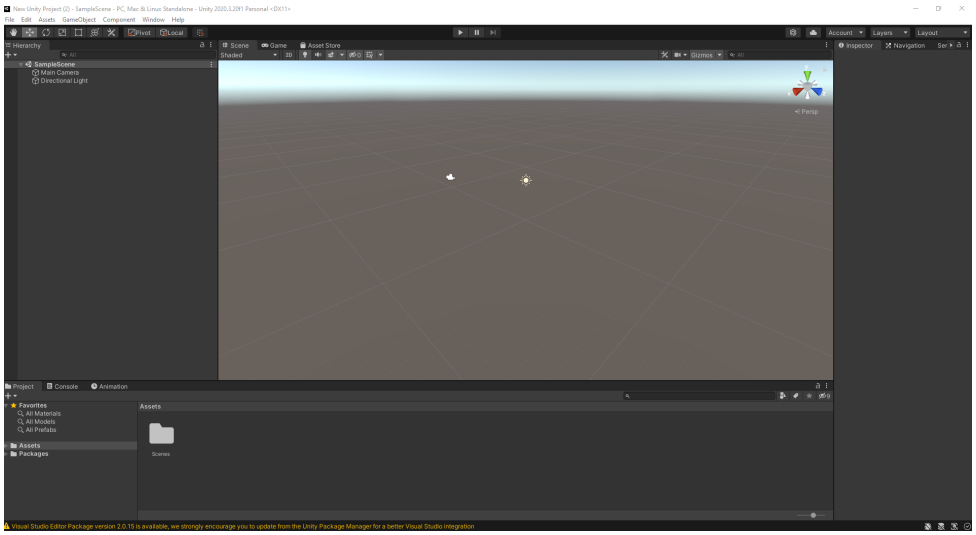


Figura 9: Interfaz Unity 3D. Fuente: Propia

En cuanto a la programación, aunque Unity incorpora por defecto el editor MonoDevelop, en este proyecto se ha utilizado Visual Studio como entorno de desarrollo, ya que ofrece una experiencia más robusta y profesional. Visual Studio permite trabajar con C# (el lenguaje que se usa en Unity) con un soporte avanzado, con sugerencias inteligentes y manejo eficiente de errores. A través de esta herramienta, es posible crear scripts que luego se vinculan a los objetos dentro del editor de Unity, dotándolos de comportamiento dinámico (Miñarro, 2016).

A continuación, se presenta una tabla comparativa (Tabla 1) de las diferentes aplicaciones informáticas mencionadas:

Tabla 1: Comparativa de aplicaciones informáticas para desarrollo de realidad virtual

Apliación informática	**Ventajas principales**	**Inconvenientes principales**
Unreal Engine	▪ Gráficos en tiempo real de alta calidad ▪ Potente motor físico y de simulación ▪ Gran versatilidad en distintos sectores	▪ Curva de aprendizaje pronunciada ▪ Requiere hardware potente ▪ Complejidad para usuarios sin experiencia
EyeCad VR	▪ Interfaz intuitiva y visual ▪ Ideal para arquitectura y diseño interior ▪ No requiere conocimientos de programación	▪ Enfoque limitado a visualización arquitectónica ▪ Poca personalización e interacción avanzada ▪ No permite desarrollos educativos complejos
Unity 3D	▪ Accesible y multiplataforma ▪ Amplia comunidad y documentación ▪ Totalmente personalizable mediante C#	▪ Requiere conocimientos básicos de programación ▪ Calidad gráfica por defecto inferior a Unreal

Dadas las características analizadas de cada una de las herramientas, Unity 3D ha sido seleccionada como la plataforma de desarrollo para este Trabajo Fin de Máster. Esta elección se debe al equilibrio que ofrece entre accesibilidad, personalización y capacidad técnica. Unity permite desarrollar aplicaciones inmersivas complejas mediante scripts en lenguaje C#, a la vez que ofrece una interfaz clara, una gran comunidad de soporte y una curva de aprendizaje asumible incluso para usuarios con experiencia media en programación y diseño 3D.

La versión que se ha usado del editor de Unity 3D en el presente proyecto es la 2022.3.53f1. Es necesario mencionar que para comenzar a crear el proyecto es necesario hacerlo a través del Unity HUB, Figura 10, donde se ha instalado el editor mencionado anteriormente (versión del HUB 3.10.0).

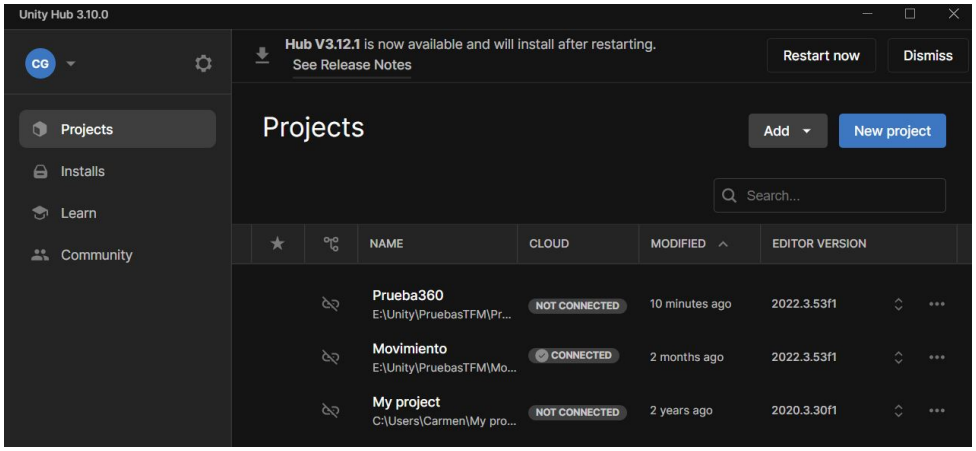


Figura 10: Unity HUB. Fuente: Propia

4. Museo virtual de la Escuela de Ingenierías Industriales de la Universidad de Málaga

En el marco de la formación en ingeniería, la comprensión del Patrimonio Industrial no solo permite conocer la evolución tecnológica de los procesos productivos, sino que también constituye una herramienta clave para contextualizar los retos actuales de la profesión dentro de una perspectiva histórica, social y cultural. En este sentido, la integración de nuevas tecnologías como la realidad aumentada (RA) y la realidad virtual (RV) en entornos educativos está demostrando ser un recurso de gran valor para acercar estos contenidos al alumno de una forma accesible, dinámica e inmersiva.

La Escuela de Ingenierías Industriales (EII) de la Universidad de Málaga ha iniciado en los últimos años una apuesta por la digitalización del patrimonio industrial mediante estas tecnologías con el fin de proporcionar nuevas herramientas tanto de aprendizaje como de experiencia para los usuarios. En este contexto, ha sido creado un museo de realidad aumentada en el que, a través de pósteres impresos con códigos QR, como se muestra en la Figura 11, el visitante puede visualizar modelos tridimensionales de máquinas-herramienta históricas sobre el entorno real mediante un dispositivo móvil.

Figura 11: Museo realidad aumentada EII, UMA

Los museos especializados en técnica e industria son un medio de especial relevancia para la protección, conservación e interpretación del patrimonio industrial. La forma en que las nuevas generaciones se acercan a los contenidos culturales ha cambiado radicalmente en los últimos años, debido a la irrupción de los medios audiovisuales, las nuevas tecnologías digitales y contenidos en línea, que han venido a sustituir a los medios expositivos tradicionales (Trujillo Vilches et al., 2024).

Junto a esta iniciativa, la EII dispone actualmente de un pequeño espacio de realidad virtual, mostrado en la Figura 12, en el que se ha comenzado a experimentar con entornos expositivos más inmersivos. Mediante el uso de gafas de RV, los usuarios pueden acceder a una sala virtual donde se encuentran algunas de las máquinas modeladas por antiguos alumnos de la escuela. Aunque esta primera versión de museo de realidad virtual no ha sido tan amplia ni ha abarcado todas las máquinas del patrimonio industrial con las que se cuenta, como el museo de realidad aumentada que ya existe, ha servido como prueba para observar el potencial que tiene este tipo de herramientas y ha despertado un notable interés.

Figura 12: Espacio de realidad virtual EII, UMA. Fuente: Ávila, 2023

El uso de gafas de realidad virtual es un complemento de notable interés para su uso en exposiciones temporales y en dinámicas didácticas en el aula de la educación reglada o cualquier otro entorno de aprendizaje. Permite al usuario de este tipo de experiencias disfrutar de otros escenarios de la historia a través de la recreación de un mundo paralelo

construido en el entorno virtual (López et al., 2023).

Como evolución natural de los avances mencionados, se ha proyectado la creación de un nuevo museo de máquinas-herramienta en realidad virtual, que consistirá en una sala expositiva completamente inmersiva simulando el espacio real de la Escuela de Ingenierías Industriales. El visitante, mediante el uso de gafas de RV, podrá recorrer un espacio virtual que reproduce el vestíbulo del centro, transformado en un entorno museístico donde se exponen modelos digitales de máquinas-herramienta de distintas épocas.

Cada máquina estará acompañada de dos elementos fundamentales:

- Un panel de control interactivo, que permitirá activar y visualizar los diferentes movimientos característicos de cada máquina. Esos movimientos se podrán activar a través de botones de ON-RESET.
- Un panel informativo, donde se presentarán los datos más relevantes de cada modelo: denominación, año, contexto de uso y otras curiosidades o datos técnicos. Esta información estará disponible mediante ventanas emergentes.

Ambos elementos se muestran en la Figura 13.

Figura 13: Paneles de interacción en el museo virtual. Fuente: Propia

El nuevo museo permitirá, por tanto, consolidar un entorno inmersivo de alta calidad, contribuyendo a la difusión y conservación digital del patrimonio industrial, favoreciendo el acceso a estos recursos desde cualquier parte.

Por último, se debe mencionar que este museo ha sido dotado con seis modelos digitales de máquinas-herramienta que se describirán en detalle en el siguiente punto.

5. Máquinas-herramienta incluidas en el museo virtual

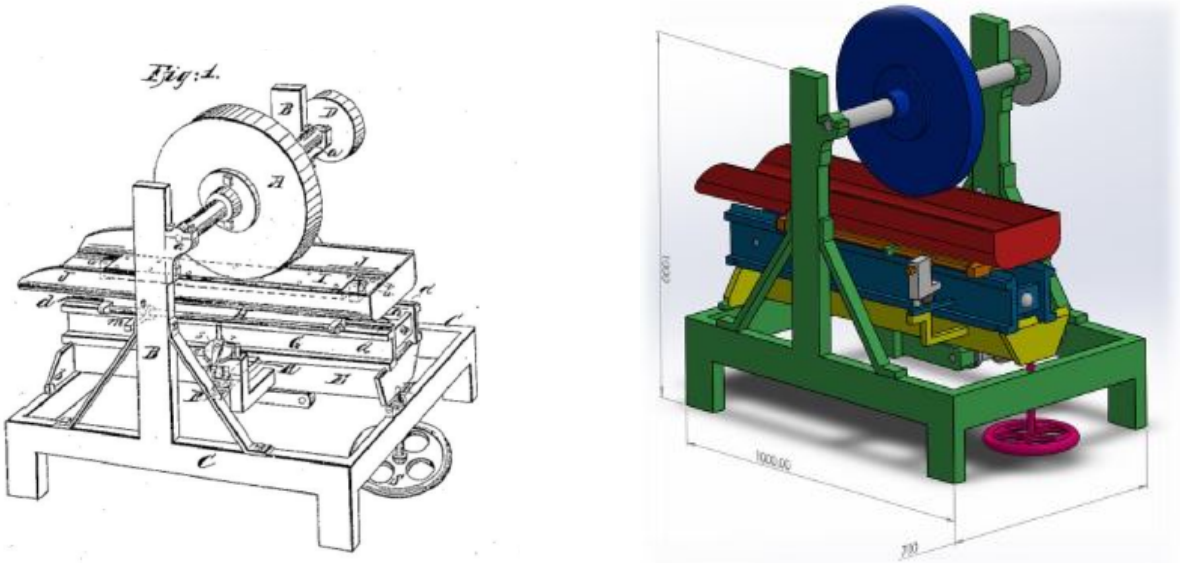
En este apartado se presentarán las principales características de las máquinas-herramienta, su relevancia en la producción industrial y su evolución a lo largo del tiempo. Se abordarán aspectos clave que definen su funcionamiento y contribución a los procesos productivos.

Aunque se tienen más máquinas digitalizadas en el museo virtual, en este caso se han seleccionado aquellas que forman parte del grupo de máquinas-herramienta pertenecientes al mismo contexto histórico: mediado y finales del siglo XIX y comienzos del siglo XX. A medida que el museo crezca, se crearán nuevas salas dedicadas a distintos periodos de la Historia, siendo esta una de las salas más representativas por su relevancia en el desarrollo industrial.

5.1. Rectificadora plana Darling

La rectificadora de superficies planas con eje horizontal (Figura 14), patentada por Samuel L. Darling en el año 1853, representa uno de los avances más significativos en la evolución de las máquinas-herramientas durante el siglo XIX. Su invención marca el inicio de la rectificación de precisión sobre superficies planas como un proceso independiente y especializado dentro del mecanizado por abrasivos, rompiendo con la tradición de utilizar el rectificado únicamente como una operación secundaria o de acabado en máquinas como el torno o la fresadora.

Antes de su aparición, los trabajos de mecanizado sobre superficies planas se realizaban fundamentalmente con planeadoras, como la de Roberts (1830), las cuales, aunque eficaces, no eran capaces de alcanzar la precisión requerida para ciertos componentes industriales. Esta necesidad creciente de exactitud —motivada en gran parte por el auge de sectores como la relojería, el ferrocarril y el armamento— llevó al desarrollo de máquinas más robustas y especializadas como la de Darling.



(a) Rectificadora plana Darling. Fuente: Woodbury, 1958

(b) Modelo 3D para museo virtual de la EII de la UMA. Fuente: Martín Vázquez, 2022

Figura 14: Recntificadora plana Darling

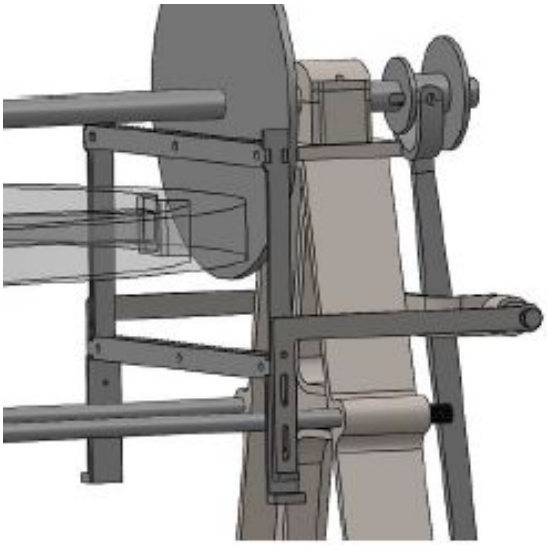
La máquina de Darling introdujo una arquitectura que se considera precursora de las rectificadoras planas modernas. Estaba dotada de un eje horizontal que hacía girar una muela abrasiva, la cual era capaz de realizar cortes precisos sobre la superficie de las piezas fijadas en una mesa desplazable en ambos sentidos: longitudinal y transversal. Este movimiento se realizaba mediante un sistema de guías en 'V' y mecanismos de cremallera, asegurando una alta estabilidad y repetitividad en el proceso de mecanizado.

La relevancia de esta máquina va más allá de su aportación tecnológica, ya que simboliza un cambio de paradigma en la concepción del rectificado como un proceso clave en la fabricación de componentes de alta precisión. Así, la rectificadora de Samuel L. Darling puede considerarse como un auténtico punto de inflexión en la historia de las máquinas-herramientas, cuyas bases técnicas siguen presentes en muchas de las soluciones industriales actuales (Martín Vázquez, 2022).

5.2. Prensa de husillo a fricción ”Delalande”

La prensa incluida en este proyecto es una prensa de husillo a fricción de considerable valor histórico e industrial. La máquina se encuentra en el Museo de las Reales Fábricas de Riópar (Figura 15), institución ubicada en los restos industriales de la antigua Real Fábrica de San Juan de Alcaraz, fundada en 1773 por Juan Jorge Graubner bajo el patrocinio de Carlos III. Estas fábricas representaron la primera industria metalúrgica del latón y tumbaga en España, y estuvieron activas durante más de dos siglos, hasta su cierre definitivo en 1996.

(a) Prensa de Husillo del Museo de las Reales Fábricas de Riópar. Fuente: Pérez et al., 2021



(b) Modelo 3D estructura auxiliar presna de husillo. Fuente: Sanchez Linares, 2021

Figura 15: Prensa de Husillo

Hoy en día, el edificio que alberga la prensa funciona como un museo industrial y oficina de turismo, y contiene maquinaria histórica representativa de la Revolución Industrial española.

La prensa fue fabricada, presumiblemente, en el año 1808 por la empresa francesa Delalande, reconocida por su producción de maquinaria industrial durante los siglos XVIII y XIX. Su recuperación forma parte de una estrategia de preservación del patrimonio industrial que ha cobrado especial relevancia en España desde la década de

los años 80, con el desarrollo del Plan Nacional de Patrimonio Industrial y la Ley 16/1985 del Patrimonio Histórico Español.

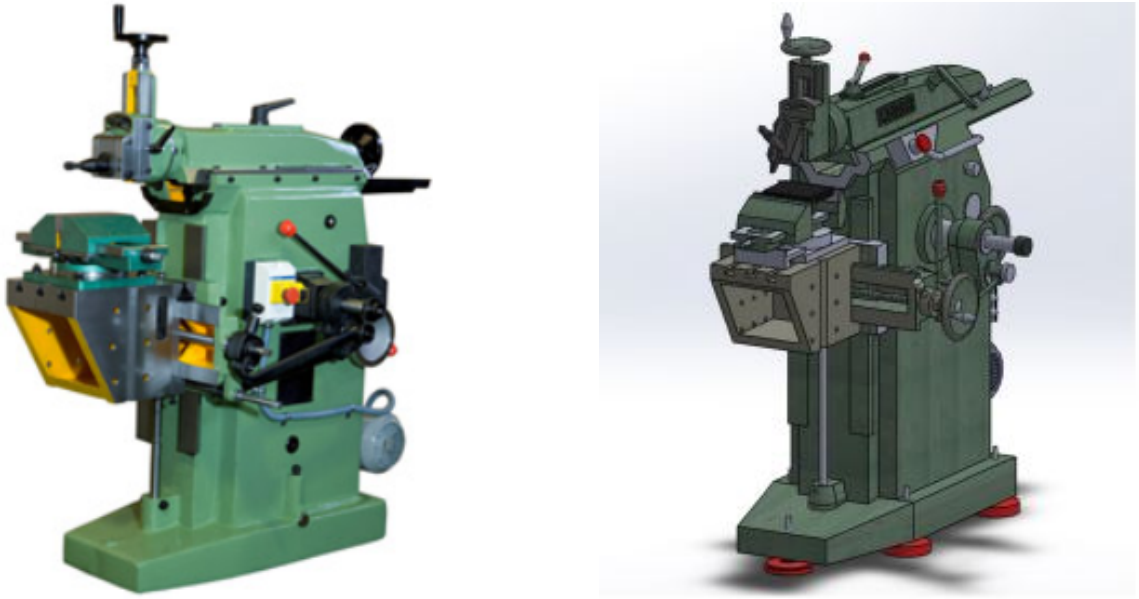
Esta máquina es un tipo de prensa mecánica cuyo funcionamiento se basa en la conversión de la energía de un volante de inercia en un movimiento lineal vertical. Este mecanismo se considera de energía limitada, ya que la deformación de la pieza se produce únicamente con la energía acumulada en el volante. Su diseño clásico incluye:

- Un husillo que transmite la fuerza al carro o portaherramientas.
- Un sistema de discos de fricción que permite el embrague y desembrague del volante.
- Una estructura robusta, formada por una base, bastidor, volante y soporte.

Este tipo de prensas fue comúnmente utilizado durante los siglos XVIII y XIX para procesos de forja, estampado y embutición, especialmente en contextos industriales donde no se disponía aún de sistemas hidráulicos avanzados (Sanchez Linares, 2021).

5.3. Limadora

La limadora mecánica incluida en este proyecto (Figura 17) es una máquina herramienta de la marca ASIDEH, que actualmente se encuentra en el taller del Departamento de Ingeniería de Fabricación de la Universidad de Málaga. Esta pieza, con más de sesenta años de antigüedad, fue donada en los años ochenta por la antigua factoría Santana Motor, ubicada en Linares (Jaén), tras el cierre de una de sus líneas de producción. La máquina fue inicialmente empleada en procesos de mecanizado de componentes automovilísticos y hoy en día se utiliza con fines formativos. Esta trayectoria le confiere un alto valor histórico, siendo representativa del pasado industrial andaluz.



(a) Máquina-herramienta limadora. Fuente: Zubacor, 2024

(b) Modelo 3D para museo virtual de la EII de la UMA. Fuente: Botella Ortega, 2021

Figura 16: Limadora

La empresa que comercializaba esta marca, Insuna S.A., estuvo activa hasta 2006 en Zamudio (Vizcaya) y centraba su actividad en la venta de máquinas herramienta. Actualmente, tanto la empresa distribuidora como la marca han desaparecido, lo que convierte a esta limadora en un ejemplar único y difícilmente reemplazable, lo cual justifica su preservación como patrimonio industrial mueble.

En cuanto a su funcionamiento, la limadora realiza operaciones de mecanizado mediante el arranque de material con una herramienta de corte montada en el carnero, el cual se desplaza de forma rectilínea y alternante. Esta herramienta actúa sobre la pieza fijada en la mesa de trabajo, que cuenta con un sistema de prensas de sujeción. El modelo concreto que aquí se incluye utiliza un sistema de accionamiento por plato-manivela, que convierte el movimiento rotativo del motor eléctrico en movimiento alternativo del carnero. Aunque este tipo de accionamiento no proporciona velocidad constante, tiene la ventaja de reducir vibraciones y esfuerzos mecánicos.

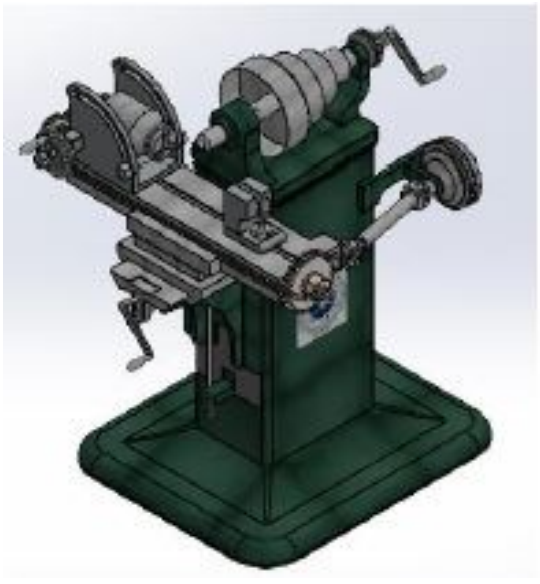
La limadora está compuesta por diversos elementos mecánicos que permiten ajustar

su operación: el volante de variación de profundidad, el mando de bloqueo, el avance automático y manual, y la torreta portaherramientas, entre otros. Esta configuración permite entender el proceso de mecanizado tradicional, previo al uso de tecnologías más modernas como el control numérico computarizado (CNC) (Botella Ortega, 2021).

5.4. Fresadora universal de Brown

La fresadora universal de Frederick W. Howe (Figura **??**) es uno de los hitos más significativos en la evolución de las máquinas herramienta. Fue creada en 1848 para la empresa Robbins & Lawrence y perfeccionada poco después por Brown & Sharpe, quienes introdujeron el plato divisor, elemento que permitió realizar operaciones como el fresado de engranajes rectos y helicoidales. Esta innovación marcó un antes y un después en la historia del mecanizado, al permitir una mayor versatilidad y precisión.

(a) Primera fresadora universal de Brown y Sharpe. Fuente: English School, 2024



(b) Modelo 3D para museo virtual de la EII de la UMA. Fuente: Porras García, 2023

Figura 17: Limadora

Frederick Webster Howe fue un brillante inventor estadounidense, considerado el "Henry Maudslay de América". A lo largo de su vida, participó en numerosos proyectos de mecanizado y armamento, y dejó un legado duradero en la industria. La fresadora universal, una de sus grandes contribuciones, fue concebida con el objetivo de realizar múltiples operaciones sin necesidad de cambiar la máquina o reposicionar la pieza, lo que supuso una drástica mejora en eficiencia y productividad para la época.

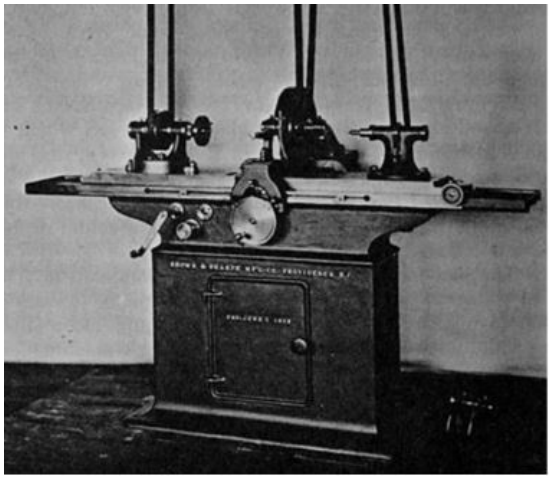
Desde el punto de vista técnico, esta fresadora está equipada con una mesa de trabajo móvil en los tres ejes (X, Y, Z) y un husillo horizontal que, mediante un accesorio especial, puede convertirse en vertical, lo que permite mecanizar en distintos planos. Sus componentes principales incluyen la mesa con ranuras en T, el carro longitudinal y transversal, el cabezal portaherramientas, el plato divisor y otros elementos de sujeción como prensas, mordazas y bloques en V.

En cuanto a su funcionamiento, la fresadora emplea una herramienta rotativa llamada fresa, que se mueve en coordinación con la pieza de trabajo gracias a los distintos sistemas de ejes y guías. Este tipo de máquina puede realizar desde cortes planos y ranuras hasta formas complejas tridimensionales, adaptándose a una amplia gama de procesos industriales. Además de su valor técnico, esta fresadora representa un elemento clave del patrimonio industrial. Esta herramienta permite no solo conservar la memoria de la fresadora, sino también difundirla con fines didácticos y culturales (Porras García, 2023).

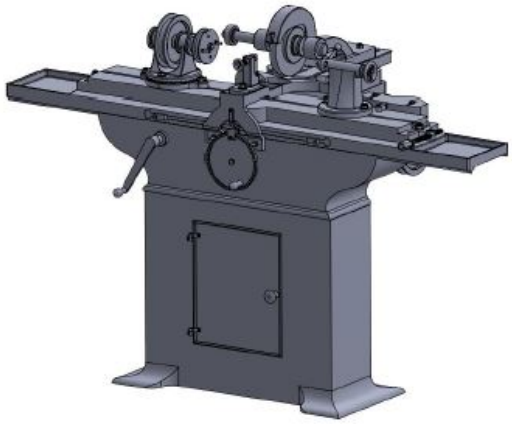
5.5. Rectificadora universal de Brown and Sharpe

La rectificadora universal de Brown and Sharpe, patentada en 1877, Figura 18, supuso una auténtica revolución en la fabricación industrial de precisión. Hasta entonces, las operaciones de rectificado estaban muy limitadas en precisión y versatilidad, al estar integradas en otras máquinas como tornos adaptados. Fue el ingeniero Joseph R. Brown, junto a otros miembros de su empresa como Samuel Darling y Oscar J. Beale, quienes desarrollaron la primera máquina concebida específicamente para el rectificado de precisión.

El primer modelo de rectificadora fue presentado en la Exposición de Máquinas de París en 1876 y, aunque inicialmente no era üniversal", es decir, no podía rectificar interiores, en 1880 se le añadió un accesorio que le permitió alcanzar esa categoría.



(a) Primera rectificadora universal de Borwn and Sharpe, 1876.Fuente: Woodbury, 1958



(b) Modelo 3D para museo virtual de la EII de la UMA. Fuente: El Lahiani Skatou, 2022

Figura 18: Rectificadora Universal

El éxito de esta rectificadora fue inmediato en sectores emergentes como el ferrocarril, la aeronáutica y, especialmente, la industria automovilística. De hecho, su adopción fue tan rápida que en 1891 Brown and Sharpe publicó un Tratado sobre rectificadoras para formar operarios ante la escasez de personal cualificado.

La rectificadora de Brown and Sharpe se basaba en tres movimientos esenciales:

- Movimiento de corte: Generado por la rotación de la muela abrasiva.
- Movimiento de avance: Proporcionado por el posicionamiento preciso de la pieza.
- Movimiento de penetración: Mediante el movimiento progresivo de la muela a la pieza.

Gracias a su diseño modular y robusto, la máquina podía realizar rectificados cilíndricos exteriores e interiores, cónicos e incluso superficiales, características que

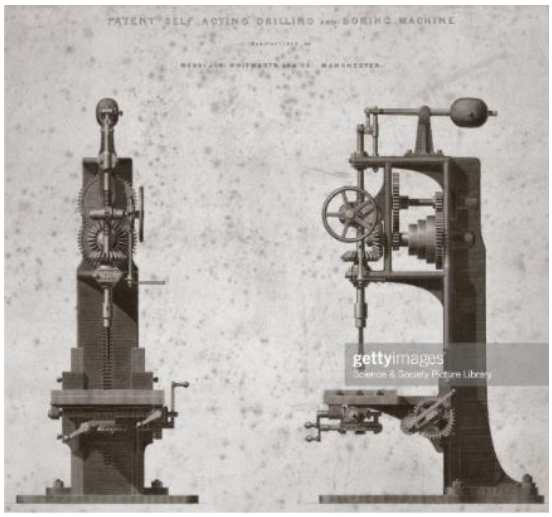
todavía conservan las rectificadoras modernas.

Además, a finales del siglo XIX, ingenieros como Charles H. Norton, tras estudiar las necesidades de la industria automotriz en Detroit, mejoraron este diseño inicial, aumentando el tamaño, el peso y la potencia de las máquinas para permitir el mecanizado de piezas más grandes y con tolerancias aún más estrictas.

Aunque las rectificadoras actuales incluyen mejoras como motores eléctricos, mayor rigidez estructural o sistemas de control numérico (CNC), conservan la misma arquitectura fundamental establecida por Brown and Sharpe en el siglo XIX (El Lahiani Skatou, 2022).

5.6. Taladradora de columna de Whitworth

La Taladradora de Columna diseñada por Sir Joseph Whitworth y patentada en 1847 (Figura 19) es considerada una de las primeras máquinas-herramienta modernas dedicadas al taladrado industrial. Su invención marcó un antes y un después en la historia del mecanizado, especialmente durante la Revolución Industrial, donde la demanda de piezas de alta precisión era cada vez mayor.



(a) Taladradora de columna de Whitworth de 1847. Fuente: Library, 2009

(b) Modelo 3D para museo virtual de la EII de la UMA. Fuente: Rojas López, 2024

Figura 19: Taladradora

Antes de su aparición, las operaciones de taladrado eran poco eficientes y dependían en gran medida de la fuerza humana o de sistemas primitivos de transmisión de potencia. Whitworth introdujo una máquina capaz de aprovechar las fuentes de energía externas (como la máquina de vapor) mediante sistemas de poleas y correas. Esto permitió un salto cualitativo en la precisión, la fuerza y la repetibilidad de las operaciones de taladrado.

Su diseño fue tan influyente que estableció las bases de lo que hoy conocemos como taladradora de columna moderna.

El funcionamiento de la taladradora de 1847 se basaba principalmente en:

- Giro de husillo: Transformación del movimiento horizontal en vertical mediante engranajes cónicos.
- Avance del taladrado: Manual, accionado por una manivela que movía un tornillo sin fin conectado al cabezal.
- Retorno: Asistido mediante el sistema de contrapesos.
- Posicionamiento de la pieza: Ajustes precisos a través de movimientos lineales y rotatorios de la mesa.

En resumen, esta taladradora combinaba por primera vez en una máquina la rigidez estructural, la posibilidad de ajuste fino y el aprovechamiento de potencia externa, elementos esenciales que siguen presentes en los diseños actuales.

Posteriormente, en 1860, la empresa J. Whitworth & Co. introdujo un modelo actualizado conservado en el Museo de Ciencias de Londres, que incluía un sistema de avance automático por correo, cuatro velocidades de trabajo en lugar de tres, mejoras en el sistema de movimientos de la mesa y eliminación progresiva de componentes en madera.

Estas mejoras permitieron un uso más eficiente de la máquina en industrias como la textil, ferroviaria y de armamento, consolidando la importancia de las máquinas-herramienta en el avance de la Revolución Industrial (Rojas López, 2024).

6. Creación del museo de realidad virtual

Para lograr el objetivo de este trabajo, se ha utilizado un enfoque secuencial que integra distintas herramientas de software y hardware específicas. En primer lugar, se han adaptado los modelos tridimensionales previamente diseñados mediante software CAD; a continuación, los modelos se han integrado en un entorno 3D creado en Unity 3D, incluyendo la ambientación del espacio, la programación de interacciones y la mejora para su visualización en gafas de realidad virtual.

Como elemento principal de hardware se han utilizado las gafas de realidad virtual Meta Quest 2, que permiten ejecutar la aplicación desarrollada de forma automática, sin necesidad de conexión constante con el ordenador. Estas gafas han sido fundamentales en el transcurso de este trabajo, ya que han facilitado la validación de cada una de las etapas de creación que se han seguido.

A continuación, se incluye un esquema visual (Figura 20) que representa las principales etapas de desarrollo, las herramientas utilizadas en cada fase y el flujo de trabajo seguido.

Figura 20: Esquema temporal del desarrollo del museo virtual

A lo largo de los siguientes apartados se describen y analizan en detalle las distintas fases desarrolladas en este trabajo.

6.1. Software empleado

Durante el desarrollo de la aplicación del museo de realidad virtual se han empleado diversas herramientas de software, cada una con una función específica dentro del flujo de trabajo. Estas herramientas han sido seleccionadas por su compatibilidad con entornos de realidad virtual y por su versatilidad en tareas de modelado, diseño, programación e implementación. A continuación, se detallan estos programas:

6.1.1. Unity 3D

Las características generales de Unity 3D (versión 2022.3.53f1) como motor gráfico para desarrollo en realidad virtual han sido tratadas en el Apartado 3 del presente trabajo, donde se ha analizado su potencial. En este apartado se describe su aplicación concreta en el desarrollo del museo virtual.

Unity ha sido la herramienta principal del trabajo, utilizada para:

- Configurar un proyecto en realidad virtual compatible con las gafas Meta Quest 2.
- Construir el escenario o entorno 3D, incluyendo elementos como paredes, suelos, techo y luces.
- Establecer interacciones mediante scripts, como el desplazamiento del usuario o la activación de objetos.
- Exportar la aplicación final en formato compatible con Android para su instalación en las gafas de realidad virtual.

La interfaz gráfica de Unity y su motor han permitido desarrollar una experiencia inmersiva fluida, con control total sobre la escena y los elementos que la componen.

6.1.2. Meta Quest Developer Hub

Meta Quest Developer Hub (versión 5.4.0), a partir de ahora MQDH, es una aplicación que ha sido empleada en el desarrollo del museo virtual como herramienta principal para conectar, gestionar y probar la aplicación obtenida de Unity 3D en las gafas de realidad virtual Meta Quest 2. MQDH es una plataforma oficial proporcionada por Meta, diseñada para facilitar el flujo de trabajo de los desarrolladores que crean aplicaciones para dispositivos de realidad virtual, ofreciendo una interfaz intuitiva y funcionalidades específicas para el desarrollo y depuración de aplicaciones en entornos inmersivos (for Developers, 2024).

La funcionalidad principal de MQDH en el marco de este trabajo ha sido la de conectar las gafas al ordenador, mediante USB, y permitir la transferencia directa del archivo ejecutable (.apk) generado en Unity. Esta operación se ha realizado a través de la interfaz gráfica del programa, que detecta automáticamente el dispositivo conectado mediante USB, como se puede observar en la Figura 21.

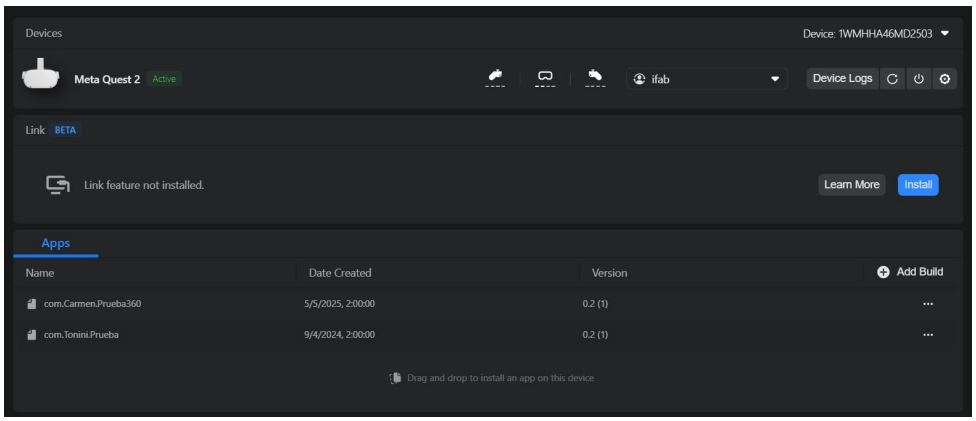


Figura 21: Interfaz Meta Quest Developer Hub .Fuente: Propia

Una vez conectadas las gafas, MQDH permite arrastrar y soltar el archivo .apk generado desde Unity directamente en el panel correspondiente mostrado en el dispositivo. En la Figura 21 se muestra cómo aparecen las aplicaciones cargadas en el dispositivo, incluyendo el nombre de la aplicación, la fecha de instalación y la versión correspondiente.

Entre las ventajas de utilizar MQDH destacan:

- La facilidad de uso gracias a su interfaz gráfica intuitiva.
- La compatibilidad directa con entornos como Unity.
- La posibilidad de verificar el estado de las gafas y los controladores en tiempo real (batería, conexión, actividad).
- La gestión centralizada de múltiples aplicaciones instaladas en el dispositivo.

6.1.3. Blender

Blender es un software de código abierto especializado en el diseño y edición de gráficos tridimensionales. Se ha consolidado como una de las herramientas más versátiles en el ámbito de la creación 3D, tanto en la industria creativa como en el entorno académico y científico.

Entre sus funcionalidades más destacadas se encuentran:

- Modelado 3D: Blender ofrece herramientas avanzadas de modelado poligonal, digital y paramétrico, lo que permite crear geometrías complejas con gran precisión. Su capacidad para importar y optimizar modelos en múltiples formatos lo hace ideal para proyectos que requieren integración con otras plataformas, como Unity (Vázquez et al., 2020).
- Renderizado: Incorpora dos motores de renderizado: Cycles, basado en trazos de rayos, que permite obtener imágenes fotorrealistas de alta calidad, y Evee, un motor en tiempo real más ligero, ideal para previsualización interactiva y proyectos que requieren rapidez en la generación de imágenes (Kawamura & Tanaka, 2021).
- Ventajas generales: Blender destaca por ser gratuito, multiplataforma, altamente personalizable y por contar con una comunidad activa que contribuye continuamente a su desarrollo. Además, permite automatizar procesos mediante scripts en Python, lo que lo hace especialmente útil en entornos de investigación o simulación.

Para ilustrar su entorno de trabajo, en la Figura 22 se muestra una captura de la interfaz principal de Blender, donde se observan las herramientas de modelado y renderizado integradas. La versión utilizada en este trabajo ha sido la 4.3.0.

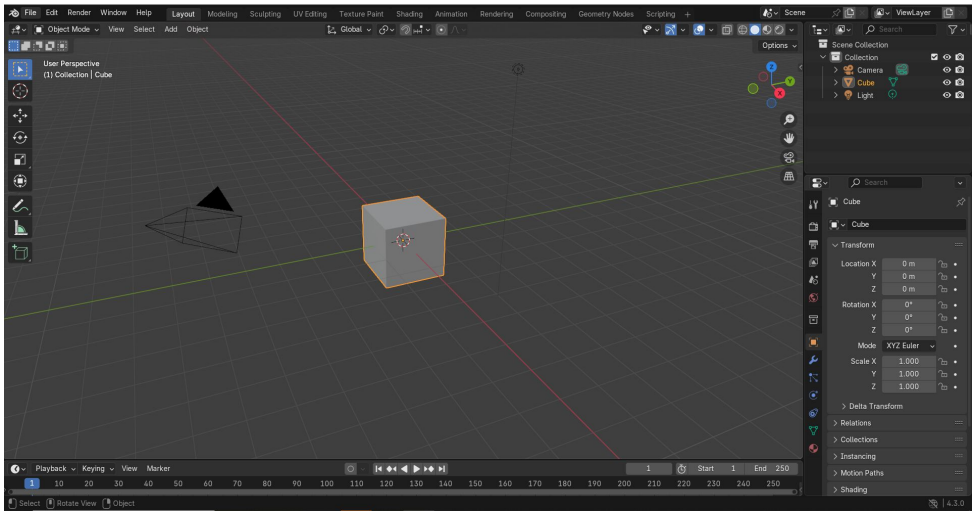


Figura 22: Interfaz Blender. Fuente: Propia

Gracias a estas características, Blender ha sido ampliamente adoptado en áreas como la visualización científica, la realidad virtual y el desarrollo de prototipos digitales, ofreciendo una solución profesional sin necesidad de licencias comerciales.

6.1.4. Visual Studio

Visual Studio es un entorno de desarrollo integrado ampliamente utilizado para la programación en diversos lenguajes, destacando especialmente en el desarrollo de aplicaciones con Unity 3D mediante el lenguaje C#. Su integración con Unity se facilita a través de la extensión Visual Studio Tools Unity (VSTU), que proporciona herramientas específicas para la creación y depuración de scripts en proyectos de Unity (Microsoft, 2023).

La integración entre Visual Studio y Unity a través de VSTU proporciona funcionalidades específicas que optimizan el proceso de programación:

- IntelliSense especializado, que ofrece autocompletado inteligente y sugerencias adatadas a Unity, facilitando la escritura eficiente y sin errores.
- Depuración en tiempo real, permitiendo establecer puntos de interrupción, inspeccionar el estado de las variables y controlar el flujo del prohrama mientras se ejecuta desde el editor de Unity (Microsoft, 2023).
- Navegacion estructurada del proyecto, mediante la cual se accede directamente desde el entorno a los archivos y carpetas que conforman el proyecto Unity.

Estas herramientas han permitido un desarrollo más ordenado, ágil y seguro del entorno interactivo en el presente trabajo. En este caso, Visual Studio, en la versión 2022, ha sido utilizado para programar los scripts necesarios que definen la lógica del entorno del museo virtual. Estos scripts en C# gestionan el comportamiento de los objetos, las interacciones del usuario y las transiciones dentro del entorno de realidad virtual. La capacidad de depurar en tiempo real de Visual Studio es fundamental para detectar y corregir errores, así como para validar la correcta implementación de cada funcionalidad (Technologies, 2021).

Además, Visual Studio presenta una interfaz, Figura 23, clara y personalizable, diseñada para maximizar la productividad del desarrollador. Cuenta con una disposición modular que incluye el panel del explorador de archivos, la ventana de código, la consola de errores y advertencias. En el desarrollo de este trabajo se ha usado la versión de 2022 de Visual Studio.

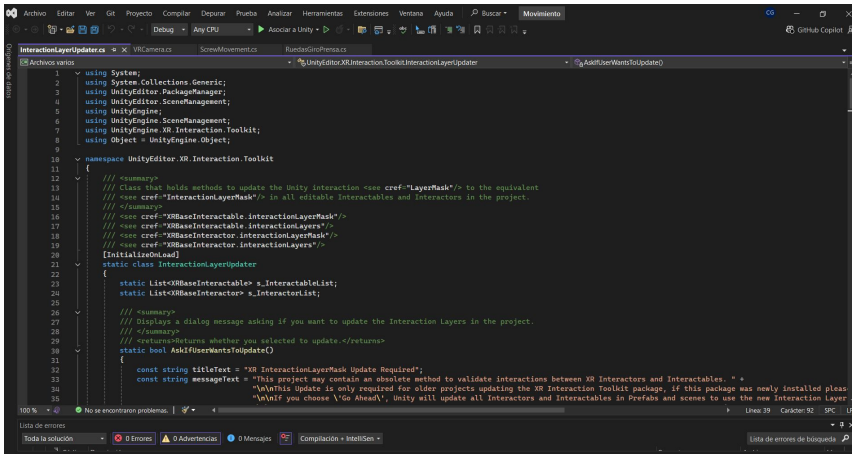


Figura 23: Interfaz Visual Studio. Fuente: Propia

6.2. Exportación modelos CAD

Una de las fases fundamentales en el desarrollo del museo virtual ha sido la incorporación de los modelos tridimensionales de las máquinas-herramienta desarrolladas en apartados anteriores y que han sido diseñadas por estudiantes de la Escuela de Ingenierías Industriales de la Universidad de Málaga, previamente mediante el software CAD SolidWorks. Estos ensamblajes existentes en SolidWorks deben ser integrados en el entorno virtual desarrollado con Unity 3D. No obstante, debido a la incompatibilidad entre los formatos nativos de SolidWorks (.sldprt o .sldasm) y los reconocidos por Unity, ha sido necesario establecer un flujo de trabajo que permitiera transformar y adaptar dichos modelos. Este proceso se ha realizado en tres fases: la exportación de los modelos desde SolidWorks a Blender y, finalmente, su importación y preparación en Unity.

El primer paso ha sido abrir los ensamblajes correspondientes a cada máquina-herramienta en SolidWorks. Como se ha mencionado, Unity no admite directamente los formatos de SolidWorks, por lo que se han exportado en formato .STL. Para ello, una vez abierto el ensamblaje en SolidWorks, se utiliza la opción ”Guardar como” y se selecciona el tipo de archivo .stl de entre todas las opciones, como se muestra en amarillo en la Figura 24.

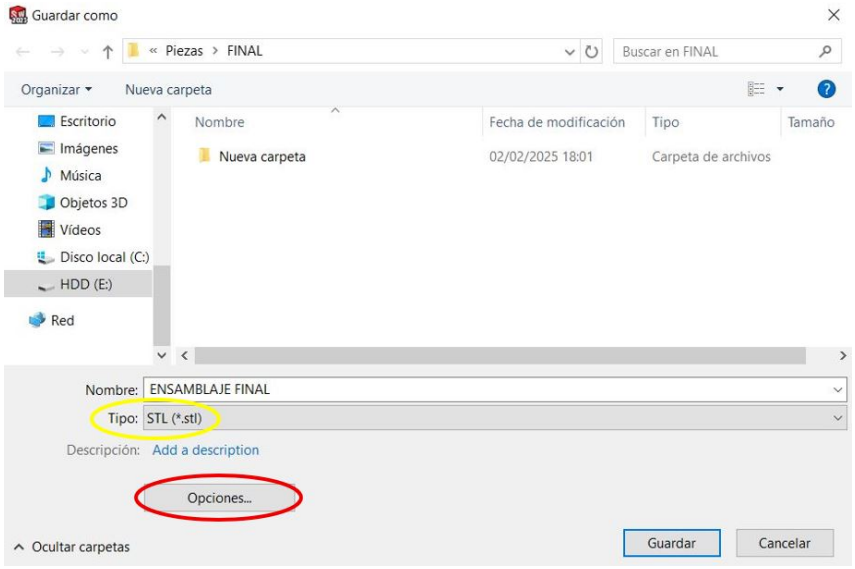


Figura 24: Guardado en .stl y opciones. Fuente: Propia

Un aspecto importante que se debe tener en cuenta en este punto es que, al pulsar la pestaña de ”Opciones” de la Figura 24, se debe desmarcar la casilla ”Guardar todos los componentes de un ensamblaje en un único archivo”, ya que si se dejase marcada la máquina-herramienta se guardaría como un solo objeto, dificultando así su edición posterior para las interacciones y movimientos. Esta casilla se muestra en la Figura 25.

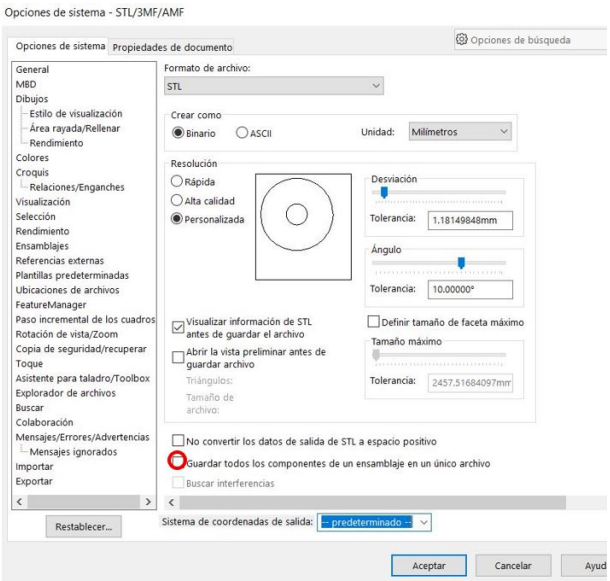


Figura 25: Casilla importante desmarcada. Fuente: Propia

Cuando se han exportado los archivos .stl, se han de importar en Blender, que ha sido la herramienta intermedia para convertir los modelos a un formato compatible con Unity y realizar algunos ajustes adicionales. Para realizar esto, se debe crear un nuevo proyecto y se importa el archivo .stl que se tiene a través de las opciones "Archivo >Importar >.stl", como se muestra en la Figura 26. En este caso, es importante aplicarle una escala de 0.01 a la hora de importar para mantener las proporciones reales de los objetos, ya que los modelos en SolidWorks están escalados en milímetros.

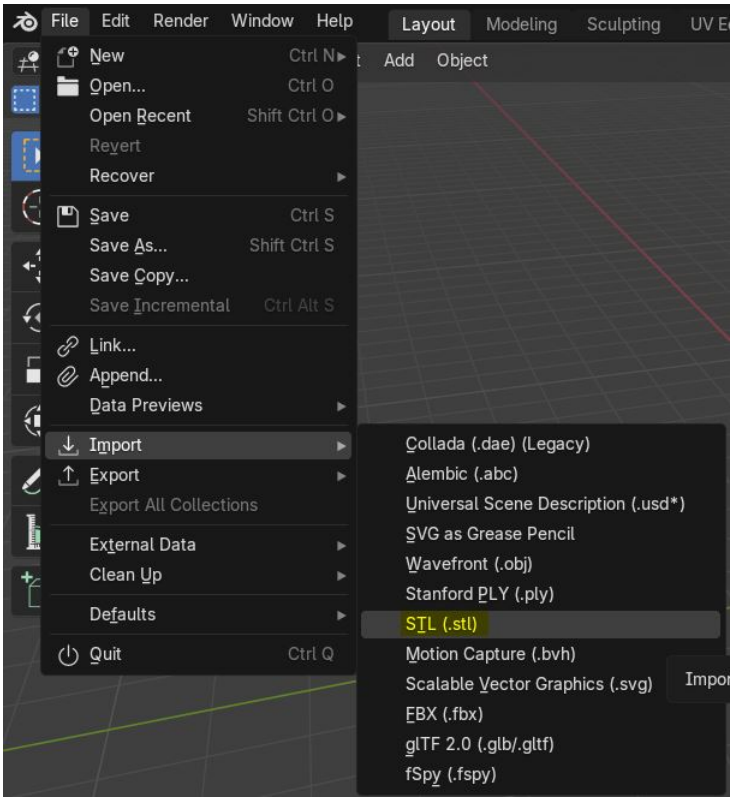


Figura 26: Importar en Blender .stl. Fuente: Propia

Es posible que en esta fase, la máquina-herramienta salga rotada; aunque Blender permite corregir esto, se decide posponer ese ajuste hasta su inserción en Unity, donde el entorno de trabajo es más visual e integrado con el resto del proyecto.

Otro paso importante que se debe realizar en Blender es la definición del origen de coordenadas de cada máquina-herramienta. Esto se realiza seleccionando todos los

componentes de la máquina-herramienta, pulsando sobre el botón derecho y seleccionando ”Establecer origen ->Al centro de masa (volumen)”; así se alinean los centros de cada pieza, lo que facilita su manipulación posterior, especialmente cuando se trate de rotar o animar la pieza en el entorno de Unity. Una vez realizados estos ajustes, se ha procedido a la exportación del conjunto utilizando el formato .obj. Importante en este momento, antes de exportar, marcar la casilla ”Grupos de objetos”, como se señala en la Figura 27, para conservar la estructura de componentes individualizados dentro del archivo resultante.

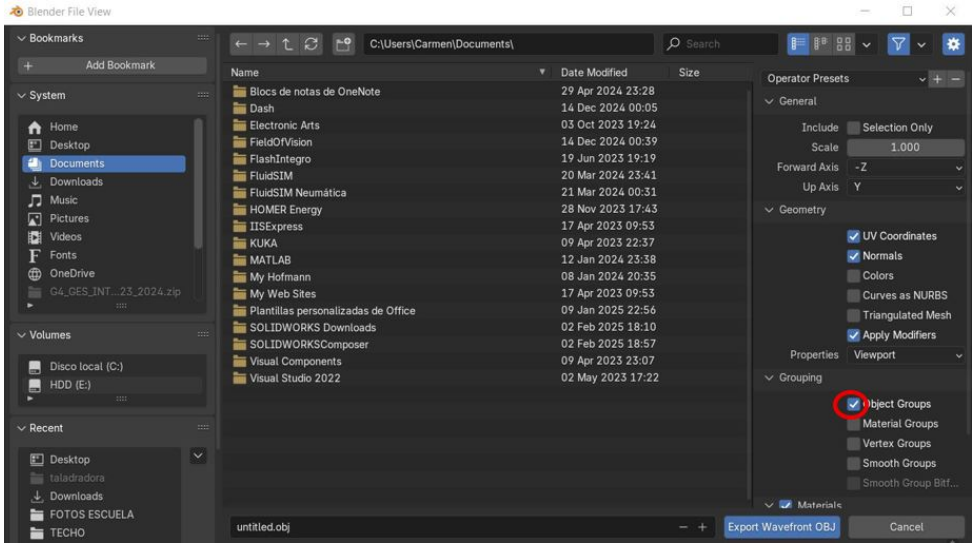


Figura 27: Exportar .obj. Fuente: Propia

El archivo .obj generado es compatible, entonces, con Unity 3D. La integración de este archivo en el proyecto creado en Unity se realiza arrastrándolo directamente desde el explorador de archivos del sistema operativo a la carpeta del proyecto en Unity. Cuando se tiene el archivo dentro del proyecto de Unity, para introducirlo en el entorno, basta con arrastrar el archivo a la jerarquía de Unity. En este punto se realizan los ajustes finales de posición, escala y rotación en caso de que sean necesarios.

Para facilitar la manipulación individual de las piezas, se debe pulsar el botón derecho sobre la máquina-herramienta en la jerarquía y marcar la opción ”Unpack Prefab”, que permite descomponer el modelo en objetos independientes para dotarlos de movimiento

individualmente según sea necesario.

6.2.1. Jerarquizado

En este apartado se explicará la estructuración jerárquica de los componentes de cada máquina dentro de la jerarquía en el entorno de Unity 3D. Este proceso, conocido como jerarquización, consiste en organizar los distintos elementos que conforman cada máquina siguiendo una estructura de árbol, donde se establecen las relaciones de dependencia entre objetos padres e hijos.

La finalidad principal de esta jerarquización es doble: por un lado, permitir una organización clara y lógica de los componentes en el editor de Unity, facilitando así su identificación y gestión; y por otro, establecer relaciones funcionales que reflejen con fidelidad la estructura real de la máquina, permitiendo que los movimientos de cada pieza se comporten de forma coherente según su relación con los demás.

Unity 3D permite representar esta estructura mediante GameObjects animados jerárquicamente. Al situar un objeto hijo dentro de un objeto padre, este hereda automáticamente sus transformaciones, como posición, rotación y escala. Esta característica resulta especialmente útil cuando se desea simular el movimiento de un conjunto en el que varias piezas están unidas físicamente. Al mover el objeto padre, todos los hijos se desplazan con él, manteniendo sus posiciones relativas. Esto permite una manipulación más eficiente y realista de los componentes de la máquina dentro del entorno virtual.

La jerarquización también facilita enormemente la organización del proyecto, permitiendo agrupar componentes por subconjuntos funcionales. Este orden lógico no solo mejora la comprensión de la estructura del modelo, sino que además permite realizar modificaciones o animaciones de forma precisa, actuando sobre grupos completos sin necesidad de ajustar manualmente cada componente individual.

Un ejemplo representativo de esta organización puede observarse en el caso de la taladradora, cuya jerarquía en Unity se muestra en la Figura 28. En esta estructura, el

objeto padre denominado ”5.Taladradora” actúa como contenedor principal de todos los elementos que conforman la máquina-herramienta. Dentro de este objeto se encuentran agrupados varios subconjuntos funcionales. En primer lugar, se encuentra el grupo ”EjeProfundidad”, que incluye el conjunto ”EjeLinealMesa” y ”SoporteMesa”, que será el objeto padre que contenga todos los elementos que se moverán en profundidad. Por otro lado, está el grupo ”EjeHerramienta”, dentro del cual se encuentra el objeto ”EjeVertical”, responsable de guiar el movimiento vertical del portaherramientas o cabezal de la taladradora.

El objeto ”Base”, sin embargo, agrupa los elementos o piezas de la máquina-herramienta que sostienen toda la máquina y que no van a tener movimiento en este entorno. Dentro de la jerarquía de la taladradora también encontramos el objeto ”BotoneraTaladradora” que representa el panel de control virtual de la máquina-herramienta, y el elemento ”MesaINFO” que contiene todo lo destinado a proporcionar información de cada máquina en el entorno del museo virtual.

Figura 28: Jerarquización taladradora. Fuente: Propia

Esta organización refleja la división de los componentes físicos de la taladradora y su integración funcional. Gracias a esta jerarquización, ha sido posible controlar de forma eficaz tanto el comportamiento de las piezas móviles como los elementos auxiliares del entorno, garantizando una simulación coherente, estructurada y accesible dentro del entorno de Unity 3D.

6.2.2. Texturizado

Una vez importados y jerarquizados los modelos 3D de las distintas máquinas-herramienta al proyecto de Unity, se ha observado que estos carecen de texturas o materiales definidos, es decir, no presentan ningún tipo de color o apariencia superficial. Este aspecto dificulta tanto la identificación visual de las distintas partes de cada máquina como la obtención de una representación realista dentro del entorno virtual.

Con el objetivo de resolver esta carencia y dotar a los modelos de una apariencia más cercana a la realidad, se ha procedido a crear manualmente, desde el propio editor de Unity, una serie de materiales básicos que simulan los colores predominantes en cada uno de los modelos 3D de las máquinas-herramienta. Para ello, se consultaron y analizaron los colores principales de cada parte (estructura, componentes móviles, elementos de agarre...).

Los materiales se crearon directamente desde el panel del proyecto en Unity, dentro de la carpeta de Assets. Para ello, se hizo clic derecho sobre el espacio vacío de la carpeta correspondiente y se seleccionó ”Create >Material”. A cada material generado se le asignó un nombre identificativo y, en el apartado de propiedades del Inspector, se definió un color base (Albedo) que imitara los colores reales de las distintas partes de las máquinas-herramienta. No se añadieron mapas de textura adicionales, ya que el objetivo principal era lograr una apariencia clara y funcional. Una vez creados, estos materiales fueron arrastrados y asignados manualmente a los diferentes componentes de cada máquina-herramienta dentro de la jerarquía de la escena.

En la Figura 29 se muestran todos los materiales generados dentro del proyecto, con sus respectivos colores definidos para cada parte característica.

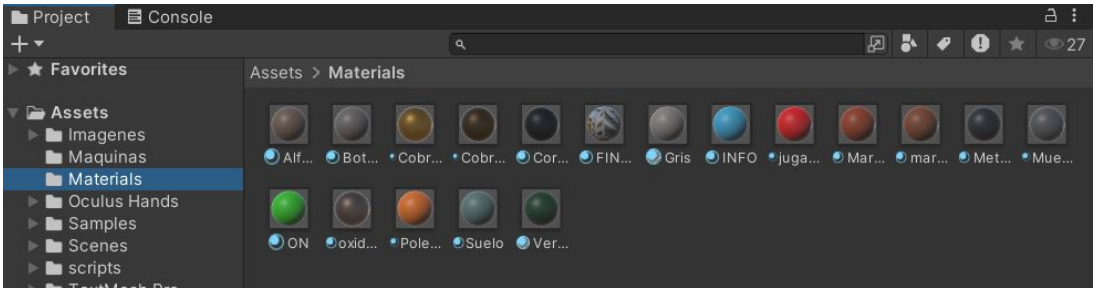


Figura 29: Materiales del proyecto. Fuente: Propia

Con el fin de verificar visualmente la coherencia de los colores aplicados en Unity respecto al aspecto real de sus modelos 3D, se observa una comparativa en las Figuras 30 y 31, donde se observa que, si bien se trata de una aproximación simplificada, los colores seleccionados cumplen adecuadamente con su función representativa y de diferenciación de partes.

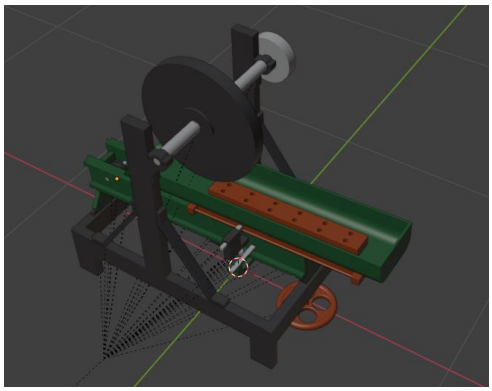


Figura 30: Color modelo 3D en Blender. Fuente: Propia

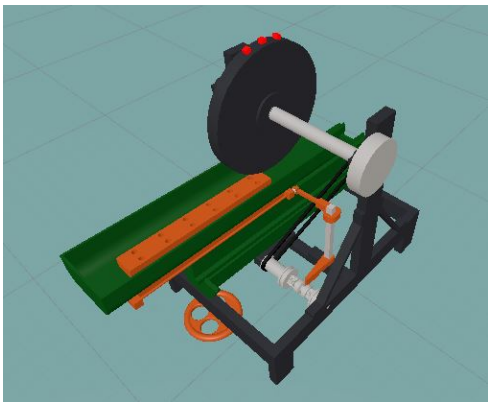


Figura 31: Color en Unity. Fuente: Propia

6.3. Creación del entorno

El entorno del museo se ha llevado a cabo utilizando el programa fSpy junto con su complemento (add-on) correspondiente para Blender, ambos disponibles a través de la página web oficial de fSpy. Una vez descargado e instalado el software, se inicia fSpy y se inserta una fotografía, en este caso, una imagen del vestíbulo de la Escuela

de Ingenierías Industriales, que se muestra en la Figura 32. Es recomendable usar una imagen que contenga profundidad, ya que el objetivo principal de fSpy es configurar los ejes perspectivos de la imagen para su posterior importación en Blender, donde se procederá al modelado del entorno mediante planos.

Figura 32: Vestíbulo EII. Fuente: Propia

Con la imagen cargada en fSpy, se procede a ajustar los ejes X (anchura) e Y (profundidad) a los puntos de fuga identificados en la fotografía. A continuación, se selecciona la opción XY Grid Floor en el menú izquierdo, con el fin de verificar que los ejes han sido correctamente alineados, como se puede observar en la Figura 33. Además, dentro de la configuración de parámetros de la imagen, se selecciona la opción ”From 3rd vanishing point” dentro del menú ”Principal point”, lo cual permite visualizar y ajustar correctamente el eje Z (altura). Todo esto se muestra en la siguiente figura.

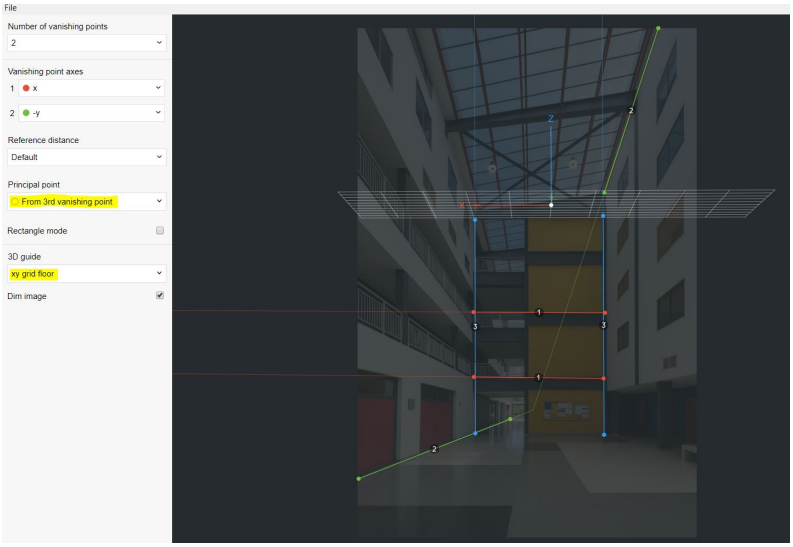


Figura 33: Configuración fSpy. Fuente: Propia

Cuando se ha finalizado esta configuración, la imagen queda lista para su exportación a Blender. Para ello, se selecciona la opción ”Archivo ->Guardar como”, asignando un nombre al archivo generado.

En Blender, con el complemento de fSpy previamente instalado, se procede a importar el archivo desde ”Archivo ->Importar ->.fSpy”, tal como se muestra en la Figura 34. Este proceso posiciona automáticamente la cámara y la imagen en el entorno de trabajo. A partir de este punto, sin alterar la posición original de la cámara importada, se modelan las paredes, el suelo y el techo utilizando planos y operaciones de extrusión, como se ejemplifica en la Figura 35. En caso de cambiar la vista de la cámara, se puede regresar a ella presionando la tecla 0 del teclado numérico.

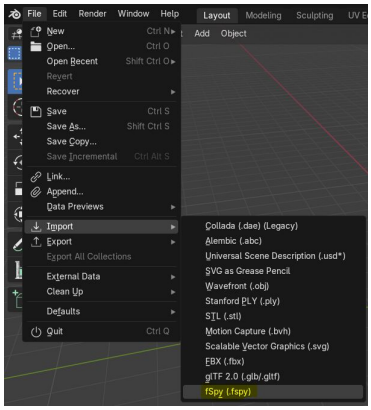


Figura 34: Importar .fSpy.
Fuente: Propia

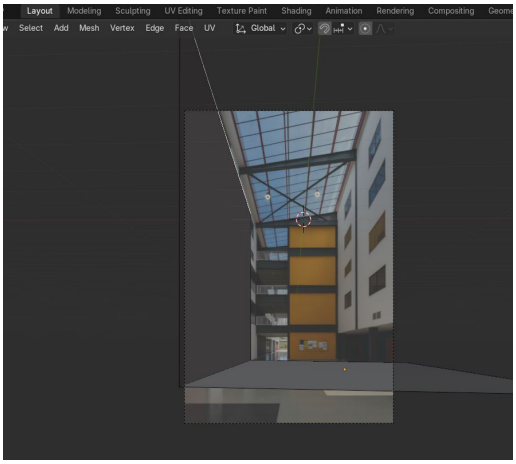


Figura 35: Modelado entorno. Fuente: Propia

Una vez modelados el techo, las paredes y el suelo (a excepción de la fachada principal que será modelada posteriormente en Unity), se procede a proyectar la textura (imagen) sobre los objetos generados. En este paso se selecciona el objeto correspondiente (planos generados) y, en el panel derecho de modificadores, se añaden dos modificadores: en primer lugar, un ”Subdivision surface”, configurado en modo ”Simple” y con cinco niveles de subdivisión; en segundo lugar, un modificador ”UV Project”, que permite aplicar el mapa UV predeterminado asociado a los planos del entorno modelado. Todos estos ajustes aplicados en el panel de modificadores se encuentran reflejados en la Figura 36.

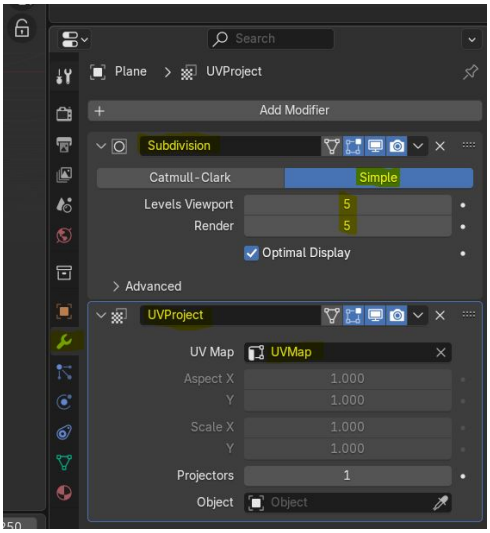


Figura 36: Moddificadores Blender. Fuente: Propia

Finalizado el modelado en Blender, se accede a la pestaña ”Shading” para aplicar la textura. Se crea un nodo ”Image Texture”, donde se selecciona la imagen utilizada en fSpy y se conecta al ”Base Color” del material. La configuración de nodos necesaria se muestra en la Figura 37. Una vez aplicada la textura, el modelo se exporta en formato .obj para su posterior incorporación en Unity 3D.

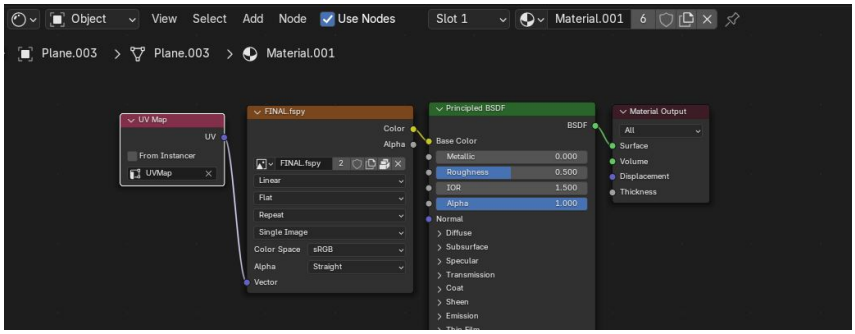


Figura 37: Nodos textura entorno, Blender. Fuente: Propia

Una vez obtenido el archivo .obj correspondiente al entorno, este se importa en Unity mediante su incorporación en la carpeta Assets del proyecto. Junto con este modelo, también se debe importar la imagen utilizada previamente en fSpy. Con ambos elementos ya disponibles en el proyecto, se procede a arrastrar el archivo .obj a la jerarquía de la escena. A continuación, se accede al menú del objeto mediante clic derecho y se selecciona la opción ”Unpack completely”, lo cual permite descomponer el objeto en todos los diferentes elementos que lo conforman (paredes, suelo y techo). Posteriormente, el modelo se posiciona y escala manualmente dentro de la escena con el fin de simular de forma aproximada el espacio correspondiente al vestíbulo de la Escuela de Ingenierías Industriales.

Inicialmente, el entorno aparecerá en color gris, ya que no dispone de ninguna textura asignada en Unity. Para darle textura al entorno, simplemente se arrastra la imagen previamente importada desde los Assets directamente sobre las superficies, o planos, del entorno en la escena. Dado que el modelo 3D del entorno conserva los parámetros asociados a la proyección de la imagen establecidos en fSpy, las texturas se aplican

automáticamente de manera ajustada a las superficies correspondientes, generando una representación visual coherente con la imagen de referencia. El resultado final del entorno recreado se puede observar en la Figura 38, donde se han señalado en la parte inferior el .obj y la imagen dentro de la carpeta Assets del proyecto y, en la parte derecha, el entorno en la jerarquía de la escena.

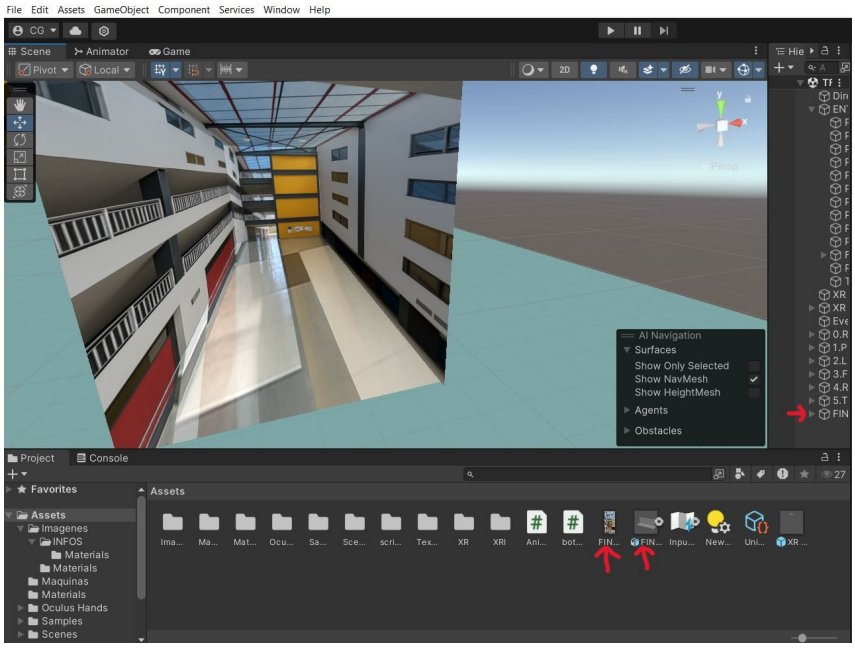


Figura 38: Entorno vestíbulo. Fuente: Propia

Finalmente se ha procedido a la recreación de la fachada principal de la Escuela de Ingenierías Industriales de la Universidad de Málaga. Para ello, se han dispuesto múltiples cubos apilados verticalmente, a los que se les ha aplicado una textura de color gris que simula el revestimiento original del edificio. Asimismo, se ha incorporado una entrada principal en cuya zona interior se ha colocado una exposición de carteles informativos que muestran las máquinas-herramienta que se encontrarán en el interior del museo. En la parte superior de la fachada, se ha añadido un rótulo identificativo con el nombre del museo. El resultado de esta reconstrucción puede apreciarse en la Figura 39.

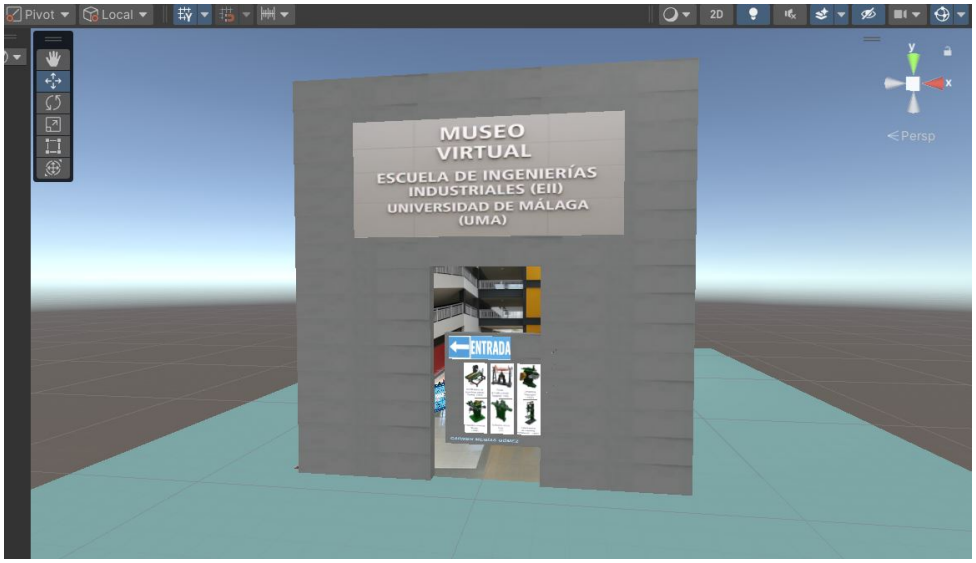


Figura 39: Entorno final. Fuente: Propia

Una vez que se tiene el entorno, ya se pueden posicionar las máquinas-herramienta, previamente jerarquizadas y texturizadas, dentro de él, en la posición que se desee.

6.4. Implementación de scripts de movimiento

En este apartado se recoge la implementación de los diferentes scripts desarrollados en lenguaje C# a través del entorno de desarrollo Visual Studio, los cuales han sido integrados en el museo de realidad virtual desarrollado en Unity 3D. El objetivo principal de estos scripts es dotar de comportamiento dinámico a los distintos componentes de las máquinas-herramienta, simulando los movimientos reales que pueden realizar en un entorno físico.

Todos los scripts han sido programados a medida para adaptarse a los requerimientos específicos del proyecto, y se han vinculado a los distintos elementos que forman parte de la jerarquía de cada máquina dentro del entorno de Unity. Estas vinculaciones permiten controlar de forma precisa los desplazamientos, rotaciones o acciones propias de cada parte móvil.

Debido a su extensión, se ha optado por incluir el contenido completo del código fuente de cada uno de los scripts en el Anexo A, mientras que en este apartado se presenta una descripción general de su funcionalidad, así como el elemento concreto al que está vinculado dentro del proyecto.

En primer lugar, se muestra un listado de los scripts que se han implementado para configurar los diferentes movimientos de las partes de las máquinas-herramienta:

- Script GiroEjeRectificadora (Anexo A1): Este script ha sido desarrollado para simular el giro continuo del eje principal de la rectificadora plana (flecha amarilla en la Figura 40), replicando el movimiento real de la rueda abrasiva. Su funcionalidad principal consiste en aplicar una rotación constante sobre el eje X del objeto al que está asignado, en función de una velocidad angular configurable desde el Inspector de Unity.

  El script se encuentra vinculado al objeto correspondiente a la rueda principal de la rectificadora plana dentro de la jerarquía del proyecto. En el método Start(), se almacena la rotación inicial del objeto, lo que permite, posteriormente, restablecer su posición mediante la función ResetearRotacion(). Esta funcionalidad resulta especialmente útil para devolver la pieza a su estado inicial tras la ejecución del movimiento.

  Además, el componente ha sido diseñado para ser activado mediante interacción del usuario en el entorno de realidad virtual. Por ello, en el Inspector de Unity, el script se mantiene desactivado por defecto, y se activa cuando el usuario pulse el botón correspondiente de la máquina ejecutando ActivarMovimiento().

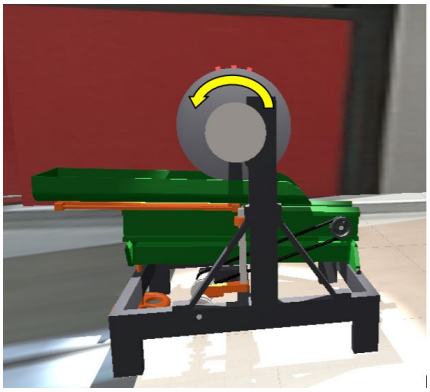


Figura 40: Movimiento Script GiroEjeRectificadora. Fuente: Propia

- Script Desplaza (Anexo A2): Este script ha sido vinculado al objeto Pieza dentro de la jerarquía de la rectificadora plana. Su funcionalidad consiste en simular el desplazamiento lineal del objeto (flecha amarilla en la Figura 41), replicando el avance real que se produce durante el proceso de rectificado.

  El comportamiento del movimiento se activa a través de una llamada externa del botón VR. Una vez completado el ciclo de movimiento, se puede resetear para volver a su estado original.

Figura 41: Movimiento Script Desplaza. Fuente: Propia

- Script GiroEje (Anexo A3): Este script ha sido desarrollado para simular el giro continuo del eje superior de la prensa de husillo (flecha amarilla en la Figura 42), representando el movimiento constante del componente durante el funcionamiento

real de la máquina. La lógica implementada permite aplicar de forma ininterrumpida una rotación sobre el eje X del objeto asignado, en función de una velocidad angular que se puede ajustar desde el inspector de Unity.

En este caso, el script se encuentra vinculado al objeto EjeRuedas dentro de la jerarquía correspondiente a la prensa de husillo. A diferencia de los demás scripts que se activan mediante interacción del usuario, este permanece activo desde el inicio de la simulación; por ello, no es necesario desmarcar el tic del script en el Inspector, ya que su ejecución forma parte del comportamiento base de la máquina desde que se carga la escena.

Figura 42: Movimiento Script GiroEje. Fuente: Propia

- Script TornilloMov (Anexo A4): Este script ha sido configurado para simular el movimiento de vaivén vertical de la pieza móvil de la prensa de husillo, replicando tanto el descenso como el ascenso que realiza el husillo en su funcionamiento real. Además, se le añade una rotación constante sobre su eje vertical durante el desplazamiento. Los movimientos se identifican en las flechas amarillas de la Figura 43.

  El script se encuentra vinculado al objeto screw de la prensa de husillo dentro de la jerarquía del proyecto y su activación está vinculada a la interacción del usuario con el botón VR correspondiente en la escena. Al ejecutarse, el script alterna entre

el descenso hasta una altura mínima establecida (stopY) y el regreso a la posición superior inicial. También se tiene una función de reinicio que permite devolver la pieza a su posición original, deteniendo el ciclo activo y ejecutando un retorno progresivo al punto de partida.

Figura 43: Movimiento Script TornilloMov. Fuente: Propia

- Script MoverPieza (Anexo A5): Ha sido implementado para simular un movimiento alternativo de vaivén en el eje horizontal (flecha amarilla en la Figura 44). El comportamiento se define mediante una velocidad y una distancia de desplazamiento configurables desde el Inspector de Unity. Al alcanzar el límite establecido, el objeto invierte su dirección de manera automática, generando un ciclo continuo.
  Este componente se encuentra vinculado a varios elementos clave de la limadora, como es el objeto EjeSuperior, dentro de la jerarquía del proyecto. Además, su activación está controlada por la interacción del usuario mediante un botón VR.

Figura 44: Movimiento Script MoverPieza. Fuente: Propia

- Script MoverPiezaB (Anexo A6): Este script complementa el funcionamiento del script anterior, simulando el avance progresivo de la mesa de trabajo sobre su eje Z. Su objetivo principal es representar el desplazamiento incremental que se produce tras cada ciclo de vaivén de la pieza superior. El movimiento de avance progresivo se identifica con las flechas amarillas de la Figura 45.

  El script se encuentra vinculado al objeto EjeBajo de la limadora dentro de la jerarquía. La lógica permite avanzar la posición del objeto en pequeños pasos mediante la función AvanzarUnPaso(), la cual es llamada automáticamente por el script MoverPieza tras completarse cada ciclo. El tamaño del desplazamiento es configurable mediante el parámetro pasoZ.

Figura 45: Movimiento Script MoverPiezaB. Fuente: Propia

- Script MovimientoY (Anexo A7): Este script ha sido implementado para simular el movimiento alternativo vertical sobre el eje Y local del objeto al que está asignado (flecha amarilla en la Figura 46). El desplazamiento sigue un ciclo de bajada y de subida, con parámetros de velocidad y distancia configurables desde el inspector de Unity. La activación se produce mediante la interacción del usuario en el entorno VR, a través de la función ActivarMovimiento(), mientras que el retorno a la posición inicial se gestiona con ResetearPosicion().

  Este script ha sido vinculado al objeto EjeHerramienta de la limadora dentro de la jerarquía de la taladradora en Unity, entre otros.

Figura 46: Movimiento Script MovimientoY. Fuente: Propia

- Script RotationZLimitedVR (nexo A8): Este script, asignado al objeto EjeHerramienta de la limadora, permite realizar una rotación limitada sobre el eje Z (flecha amarilla en la Figura 47).

  Al activarse mediante un botón VR, el objeto rota suavemente hasta -45º, y luego puede regresar a su posición inicial si se vuelve a pulsar el botón VR. Esta funcionalidad simula el comportamiento mecánico realista de control preciso de la herramienta desde la escena.

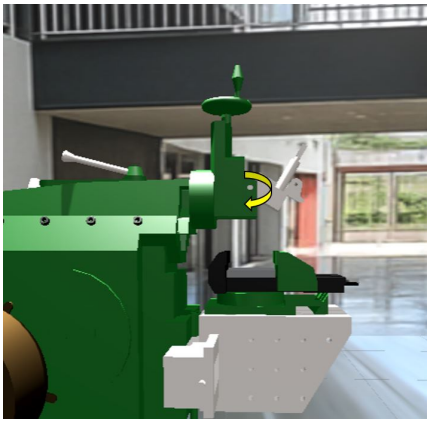


Figura 47: Movimiento RotationZLimitedVR. Fuente: Propia

- Script RotacionEjeY (Anexo A9): Este script ha sido desarrollado para simular una rotación continua sobre el eje Y local del objeto asignado, como se muestra en la flecha amarilla de la Figura 48, con una velocidad configurable desde el Inspector de Unity. Su funcionamiento se activa mediante la interacción del usuario a través de un botón VR, iniciando el giro mediante la función ActivarGiro(). Asimismo, puede ser detenido y devuelto a su orientación inicial mediante la función ResetearRotacion().

  El script ha sido vinculado al objeto EjeHerramienta de la fresadora dentro de la jerarquía.

Figura 48: Movimiento RotacionEjeY. Fuente: Propia

- Script AvanceY (Anexo A10): Este script ha sido implementado para simular un desplazamiento alternativo sobre el eje X local del objeto asignado (indicado en la flecha amarilla de la Figura 49). El comportamiento consiste en un movimiento de ida y vuelta con velocidad y distancia configurables desde el Inspector de Unity. Su activación se realiza desde el botón VR.

  Este script ha sido asignado al EjeAvanceY de la fresadora.

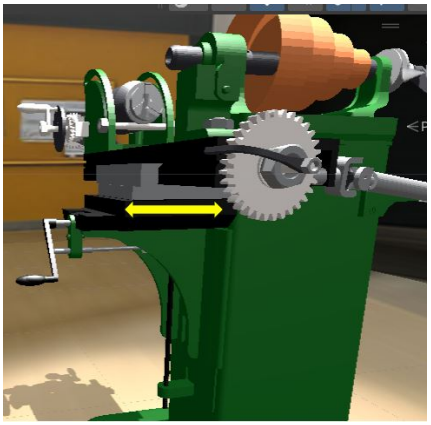


Figura 49: Movimiento AvanceY. Fuente: Propia

- Script MovimientoVaivenZ (Anexo A11): Este script simula un movimiento de vaivén continuo sobre el eje Z local del objeto al que esté asignado (indicado en la flecha amarilla en la Figura 50). Define dos posiciones límite a partir de la posición inicial, hacia un lado y hacia otro, entre las cuales el objeto se desplaza de forma suave. La velocidad y la distancia del recorrido son configurables desde el Inspector de Unity. Su activación y detención se controlan mediante un botón VR.

  Este script ha sido asignado al objeto EjeAvance de la fresadora en la jerarquía del proyecto de Unity, entre otros.

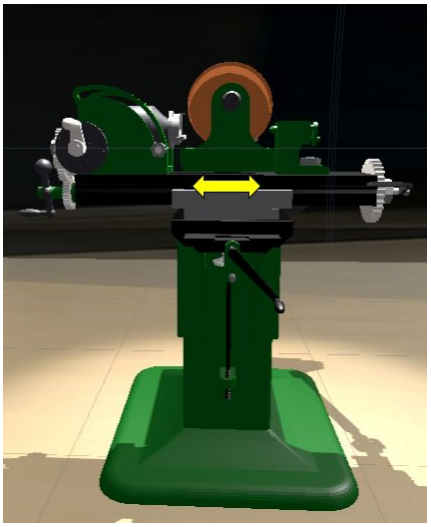


Figura 50: Movimiento MovimientoVaivenZ. Fuente: Propia

- Script RotacionControladaX (Anexo A12): Ha sido implementado para simular una rotación continua del objeto sobre su eje X local cuando el usuario active la acción en el botón de realidad virtual. Cuando se desactiva la acción en el entorno, el objeto vuelve de forma progresiva y suave a su orientación original. La velocidad de ambos movimientos puede configurarse en el Inspector de Unity mediante dos parámetros independientes.

  Este script ha sido vinculado al componente llamado EjeMuela de la rectificadora universal, para simular un movimiento de rotación puntual y controlado de la muela, como se muestra en la flecha amarilla en la Figura 51.

Figura 51: Movimiento RotacionControladaX. Fuente: Propia

- Script SutilVaiven (Anexo A13): En este script se ha implementado la simulación oscilante muy leve sobre el eje Z local del objeto al que está asignado, generando un efecto de vaivén suave, como se muestra en las flechas de la Figura 52. El ángulo de oscilación y la velocidad de movimiento pueden configurarse desde el Inspector, lo que permite ajustar el comportamiento del objeto. El funcionamiento se activa mediante el botón VR cuando el usuario interactúa con él.

  Ha sido incorporado al objeto MesaGiratoria de la rectificadora universal, con el propósito de representar el leve movimiento que esta parte podría experimentar durante el proceso de rectificado, mejorando así la sensación de realismo en la escena.

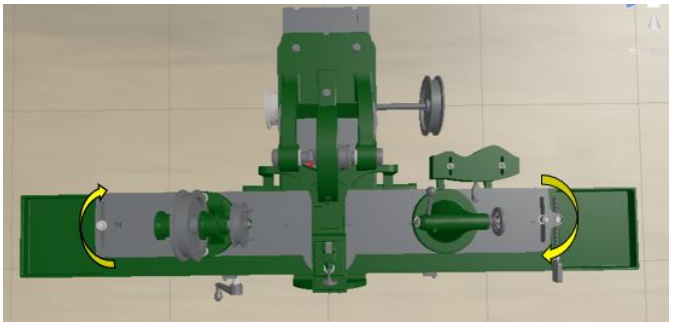


Figura 52: Movimiento SutilVaiven. Fuente: Propia

- Script RotacionControladaZ (Anexo A14): Permite una rotación controlada y continua sobre el eje Z local del objeto al que está asignado (indicado en la flecha amarilla de la Figura 53). La rotación se activa mediante interacción con el botón VR en la escena virtual y puede detenerse y volver a su posición inicial si se vuelve a interactuar con el botón VR.

  Este script ha sido vinculado al objeto EjeHerramienta de la taladradora, con el fin de simular el giro controlado de la broca durante el proceso de taladrado, replicado de manera virtual.

Figura 53: Movimiento RotacionControladaZ. Fuente: Propia

Después de haber explicado y referenciado cada uno de los scripts diseñados para simular los movimientos de las distintas partes móviles de las máquinas-herramienta, es

necesario describir un script fundamental que hace posible la activación y control desde el entorno de realidad virtual, denominado botón VR.

Este script (Anexo A) constituye uno de los elementos clave del sistema interactivo, ya que permite simular el comportamiento físico de un botón dentro del entorno de realidad virtual. Su lógica se basa en detectar colisiones mediante ”OnTriggerEnter” y ”OnTriggerExit”, de modo que cuando la mano del usuario (controlador VR) entra en contacto con el botón, este se hunde visualmente, modificando su posición local y reproduce un sonido de confirmación.

Además, el script utiliza eventos de Unity (UnityEvent) para invocar acciones configurables desde el Inspector, lo que permite asociar diferentes funciones, como la activación o desactivación de scripts de movimiento, sin necesidad de modificar el código fuente.

El script ha sido asignado a todos los botones físicos presentes en la escena del museo virtual, actuando como el principal mecanismo de interacción del usuario con las máquinas-herramienta. Gracias a su implementación, es posible controlar de manera directa y personalizada los distintos movimientos simulados en el entorno, dando una experiencia inmersiva.

6.5. Inserción soporte para realidad virtual

En este apartado se describirá el procedimiento necesario para habilitar el soporte de realidad virtual en el proyecto desarrollado con Unity 3D. El primer paso fundamental consiste en asegurarse de que el editor de Unity que se va a utilizar tenga instalados los módulos correspondientes a las plataformas Android y Windows, tal como se muestra en la Figura 54. Esta comprobación es esencial, ya que las gafas de realidad virtual utilizadas en este trabajo son las Meta Quest 2, que funcionan sobre el sistema operativo Android.

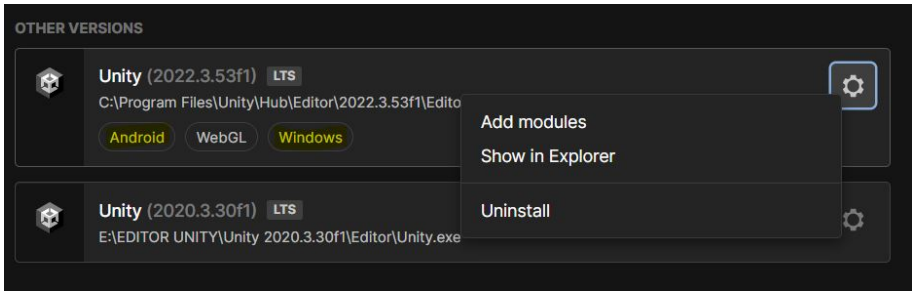


Figura 54: Módulos editor Unity. Fuente: Propia

En el caso de que no estuviesen instalados, se debe pulsar el engranaje de opciones del editor como aparece en la Figura 54 y clicar sobre ”Add modules” y añadir estos dos que se han mencionado.

Una vez verificado este requisito, es necesario instalar una serie de paquetes que facilitan la integración y el uso de la realidad virtual dentro del entorno de desarrollo de Unity. Este procedimiento se realiza desde el panel ”Window ->Package Manager”, ubicado en la barra superior de la pantalla principal de Unity, donde se encuentra la escena. En la ventana emergente del Package Manager, se debe seleccionar la opción ”Unity Registry”, como se observa en la Figura 55, para acceder al listado completo de paquetes oficiales disponibles. A continuación, se deben buscar e instalar los siguientes paquetes en el proyecto en el que se está trabajando:

- XR Interaction Toolkit: proporciona un conjunto completo de herramientas para la interacción en entornos de realidad virtual, como agarre, selección o teletransporte. Además, es recomendable importar los dos ejemplos incluidos en la pestaña ”Samples: Starter Assets y Hands Interaction Demo”, los cuales ofrecen prefabricados y scripts útiles para manejar interacciones básicas y con manos en entornos VR.
- XR Plugin Management: permite gestionar e instalar de forma centralizada los distintos proveedores de realidad virtual y aumentada compatibles con Unity.
- OpenXR Plugin: proporciona compatibilidad con el estándar OpenXR, un sistema unificado para desarrollar experiencias XR que funcionen en múltiples dispositivos, facilitando la portabilidad del proyecto entre distintas plataformas.

- AI Navigation: Agrega herramientas para navegación e inteligencia artificial, permitiendo definir superficies caminables y rutas para personajes y objetos dentro del entorno virtual.

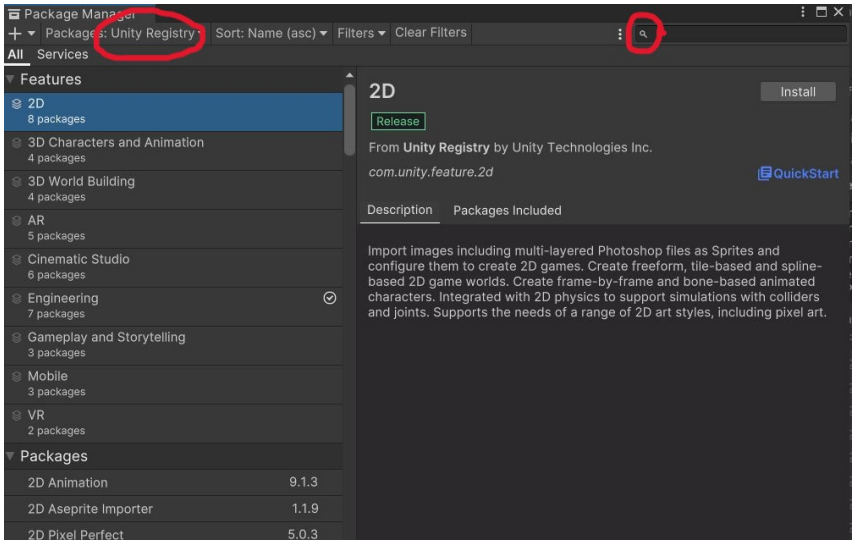


Figura 55: Paquetes Unity 3D. Fuente: Propia

Con los paquetes anteriores instalados, se debe acceder al menú ”Edit ->Project Settings” y dentro del panel izquierdo, seleccionar la opción ”XR Plug-in Management”.En este apartado se habilita el proveedor correspondiente a cada plataforma: para Windows, se debe seleccionar la opción OpenXR y, para Android, la opción Oculus, tal como se muestra en la Figura 56 y en la Figura 57. Esto permitirá al proyecto ejecutarse correctamente en las gafas Meta Quest 2, asegurando que Unity se comunique con el hardware de realidad virtual adecuado según la plataforma de ejecución.

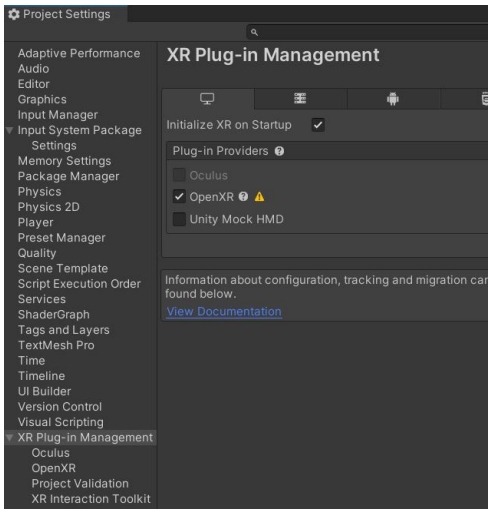


Figura 56: Plataforma windows. Fuente: Propia

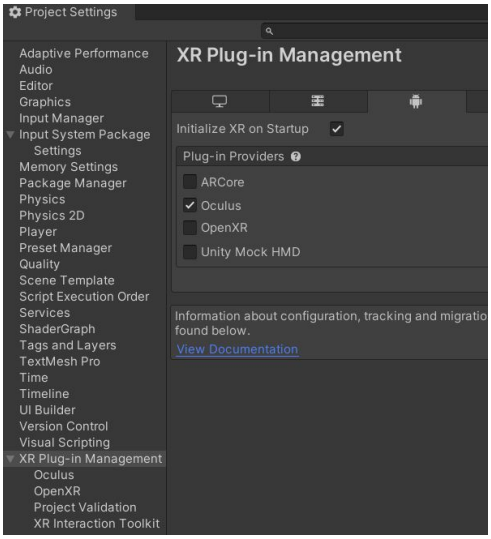


Figura 57: Plataforma android. Fuente: Propia

Una vez instalados todos los complementos necesarios y configurados los parámetros requeridos para habilitar la interacción en realidad virtual dentro de Unity, el siguiente paso consiste en insertar en la jerarquía de la escena los tres elementos fundamentales que permiten el correcto funcionamiento del entorno VR en Unity 3D. Estos elementos son: XR Interaction Manager, Event System y XR Origin (XR Rig).

En primer lugar, se crea un objeto vacío que se denominará XR Interaction Manager, al que se le añade el componente del mismo nombre. Este script se encuentra disponible en la ruta: Packages → XR Interaction Toolkit → Runtime → Interaction → Scripts → XR Interaction Manager. Basta con arrastrar este script sobre el objeto vacío previamente creado para que éste actúe como gestor de las interacciones XR dentro del entorno de realidad virtual.

A continuación, se crea un segundo objeto vacío llamado EventSystem. En el panel Inspector, se le añade el componente Event System, que se encuentra por defecto dentro del listado estándar de componentes de Unity. Adicionalmente, es imprescindible añadir el script XR UI Input Module, ubicado en la siguiente ruta: Packages → XR Interaction Toolkit → Runtime → UI → Scripts → XR UI Input Module. Este módulo permite la

gestión de entradas en interfaces de usuario dentro del entorno VR. En este caso, no es necesario realizar modificaciones adicionales en los parámetros del componente, ya que su configuración predeterminada es funcional para el proyecto.

Finalmente, se incorpora el objeto XR Origin (XR Rig), el cual representa al jugador dentro del entorno de realidad virtual y se encarga de simular su posición, desplazamientos y orientación en la escena. Para ello, se hace clic derecho en la jerarquía de Unity y se selecciona la opción XR → XR Origin, tal como se muestra en la Figura 58.

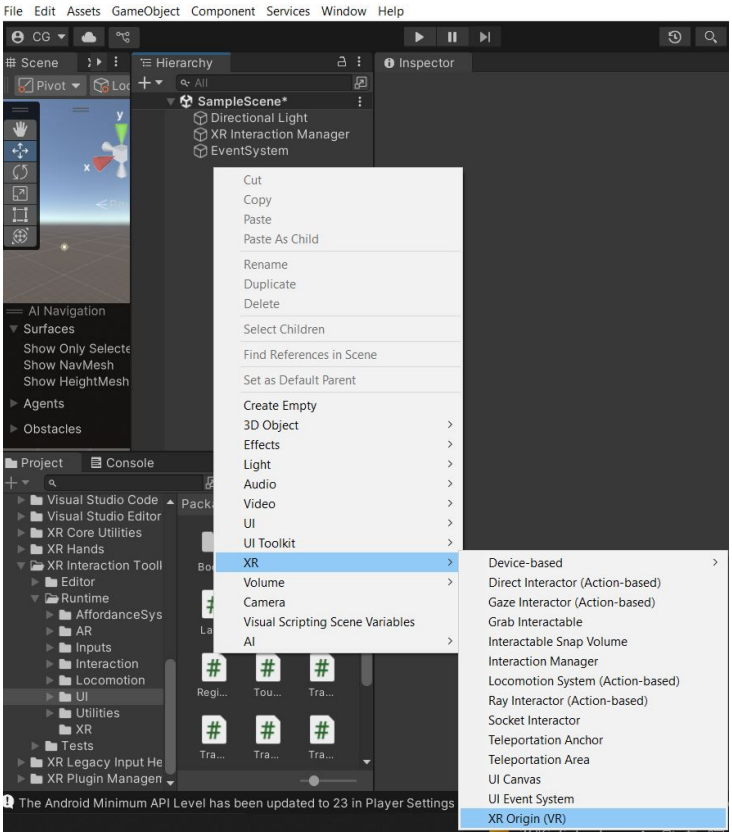


Figura 58: Objeto XR Origin. Fuente: Propia

Este objeto ya incluye de forma predeterminada dos scripts: XR Origin y Input Action Manager, los cuales se mantienen con su configuración por defecto. No obstante, para dotar a este objeto de capacidades de movimiento y navegación, es necesario añadirle varios componentes adicionales:

- Locomotion System; Ruta: Packages → XR Interaction Toolkit → Runtime → Locomotion → Scripts → Locomotion System. En este componente debe vincularse el objeto XR Origin (XR Rig) al campo denominado XR Origin, lo que permite gestionar los distintos sistemas de locomoción asociados al rig del jugador.

- Continuous Move Provider (Action Based); Ruta: Packages → XR Interaction Toolkit → Runtime → Locomotion → Continuous → Scripts → Action Based Continuous Move Provider. Este componente se encarga del movimiento continuo del jugador. En su campo System también se debe asignar el objeto XR Origin (XR Rig). Además, se puede configurar la velocidad de desplazamiento del jugador según las necesidades del proyecto.

- Character Controller: este componente se añade directamente desde el menú Add Component. Proporciona una forma básica de colisión para el jugador y es necesario para que otros scripts de movimiento funcionen correctamente

- Continuous Turn Provider (Action Based); Ruta: Packages → XR Interaction Toolkit → Runtime → Locomotion → Continuous → Scripts → Action Based Continuous Turn Provider. Permite realizar giros continuos con el jugador. Como en el caso anterior, se debe asignar el objeto XR Origin (XR Rig) al campo System. También es posible ajustar la velocidad de rotación del jugador desde este componente.

- Character Controller Driver; Ruta: Packages → XR Interaction Toolkit → Runtime → Locomotion → Scripts → Character Controller Driver. Este script se encarga de vincular el controlador de personaje con el sistema de locomoción. En el campo Locomotion Provider, debe añadirse nuevamente el objeto XR Origin (XR Rig).

Con estos componentes debidamente configurados, el objeto XR Origin (XR Rig) queda preparado para simular de forma precisa los movimientos y comportamientos del usuario dentro del entorno virtual. La Figura 59 muestra cómo queda finalmente configurado este objeto dentro del panel Inspector, con todos los scripts añadidos y listos para su uso.

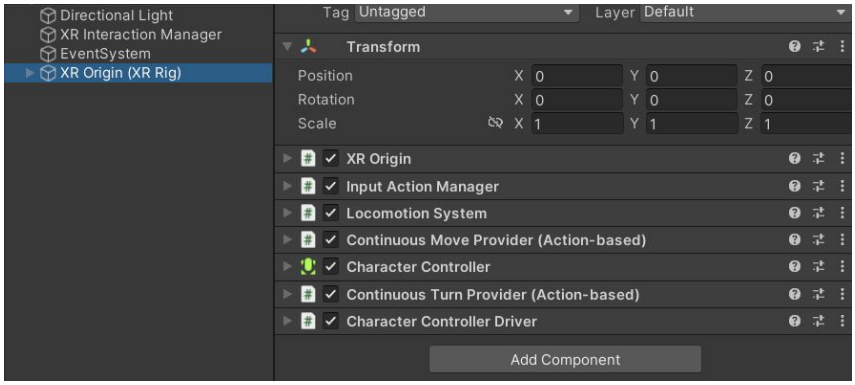


Figura 59: Componentes XR Origin. Fuente: Propia

A continuación, se describe el procedimiento para integrar la mano izquierda del jugador en el entorno de realidad virtual de la escena, así como todas las configuraciones necesarias para su correcto funcionamiento. Este mismo procedimiento deberá repetirse, de manera análoga, para la incorporación de la mano derecha.

Dentro del objeto previamente incorporado en la jerarquía con el nombre ”XR Origin”, se encuentra un objeto hijo denominado ”Left Hand” por defecto. Este elemento debe ser modificado para incluir los componentes que se muestran en la Figura 60.

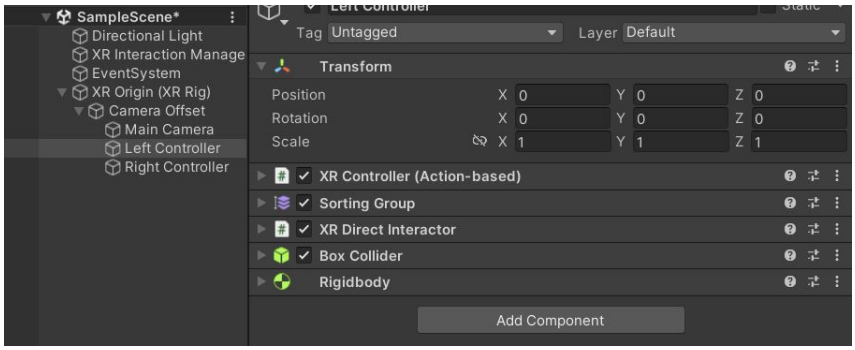


Figura 60: Componentes Left Hand. Fuente: Propia

Una vez añadidos los componentes, se procede a configurar cada uno de ellos. En primer lugar, en el componente ”XR Controller”, es necesario vincular cada una de las acciones a una referencia específica del controlador de realidad virtual. Para ello, se despliega el menú de opciones (tres puntos verticales) ubicado junto a cada acción,

se activa la opción ”Use Reference”, y se selecciona la referencia correspondiente al controlador izquierdo, de acuerdo con la acción asignada. El resultado final debe coincidir con la configuración mostrada en la Figura 61.

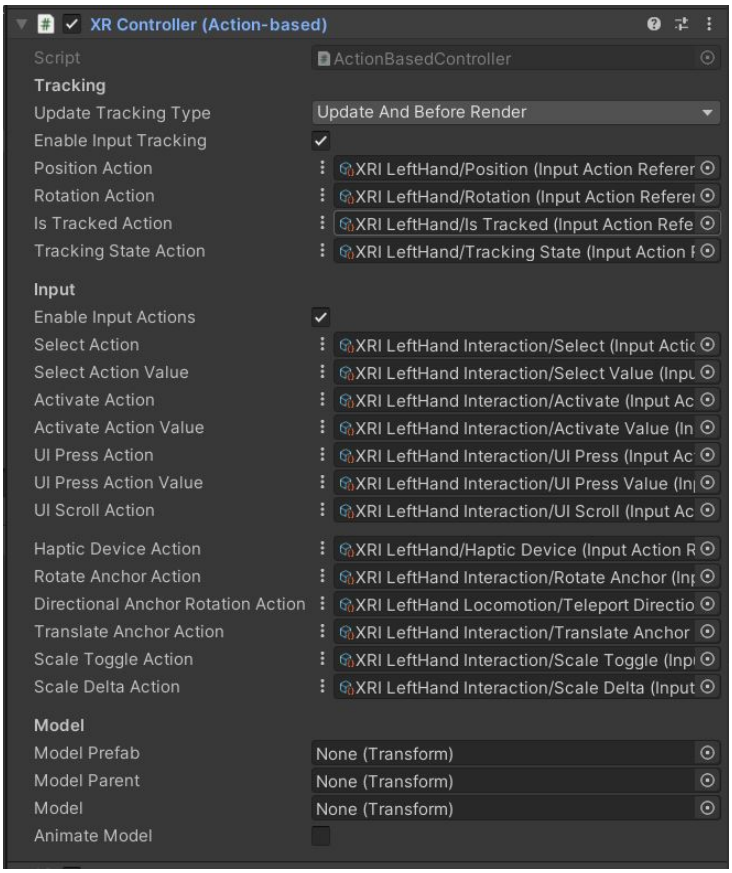


Figura 61: Acciones Left Hand. Fuente: Propia

El componente ”Sorting Group” no requiere modificaciones, por lo que se conservan sus valores por defecto. En el componente ”XR Direct Interactor”, debe asignarse en el campo ”Interaction Manager” el objeto ”XR Interaction Manager” previamente creado en la jerarquía, y activarse la casilla ”Allow Priority Activate”. En cuanto al componente ”Box Collider”, debe marcarse la opción ”Is Trigger”. Finalmente, en el componente ”Rigidbody”, se debe desmarcar la opción ”Use Gravity” y activar la casilla ”Is Kinematic” para que el comportamiento de la mano sea consistente con el entorno virtual.

A continuación, se incorpora el modelo 3D de la mano izquierda para proporcionar una representación visual realista. Este modelo se arrastra desde la carpeta ”Oculus Hand”, incluida previamente en el proyecto mediante los paquetes instalados en pasos anteriores, tal como se muestra en la Figura 62.

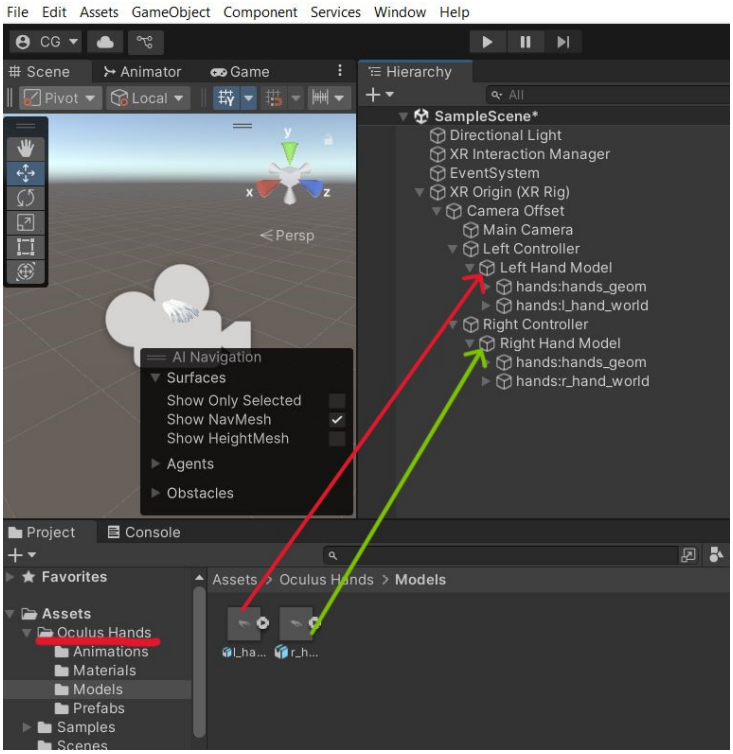


Figura 62: Modelos manos RV. Fuente: Propia

La última configuración necesaria consiste en definir el sistema de locomoción del jugador dentro del entorno virtual, es decir, especificar con qué controlador se realizará el desplazamiento y con cuál se ejecutarán los giros. En este proyecto, se ha optado por un sistema de control dual en el que el movimiento del jugador se gestiona mediante el joystick del controlador izquierdo, mientras que el giro se realiza utilizando el joystick del controlador derecho. Esta asignación permite una experiencia más intuitiva y fluida en la navegación del entorno virtual.

Para implementar esta funcionalidad, es necesario configurar correctamente los scripts previamente añadidos al objeto ”XR Origin (XR Rig)”. En concreto, dentro del

componente ”Action-Based Continuous Move Provider”, debe establecerse la acción correspondiente al movimiento del jugador en el campo denominado ”Left Hand Move Action”. En este campo se debe vincular la acción de tipo ”Move”, asegurándose de que la referencia provenga del controlador izquierdo. Esta configuración se ve en la Figura 63.

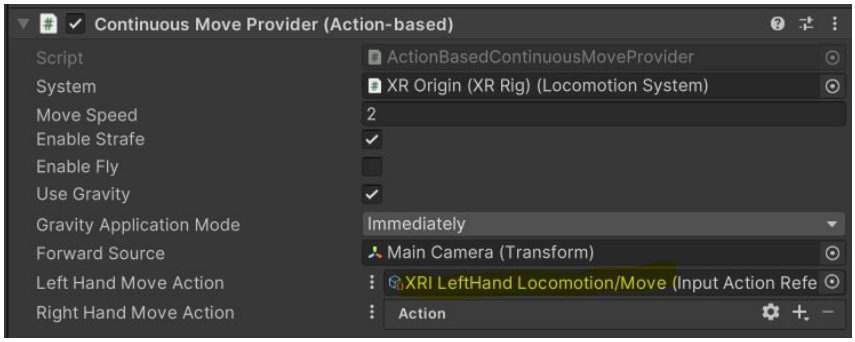


Figura 63: Configuración Move. Fuente: Propia

De forma análoga, para habilitar el giro del jugador, se debe configurar el componente ”Action-Based Continuous Turn Provider”. En este caso, en el campo ”Right Hand Turn Action”, se debe asignar la acción de tipo ”Turn”, haciendo referencia al controlador derecho. Esta configuración se muestra en la Figura 64.

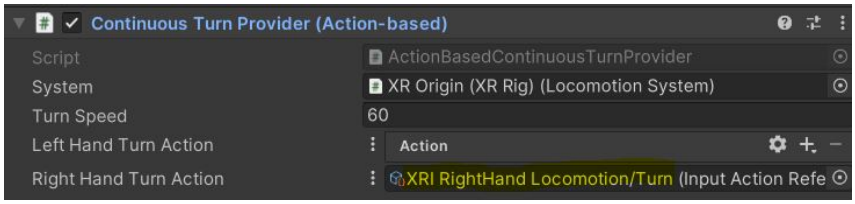


Figura 64: Configuración Turn. Fuente: Propia

Con esta configuración, se garantiza que el usuario pueda desplazarse y rotar dentro del entorno virtual de forma natural y precisa, mejorando significativamente la inmersión y la usabilidad del sistema de realidad virtual desarrollado.

6.6. Creación de botones

Para permitir la interacción del usuario en el entorno de realidad virtual, se han desarrollado botones personalizados que activan y desactivan tanto los movimientos de las distintas máquinas-herramienta como la visualización de los carteles informativos. Este sistema se ha implementado mediante la creación y configuración de objetos en Unity, con la ayuda de componentes físicos y scripts propios.

El primer paso ha consistido en la creación de un objeto vacío en la jerarquía, al que se le ha asignado el nombre "Mesa". Este objeto actúa como elemento padre de todos los botones relacionados con una máquina concreta, organizando de manera estructurada los distintos elementos necesarios para su funcionamiento (véase la Figura 65).

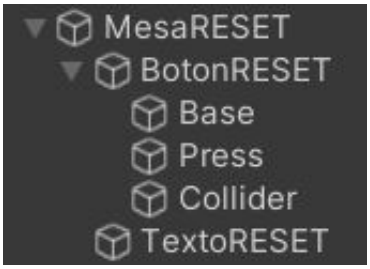


Figura 65: Jerarquía botonera. Fuente: Propia

Dentro del objeto mesa, se ha creado otro objeto vacío denominado "Botón", que contiene a su vez a tres elementos hijos, cada uno con una función específica:

- Base: Este objeto contiene únicamente la geometría visual de la base del botón, es decir, su apariencia externa. Se ha insertado un modelo 3D simple que representa la parte baja del botón con un cilindro, escalándolo a las medidas que se han considerado necesarias.
- Press: Este elemento es la parte móvil del botón, la que el usuario debe presionar para activar una acción. Está compuesto por varios componentes físicos que permiten su correcto comportamiento en la escena, como los que se muestran en la Figura 66, destacando el "Configurable Joint" que restringe el movimiento del objeto únicamente a lo largo del eje Y, simulando el comportamiento realista de un botón al ser

presionado. Gracias a este componente, el objeto ”Press” puede desplazarse hacia abajo y volver a su posición original automáticamente tras dejar de aplicar fuerza.

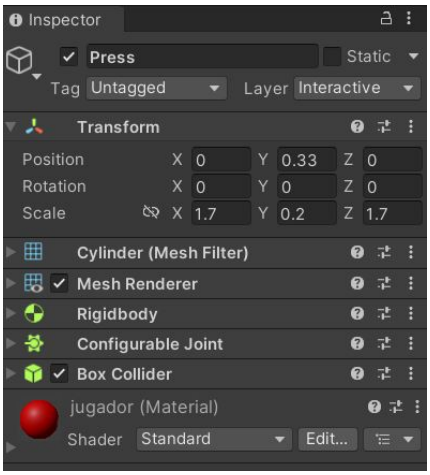


Figura 66: Configuración Press. Fuente: Propia

- Collider: Este tercer objeto es el encargado de detectar la colisión entre la parte móvil del botón (press) y una zona de activación. Su configuración, mostrada en la Figura 67 incluye un box collider, un rigidbody, un componente de audio para reproducir un sonido al activarse el botón y un script, llamado ”botonVR”, encargado de ejecutar una acción determinada cuando se produce la colisión entre el objeto ”Press” y el objeto ”Collider”, explicado con detalle en el apartado 6.4.

Figura 67: Configuración Collider. Fuente: Propia

Este diseño permite reutilizar la misma estructura de botón para múltiples funcionalidades dentro del entorno virtual, ajustando los objetos o movimientos que se activan dentro del script de botonVR, cambiando el objeto insertado dentro de la pestaña ”OnPress”.

Una vez completada la configuración de todos los elementos que componen cada botón, así como la asignación de sus respectivas funcionalidades mediante scripts, se ha obtenido un sistema interactivo plenamente funcional dentro del entorno de realidad virtual.

En la Figura 68 se muestra el resultado final de botones para una máquina-herramienta del museo virtual, posicionados junto a ella y listos para ser utilizados por el usuario. Este botón es capaz de detectar la interacción del usuario mediante el sistema de colisión descrito anteriormente, y activa correctamente la función correspondiente definida en el script.

Este diseño se ha replicado y adaptado para las distintas máquinas presentes en el museo, así como para los elementos informativos de todas ellas.

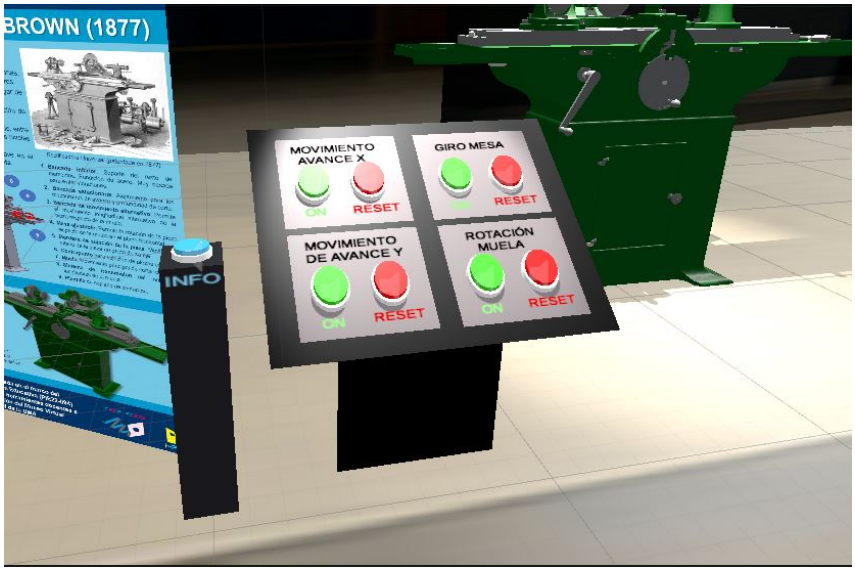


Figura 68: Botones final máquina-herramienta. Fuente: Propia

6.7. Luces

Con el objetivo de dotar de mayor realismo e inmersión al entorno virtual del museo, se han incorporado distintas fuentes de luz dentro del proyecto desarrollado en Unity. La correcta configuración e implementación de la iluminación resulta fundamental para lograr una experiencia visual coherente y funcional dentro del espacio expositivo simulado.

En primer lugar, se han colocado luces del tipo ”Spot” sobre cada una de las máquinas-herramienta expuestas en el museo. Estas luces han sido posicionadas directamente sobre cada máquina, simulando los focos de iluminación puntual que se utilizan habitualmente en entornos museísticos reales para resaltar elementos concretos de una exposición. Además, se ha añadido una luz ”Spot” adicional sobre cada botonera asociada a las máquinas, con el fin de facilitar su visualización por parte del usuario. Este tipo de luz permite crear luces dirigidas, generando sombras y contrastes que ayudan a centrar la atención del usuario sobre los objetos de interés. Un ejemplo de esta configuración de luces puede observarse en la Figura 69.

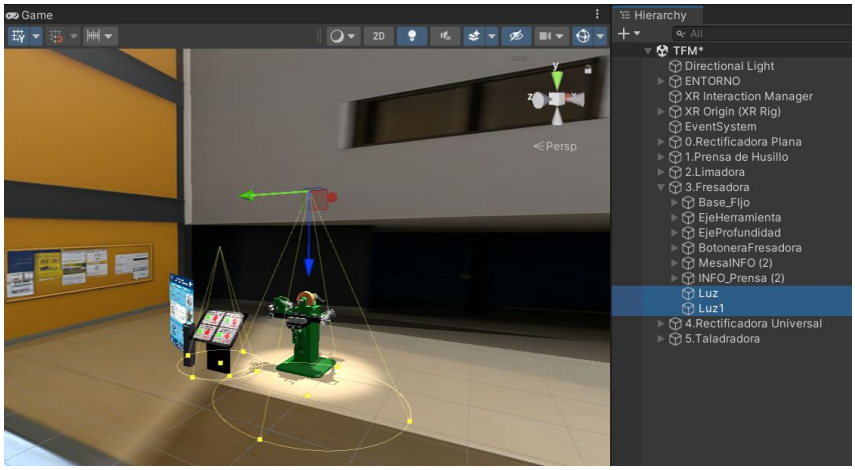


Figura 69: Luces Spot. Fuente: Propia

Las luces se incorporan en Unity de manera sencilla desde la jerarquía del proyecto: haciendo clic derecho sobre un espacio vacío y seleccionando ”Light” en el menú, se puede elegir entre los distintos tipos de fuentes de luz disponibles (como ”Spot”, ”Directional”...). Una vez creada, cada luz puede colocarse manualmente en la escena, ajustando su posición y rotación en función de las necesidades de iluminación de cada zona del museo.

Además de estas luces localizadas, se ha incorporado una luz de tipo ”Directional” situada en el exterior del edificio del museo. Esta luz tiene como finalidad simular la iluminación solar natural, y se ha orientado de forma que incida en dirección a la entrada principal del museo, contribuyendo a la ambientación general del entorno y reforzando el efecto de luz diurna que entra desde el exterior al museo. La Figura 70 muestra un ejemplo de esta configuración.

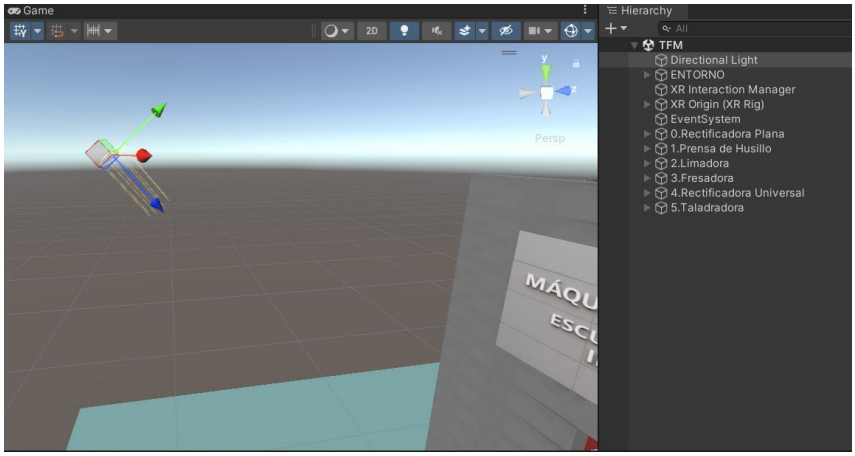


Figura 70: Luz Directional. Fuente: Propia

Ambas configuraciones han sido diseñadas para lograr una iluminación equilibrada que respete tanto los requisitos estéticos como funcionales del entorno virtual del museo.

6.8. Exportación del proyecto a gafas VR

Una vez finalizado el desarrollo del entorno virtual en Unity y verificado su correcto funcionamiento, el siguiente paso consiste en exportar el proyecto para su ejecución en un dispositivo de realidad virtual. En este caso, el dispositivo seleccionado es Meta Quest 2, por lo que es necesario generar un archivo ejecutable en formato .apk, compatible con el sistema operativo Android que utilizan estas gafas.

El proceso de exportación comienza accediendo al menú File >Build Settings dentro del entorno de desarrollo de Unity. A continuación, se debe seleccionar la plataforma de destino, en este caso Android, y pulsar el botón ”Switch Platform” si no está ya configurada como predeterminada. Este paso permite adaptar el proyecto a los requisitos y características específicas de compilación para Android.

Una vez seleccionada la plataforma correcta, en la parte inferior de la ventana ”Build Settings”, se deben añadir las escenas que se desean incluir en la compilación. Esto se realiza pulsando el botón ”Add Open Scenes”, asegurándose de que la escena principal del proyecto esté seleccionada y correctamente configurada.

A continuación, se debe hacer clic en el botón ”Build”. Unity solicitará al usuario seleccionar una carpeta de destino donde se guardará el archivo .apk resultante. En la Figura 71, se muestra la ventana, con la escena seleccionada y el botón ”Build” marcado, que debe pulsarse para comenzar el proceso de exportación.

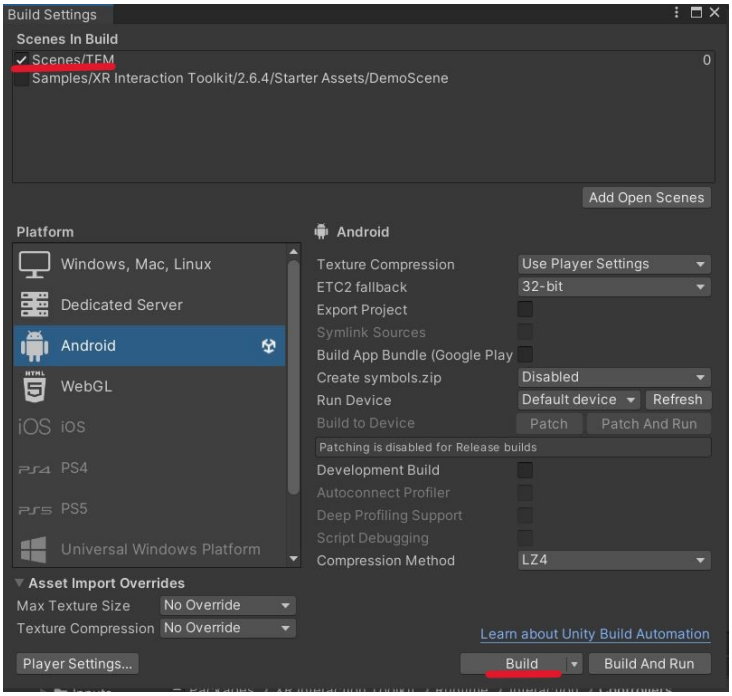


Figura 71: Ventana Build Settings. Fuente: Propia

Durante el proceso de compilación, Unity empaqueta todos los recursos, scripts, configuraciones y elementos de la escena en un único archivo ejecutable que podrá ser transferido e instalado en el dispositivo Meta Quest 2. Su instalación en las gafas se detalla en el apartado siguiente de este documento.

6.9. Instalación de la aplicación en las gafas VR

Cuando ya se tiene el archivo ejecutable .apk, el siguiente paso es instalarlo en el dispositivo de realidad virtual, las Meta Quest 2, para poder ejecutar la aplicación de manera autónoma dentro del dispositivo.

Es importante destacar que, para poder instalar aplicaciones desarrolladas externamente, como es el caso de un proyecto creado en Unity, las gafas deben tener activado el modo desarrollador. Este es un requisito indispensable, ya que sin esta opción habilitada no es posible ejecutar aplicaciones que no provengan directamente de la tienda oficial de Meta. En el caso del presente proyecto, las gafas ya disponían de este modo activado, por lo que no ha sido necesario realizar ningún ajuste adicional. No obstante, se menciona este aspecto por su relevancia en futuros desarrollos similares.

Para llevar a cabo la instalación del archivo. apk, se ha utilizado la aplicación Meta Quest Developer Hub, una herramienta proporcionada por Meta que facilita la gestión de dispositivos, la instalación de aplicaciones y el acceso a herramientas extra.

El procedimiento seguido es el siguiente:

- Se conecta el dispositivo Meta Quest 2 al ordenador mediante un cable USB.
- Una vez abierto Meta Quest Developer Hub, se accede a la pestaña ”Device Manager”, situada en la parte superior izquierda de la interfaz.
- En esta sección, se muestra información del dispositivo conectado. Para instalar la aplicación, se arrastra el archivo .apk hasta el área denominada ”Installed Apps”, donde se gestiona el conjunto de aplicaciones instaladas en las gafas.
- El sistema procede automáticamente a transferir e instalar la aplicación en el dispositivo.

En la parte superior de la Figura 72 se puede observar que el dispositivo de Meta Quest 2 está conectado al ordenador y listo para realizar cualquier tarea sobre él, como insertar el archivo descargado de Unity, que se observa en la parte media de la interfaz como ”com.Carmen.Prueba360”.

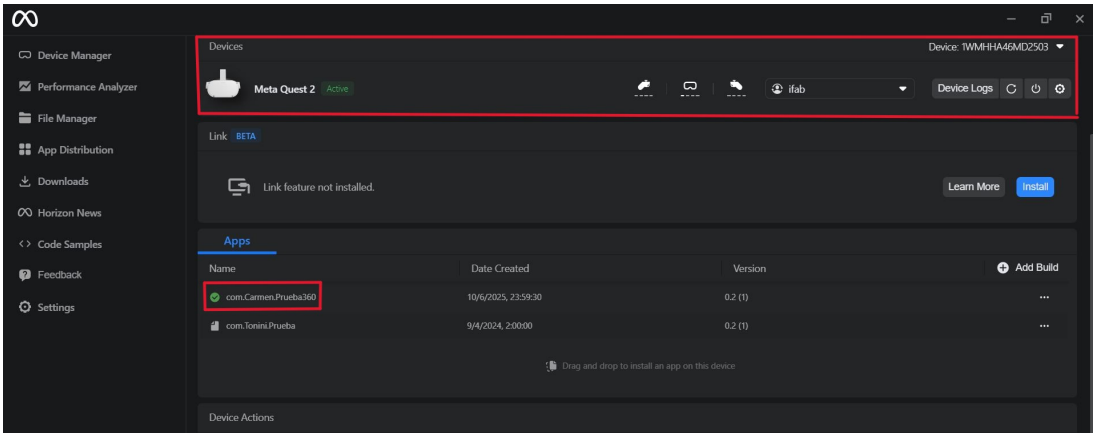


Figura 72: Gafas disponibles en MQDH. Fuente: Propia

6.10. Ejecución de la aplicación en las gafas VR

Una vez que el archivo ha sido correctamente instalado en las gafas Meta Quest 2, el siguiente paso consiste en acceder a la aplicación desde el propio dispositivo para iniciar la experiencia de realidad virtual.

En el caso de Meta Quest, todas las aplicaciones que han sido instaladas manualmente desde fuentes externas, como es el caso de la de este trabajo desarrollada en Unity, se almacenan dentro de una carpeta específica denominada ”Orígenes desconocidos”. Esta sección agrupa todas las apps que no provienen de la tienda oficial de Meta, pero que han sido cargadas del modo desarrollador.

Para ejecutar la aplicación del museo de realidad virtual, se deben seguir los siguientes pasos:

- Encender las gafas Meta Quest 2 y colocarlas en la cabeza correctamente.
- Desde el menú principal, acceder al panel de biblioteca pulsando el icono correspondiente, como se muestra en la Figura 73.

Figura 73: Icono biblioteca en gafas VR. Fuente: Propia

- En el menú desplegable de la parte izquierda, seleccionar la carpeta ”Orígenes desconocidos”.
- Pulsar sobre el nombre de la aplicación para iniciarla, como se ve en la Figura 74. Una vez cargada, el usuario se encontrará dentro del museo virtual, pudiendo desplazarse libremente e interactuar con las diferentes máquinas-herramienta.

Figura 74: Pestaña orígenes desconocidos y aplicación museo. Fuente: Propia

Este proceso es necesario cada vez que se quiera ejecutar la aplicación desde las gafas, ya que no se integra directamente en el listado principal de aplicaciones instaladas.

7. Resultados

Como resultado final del presente Trabajo Fin de Máster, se ha logrado desarrollar un museo de realidad virtual de la Escuela de Ingenierías Industriales, que permite la visualización e interacción con distintas máquinas-herramienta pertenecientes al patrimonio industrial. El proyecto se ha llevado a cabo utilizando el motor de desarrollo Unity 3D y se ha adaptado para su ejecución en las gafas de realidad virtual Meta Quest 2, generando una experiencia inmersiva e intuitiva.

A lo largo del proceso, se han alcanzado los siguientes puntos:

- Diseño y modelado de un entorno virtual que simula el vestíbulo de la Escuela de Ingenierías Industriales para permitir la exposición de cada máquina y el recorrido del usuario.
- Incorporación de los modelos tridimensionales de las distintas máquinas-herramienta con su correspondiente organización en la jerarquía de Unity para permitir la gestión de sus componentes.
- Desarrollo de scripts en lenguaje C# que permiten simular los movimientos de cada componente móvil (giros, desplazamientos lineales, oscilaciones...) de las máquinas, adaptando el comportamiento real al entorno virtual. Cada uno de estos scripts ha sido vinculado a un componente específico de la máquina dentro de la jerarquía del proyecto.
- Para la activación de los movimientos, se han implementado sistemas de interacción mediante botones VR. Cada botón permite al usuario iniciar o detener una acción concreta en la máquina acercando la mano al botón en el entorno de realidad virtual.
- Inclusión de elementos visuales informativos asociados a cada máquina, adoptando un valor didáctico al recorrido del museo.
- Configuración del entorno virtual para su correcto funcionamiento en dispositivos Meta Quest, incluyendo locomoción libre y orientación del jugador.

- Generación del archivo ejecutable en formato .apk y posterior instalación en las gafas, permitiendo iniciar y explorar el museo directamente desde las gafas de realidad virtual.

El resultado final es un entorno virtual interactivo en el que el usuario puede desplazarse libremente, activar o desactivar el movimiento de las máquinas y consultar información asociada, todo ello en un espacio simulado que reproduce el entorno del vestíbulo de la Escuela de Ingenierías Industriales de la Universidad de Málaga.

A continuación, se muestran distintas capturas que evidencian los resultados obtenidos, incluyendo vistas generales del entorno, detalle de las máquinas, sus interacciones mediante botones y los paneles informativos integrados:

Figura 75: Entrada al museo virtual. Fuente: Propia

Figura 76: Vista general del museo. Fuente: Propia

Figura 77: Rectificadora plana en el museo. Fuente: Propia

Figura 78: Pulsando botón en prensa de husillo del museo. Fuente: Propia

Figura 79: Pulsando botón en limadora del museo. Fuente: Propia

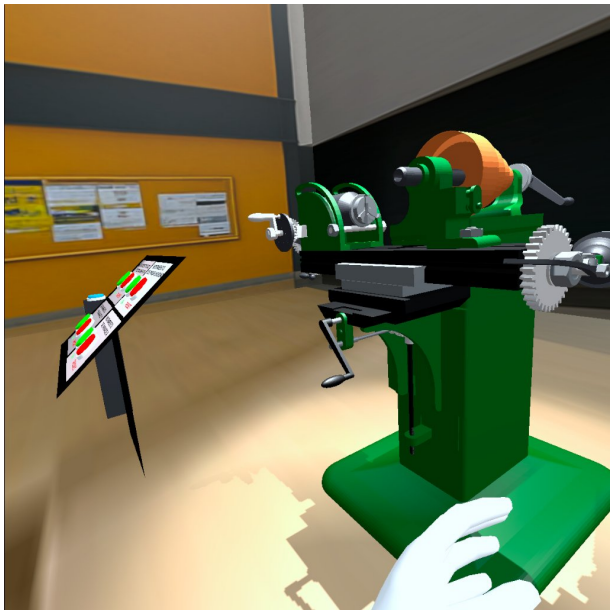


Figura 80: Fresadora en el museo. Fuente: Propia

Figura 81: Rectificadora universal en el museo. Fuente: Propia

Figura 82: Taladradora en el museo. Fuente: Propia

8. Conclusiones y líneas futuras de trabajo

8.1. Conclusiones

A lo largo de este Trabajo Fin de Máster se ha logrado diseñar y desarrollar un museo de realidad virtual que simula el funcionamiento de diversas máquinas-herramienta utilizadas en el ámbito de la ingeniería industrial. Este museo tiene como finalidad principal la preservación, divulgación y enseñanza del patrimonio industrial.

El trabajo ha cumplido de forma satisfactoria todos los objetivos establecidos inicialmente. Se ha conseguido la integración funcional de seis máquinas-herramienta, cada una de ellas dotada de comportamiento mecánico simulado y paneles informativos complementarios. Estas características han sido implementadas dentro de un entorno construido en el motor gráfico Unity 3D, lo que ha permitido no solo una representación visual, sino también una experiencia fluida e intuitiva para el usuario.

La programación de scripts en lenguaje C# ha posibilitado que cada máquina responda a acciones interactivas mediante botones VR y que esté dotada de un movimiento similar al de la realidad. A nivel de interacción, la implementación de botones VR ha facilitado una experiencia intuitiva para el usuario.

Cabe destacar también la exportación del entorno virtual a las gafas Meta Quest 2, lo que ha proporcionado al proyecto una alta portabilidad y autonomía de uso, sin necesidad de conexión permanente a equipos. Esto favorece su aplicación en cualquier entorno, como en actividades docentes o formativas.

En definitiva, el trabajo realizado constituye una base sólida para el desarrollo de nuevas herramientas educativas y de difusión de patrimonio industrial basadas en entornos virtuales inmersivos. La combinación de tecnologías de la Industria 4.0. con el ámbito del patrimonio industrial abre nuevas vías para reinterpretar y conservar elementos históricos, adaptándolos a los avances tecnológicos actuales.

8.2. Líneas futuras de trabajo

Tras la finalización del presente Trabajo Fin de Máster, se identifican diversas líneas de trabajo que pueden desarrollarse para mejorar y ampliar el museo virtual creado:

- Incorporación de nuevas máquinas-herramienta: Se plantea añadir progresivamente otros modelos del patrimonio industrial.
- Mejora del entorno: Algunos elementos del entorno podrían ajustarse, como por ejemplo, texturas, luces o colisiones, para mejorar la fluidez de la aplicación y el aspecto visual.
- Nuevo sistema de navegación: Se podría estudiar la posibilidad de incorporar teletransporte por puntos en el entorno, en lugar de desplazamiento libre a través de los joysticks.
- Interacción directa con las máquinas: Explorar nuevas formas de interacción en las que el usuario pueda manipular partes móviles de cada máquina con sus manos virtuales, como giros de manivelas.
- Introducción de información adicional con paneles emergentes: Ampliar los contenidos informativos que aparecen al interactuar con cada máquina, incluyendo, por ejemplo, datos técnicos, imágenes antiguas, enlaces a fuentes externas o vídeos que muestren la máquina real.
- Implementación de una visita guiada automática: Incorporar una opción de recorrido asistido que explique paso a paso las distintas máquinas y su relevancia, ideal para usuarios no familiarizados con la navegación en VR.
- Pruebas con usuarios: Se podrían realizar sesiones con estudiantes para recoger opiniones sobre la experiencia y detectar mejoras de accesibilidad o comprensión.

Referencias

Alexandre, B., Salguero, J., Peralta-Álvarez, M.-E., Aguayo-González, F., & Ares, E. (2017). Aplicación de las tecnologías de la Industria 4.0 al diseño y fabricación de productos artesanales. *DYNA*, *92*(4), 435-441. https://doi.org/10.6036/8169

Arias, D., López, A., & González, M. (2022). Gemelo digital en edificios patrimoniales y la evolución de este concepto como herramienta de gestión del plan de conservación programada. *Revista Tecnología en Marcha*. https://dialnet.unirioja.es/descarga/articulo/9272709.pdf

Autodesk App Store. (s.f.). Eyecad VR - Virtual Reality Architecture | Revit [Consultado el 15 de mayo de 2025].

Ávila, A. J. R. (2023). *Metodología para el desarrollo de un museo virtual de máquinas-herramienta de relevancia histórica.* [Trabajo Fin de Máster]. Universidad de Málaga [Máster en Ingeniería Industrial].

Bolaños, J. P. B. (2019). El potencial de la realidad aumentada en la gestión de restauración de edificios patrimoniales. *Tecnología en Marcha*, *32*(Especial), 27-36. https://doi.org/10.18845/tm.v32i5.4169

Botella Ortega, E. J. (2021). Aplicación de técnicas de fotogrametría para el modelado de una limadora como elemento de patrimonio industrial. *Trabajo Fin de Grado, Universidad de Málaga*, 1-115.

Cardona, Á. C., & Herrera, P. A. B. (2014). *Manual básico de Unity 3D como apoyo al desarrollo turístico nacional* [Tesis de grado]. Universidad Tecnológica de Pereira, Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación [Ingeniería de Sistemas y Computación].

Chaves Palacios, J. (2004). Desarrollo tecnológico en la Primera Revolución Industrial. *Norba. Revista de Historia*, *17*, 93-109.

Culturalab. (2022). Digitalización del Patrimonio. https : / / culturalab . es / digitalizacionpatrimonio/

Digital Atom SRL. (s.f.). Eyecad VR: Software de visualización arquitectónica [Consultado el 15 de mayo de 2025].

El Lahiani Skatou, O. (2022). Modelización 3D de la rectificadora universal de Brown and Sharpe de 1977. *Trabajo Fin de Grado, Universidad de Málaga*, 1-168.

English School. (2024). Primera fresadora universal de Brown y Sharpe [Accedido el 22 de abril de 2025]. https://www.meisterdrucke.es/impresion-art%C3%ADstica/English-School-English-School/589713/Primera-fresadora-universal-de-Brown-y-Sharpe.html

Epic Games. (2024). Unreal Engine [Accedido el 14 de mayo de 2025].

Escartín, E. R. (2023). La realidad virtual, una tecnología educativa a nuestro alcance [Documento subido el 6 de febrero de 2023].

EVE Museografía. (2023, diciembre). *Evolución de los museos en la era digital* [EVE Museografía]. https://evemuseografia.com/2023/12/06/evolucion-de-los-museos-en-la-era-digital/

Fernández, A. A. (2017). La Educación y el patrimonio industrial en España. La protección del patrimonio industrial a través de la competencia en conciencia y expresiones culturales [orcid: 0000-0003-4415-7125]. *II Congreso Internacional Virtual sobre la Educación en el siglo XXI (15-29 de marzo de 2017)*.

for Developers, M. (2024). *Meta Quest Developer Hub* [Accedido el 21 de mayo de 2025]. https://developer.oculus.com/documentation/unity/meta-quest-developer-hub-overview/

González, L. M. G. (s.f.). El patrimonio industrial y los museos: evolución histórica y propuestas museísticas en Europa. *Universidad de Murcia*, 385-402.

González-Hernández, I. J., Armas-Álvarez, B., Coronel-Lazcano, M., Vergara-Martínez, O., Maldonado-López, N., & Granillo-Macías, R. (2021). El desarrollo tecnológico en las revoluciones industriales. *Ingenio y Conciencia: Boletín Científico de la Escuela Superior Ciudad Sahagún*, *8*(16), 41-52. https://repository.uaeh.edu.mx/revistas/index.php/sahagun/article/view/7922

Grande Álvarez, N. (2017). La musealización del Patrimonio Industrial. Dos modelos de intervención en la provincia de Huelva: Casa Dirección en Valverde del Camino y Molino de Mareas .El Pintado.en Ayamonte. *PASOS. Revista de Turismo y Patrimonio Cultural*, *15*(3), 659-672. http://www.redalyc.org/articulo.oa?id=88151417010

Kawamura, K., & Tanaka, H. (2021). Comparison of Eevee and Cycles rendering engines for scientific visualization. *Journal of Visualization and Computer Animation*, *32*(5).

Library, S. S. P. (2009). Vertical Drilling Machine by J. Whitworth & Co in 1847 [Accedido el 27 de abril de 2025]. https://www.gettyimages.es/detail/fotograf%C3%ADa-de-noticias/engraving-by-g-gladwin-after-a-drawing-produced-fotograf%C3%ADa-de-noticias/90742027

López, G. A., Sáiz-Manzanares, M. C., Sánchez, R. M., Llamazares, M. C. E., & Arribas, S. R. (2023). Aplicación de técnicas de realidad virtual en el aprendizaje del patrimonio industrial [Correo: gandres@ubu.es]. *Universidad de Burgos*.

López, G. A., & Soria Cáceres, C. H. (2023). El estudio del patrimonio industrial en España: cincuenta años de análisis sobre el legado de la industrialización contemporánea (1972–2022). *Cuadernos Geográficos*, *62*(1), 208-232. https://doi.org/10.30827/cuadgeo.v62i1.26594

Martín Béjar, S., Trujillo Vilches, F. J., Bermudo Gamboa, C., Herrera Fernández, M., & Sevilla Hurtado, L. (2023). Photogrammetry Techniques Application for the Digitization of Machine Tools as Industrial Heritage Movable Assets [Paper 03-035]. *Proceedings of the 27th International Congress on Project Management and Engineering*.

Martín Vázquez, M. (2022). Modelización de una tectificadora del siglo XIX. *Trabajo Fin de Grado, Universidad de Málaga*, 1-127.

Martín-Béjar, S., Claver, J., Sebastián, M. A., & Sevilla, L. (2020). Graphic Applications of Unmanned Aerial Vehicles (UAVs) in the Study of Industrial Heritage Assets. *Applied Sciences*, *10*(24), 8821. https://doi.org/10.3390/app10248821

Microsoft. (2023). *Uso de Visual Studio Tools para Unity* [Accedido el 23 de mayo de 2025]. https://learn.microsoft.com/es-es/visualstudio/gamedev/unity/get-started/using-visual-studio-tools-for-unity

Miñarro, P. M. (2016). *Desarrollo de una aplicación de realidad virtual* [Trabajo Fin de Grado]. Universidad Politécnica de Valencia [Grado en Ingeniería Informática].

Miranda, V. (s.f.). *Preservación del patrimonio cultural* [Citaliarestauro. Recuperado de la web]. https://es.citaliarestauro.com/producto/preservacion-del-patrimonio-cultural/

Navarro, F., Martínez, A., & Martínez, J. M. (2018). *Realidad virtual y realidad aumentada: desarrollo de aplicaciones*. RA-MA Editorial.

Orive, D., López, A., Estévez, E., Orive, A., & Marcos, M. (2021). Desarrollo de gemelos digitales para la simulación e integración de activos de fabricación en la industria 4.0. *XLII Jornadas de Automática: libro de actas. Castelló, 1-3 de septiembre de 2021*, 709-716. https://doi.org/10.17979/spudc.9788497498043.709

Otero, A., & Flores, J. (2011). Realidad Virtual. Un medio de comunicación de contenidos. *Revista ICONO14. Revista de comunicación y tecnologías emergentes*, *9*(2), 185-211.

Pérez, M. A. S., Gil, J. C., Prieto, M. V., Hurtado, L. S., & Carretero, A. G. (2021). Estudio técnico y contextual de las prensas de husillo de las Reales Fábricas de San Juan de Alcaraz de Ríopar (Albacete).

Porras García, M. (2023). Modelado de la fresadora universal de Frederick W.Howe como elemento del patrimonio industrial. *Trabajo Fin de Grado, Universidad de Málaga*, 1-101.

Prada, C., Galán-Casado, S., Pitarch, J. L., Sarabia, D., Galán, A., & Gutiérrez, G. (2021). Flujos de trabajo digitales en la documentación del patrimonio industrial inmueble. La cadena de datos del proyecto PITGUADIANA. *Sanespat*. https://santanderestudiospatrimonio.unican.es/index.php/sanespat/article/view/188

Prada, C., Galán-Casado, S., Pitarch, J. L., Sarabia, D., Galán, A., & Gutiérrez, G. (2022). Gemelos Digitales en la Industria de Procesos. *Revista Iberoamericana de Automática e Informática Industrial*, *19*(3), 285-296. https://doi.org/10.4995/riai.2022.1690

Rojas López, A. (2024). Modelización de la taladradora de columna de Whitworth de 1847. *Trabajo Fin de Grado, Universidad de Málaga*, 1-157.

Rojas-Sola, J. I., García, M. C., Cañadas, M. D. P. C., & Anguita, F. J. C. (2013). Herramientas CAD/CAE en la caracterización tecnológica del Patrimonio Histórico

Industrial: Aplicación a una prensa de aceite de oliva. *Virtual Archaeology Review*, *4*(8), 69-73.

Sanchez Linares, C. (2021). Modelado de una prensa de husillo a fricción ubicada en el museo de las Reales Fábricas de Riópar (Albacete). *Trabajo Fin de Grado, Universidad de Málaga*, 1-55.

Technologies, U. (2021). *Creando y usando scripts* [Accedido el 23 de mayo de 2025]. https://docs.unity3d.com/es/2021.1/Manual/CreatingAndUsingScripts.html

Tolsan, V. (2022, enero). *La realidad aumentada en procesos de mantenimiento y reparación* [Consultado el 12 de mayo de 2025]. https://2ixr.com/blog/la-realidad-aumentada-en-procesos-de-mantenimiento-y-reparacion/

Trujillo, F. J., Martín-Bejar, S., Bermudo, C., Herrera, M., & Sevilla, L. (2022). Modelado 3D del ingenio azucarero de San Joaquín de Maro mediante técnicas Fotogramétricas.

Trujillo Vilches, F. J., Martín Béjar, S., Herrera Fernández, M. J., Bermudo Gamboa, C., & Sevilla Hurtado, L. (2024). Development of the EII Virtual Museum (UMA) as a tool for dissemination and enhancement of Industrial Heritage [Universidad de Málaga]. *28th International Congress on Project Management and Engineering*.

Vázquez, T., Rodríguez, A., & Martín, J. (2020). Design of interactive 3D applications using Blender and Unity. *Procedia Computer Science*, *176*.

Woodbury, R. S. (1958). *History of the Grinding Machine*. The Technology Press.

Zubacor. (2024). Máquina limadora horizontal [Accedido el 22 de abril de 2025]. https://zubacor.com/zubacor/productos-2/maquina-limadora-horizontal/

Anexos

A continuación, se presentan los anexos correspondientes a este trabajo, los cuales contienen información complementaria que respalda los resultados y análisis realizados.

Índice de Anexos

A. Scripts

En este anexo se recogen todos los scripts desarrollados en lenguaje C# que han sido implementados en el proyecto de realidad virtual descrito a lo largo de este trabajo. Cada uno de ellos ha sido detallado previamente en el apartado 6.4, donde se explica su funcionalidad y la relación con los elementos de la escena en Unity. A continuación, se presenta el contenido completo del código fuente correspondiente a cada script.

A1. GiroEjeRectificadora.cs

```
using System.Collections;
using UnityEngine;

public class GiroEjeRectificadora : MonoBehaviour
{
    public float rotationSpeed = 100f; // Velocidad de rotaci n en grados por segundo

    private Quaternion rotacionInicial;
    private bool girando = false;
    private bool enReset = false;

    void Start()
    {
        rotacionInicial = transform.rotation; // Guarda la rotaci n inicial
    }

    void Update()
    {
        if (girando && !enReset)
        {
            float rotationThisFrame = rotationSpeed * Time.deltaTime;
            transform.Rotate(Vector3.right * rotationThisFrame);
        }
    }

    public void ActivarMovimiento()
    {
        if (!enReset)
        {
            Debug.Log("Giro continuo activado desde el bot n VR.");
            this.enabled = true;
            girando = true;
        }
    }

    public void ResetearRotacion()
    {
        Debug.Log("Reseteando rotaci n...");
        girando = false;
        StartCoroutine(VolverARotacionInicial());
    }

    private IEnumerator VolverARotacionInicial()
    {
```

```
45          enReset = true;
46          float t = 0;
47          Quaternion rotacionActual = transform.rotation;
48
49          while (t < 1)
50          {
51              t += Time.deltaTime;
52              transform.rotation = Quaternion.Lerp(rotacionActual, rotacionInicial, t);
53              yield return null;
54          }
55
56          transform.rotation = rotacionInicial;
57          enReset = false;
58          Debug.Log("Rotaci n reseteada completamente.");
59      }
60  }
```

A2. Desplaza.cs

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Desplaza : MonoBehaviour
6   {
7       public float velocidad = 2f;
8       public float distanciaMaxima = 1f;
9
10      private Vector3 posicionInicial;
11      private Vector3 direccion = new Vector3(0, -0.04f, 1).normalized;
12
13      private bool yendo = true;
14      private bool movimientoActivo = false;
15      private bool regresando = false;
16
17      void Start()
18      {
19          posicionInicial = transform.position;
20      }
21
22      void Update()
23      {
24          if (movimientoActivo)
25          {
26              Vector3 destino = yendo ?
```

```
27                  posicionInicial + direccion * distanciaMaxima :
28                  posicionInicial;
29
30              transform.position = Vector3.MoveTowards(
31                  transform.position, destino, velocidad * Time.deltaTime);
32
33              if (Vector3.Distance(transform.position, destino) < 0.01f)
34              {
35                  yendo = !yendo;
36              }
37          }
38          else if (regresando)
39          {
40              transform.position = Vector3.MoveTowards(
41                  transform.position, posicionInicial, velocidad * Time.deltaTime);
42
43              if (Vector3.Distance(transform.position, posicionInicial) < 0.01f)
44              {
45                  transform.position = posicionInicial;
46                  regresando = false;
47                  Debug.Log("La pieza ha regresado y se ha detenido.");
48              }
49          }
50      }
51
52      public void ActivarMovimiento()
53      {
54          Debug.Log("Movimiento de vaiv n activado desde el bot n VR.");
55          this.enabled = true;
56          movimientoActivo = true;
57          regresando = false;
58      }
59
60      public void VolverAPosicionInicial()
61      {
62          Debug.Log("Regresando a la posici n inicial...");
63          movimientoActivo = false;
64          regresando = true;
65      }
66  }
```

A3. GiroEje.cs

```
1  using UnityEngine;
2
```

```
public class GiroEje : MonoBehaviour
{
    public float rotationSpeed = 100f; // Velocidad de rotaci n en grados por segundo
    private float totalRotation = 0f;

    void Update()
    {
        float rotationThisFrame = rotationSpeed * Time.deltaTime;
        transform.Rotate(-Vector3.right * rotationThisFrame);
        totalRotation += rotationThisFrame;
    }
}
```

A4. TornilloMov.cs

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

public class TornilloMov : MonoBehaviour
{
    public float descentSpeed = 0.1f;       // Velocidad de bajada/subida
    public float rotationSpeed = 100f;      // Velocidad de rotaci n
    public float stopY = 0f;                // Altura m nima (punto inferior)

    private Vector3 startPosition;          // Posici n original (punto superior)
    private bool movimientoActivo = false; // Controla si el movimiento est  activo
    private bool bajando = true;            // Direcci n actual del vaiv n
    private bool regresando = false;        // Para controlar el reset manual

    void Start()
    {
        startPosition = transform.position;
    }

    void Update()
    {
        if (movimientoActivo && !regresando)
        {
            Vector3 destino = bajando
                ? new Vector3(transform.position.x, stopY, transform.position.z)
                : startPosition;

            transform.position = Vector3.MoveTowards(transform.position, destino,
                descentSpeed * Time.deltaTime);
```

```
30              transform.Rotate(Vector3.up * rotationSpeed * Time.deltaTime, Space.Self);
31
32              if (transform.position == destino)
33              {
34                  bajando = !bajando;
35              }
36          }
37          else if (regresando)
38          {
39              transform.position = Vector3.MoveTowards(transform.position, startPosition,
                    descentSpeed * Time.deltaTime);
40              transform.Rotate(Vector3.up * rotationSpeed * Time.deltaTime, Space.Self);
41
42              if (transform.position == startPosition)
43              {
44                  regresando = false;
45                  Debug.Log("El tornillo ha vuelto a la posici n inicial.");
46              }
47          }
48      }
49
50      public void ActivarMovimiento()
51      {
52          Debug.Log("Movimiento de vaiv n activado por bot n VR.");
53          movimientoActivo = true;
54          regresando = false;
55      }
56
57      public void ResetearPosicion()
58      {
59          Debug.Log("Deteniendo y regresando a posici n inicial...");
60          movimientoActivo = false;
61          regresando = true;
62      }
63  }
```

A5. MoverPieza.cs

```
1   using System.Collections;
2   using UnityEngine;
3
4   public class MoverPieza : MonoBehaviour
5   {
6       public float velocidad = 2f; // Velocidad del movimiento
7       public float distanciaObjetivo = 0.05f; // Distancia a recorrer en X
```

```
8
9       private Vector3 posicionInicial;
10      private bool enMovimiento = false;
11      private bool regresando = false;
12
13      public MoverPiezaB piezaB;
14      void Start()
15      {
16          posicionInicial = transform.position; // Guardamos la posici n inicial
17      }
18
19      void Update()
20      {
21          if (enMovimiento)
22          {
23              // Direcci n del movimiento
24              Vector3 direccion = regresando ? Vector3.left : Vector3.right;
25
26              // Movimiento
27              transform.position += direccion * velocidad * Time.deltaTime;
28
29              // Condici n para cambiar de direcci n
30              float distanciaDesdeInicio = transform.position.x - posicionInicial.x;
31
32              if (!regresando && distanciaDesdeInicio >= distanciaObjetivo)
33              {
34                  transform.position = new Vector3(posicionInicial.x + distanciaObjetivo,
                        transform.position.y, transform.position.z);
35                  regresando = true;
36              }
37              else if (regresando && distanciaDesdeInicio <= 0f)
38              {
39                  transform.position = new Vector3(posicionInicial.x, transform.position.y,
                         transform.position.z);
40                  regresando = false;
41                  if (piezaB != null)
42                  {
43                      piezaB.AvanzarUnPaso();
44                  }
45              }
46          }
47      }
48
49      public void ActivarMovimiento()
50      {
51          this.enabled = true;
52          enMovimiento = true;
```

```
53      }
54
55      public void ReiniarPosicion()
56      {
57          enMovimiento = false;
58          transform.position = posicionInicial;
59          regresando = false;
60
61          if (piezaB != null)
62          {
63              piezaB.ResetearPosicion();
64          }
65      }
66  }
```

A6. MoverPiezaB.cs

```
1   using UnityEngine;
2
3   public class MoverPiezaB : MonoBehaviour
4   {
5       public float pasoZ = 0.01f; // Tama o del paso en Z
6       private Vector3 posicionInicial;
7
8       private void Start()
9       {
10          posicionInicial = transform.position;
11      }
12
13      public void AvanzarUnPaso()
14      {
15          transform.position += -Vector3.forward * pasoZ;
16          Debug.Log("Pieza B ha avanzado un paso en Z.");
17      }
18
19      public void ResetearPosicion()
20      {
21          transform.position = posicionInicial;
22          Debug.Log("Pieza B ha vuelto a su posici n inicial.");
23      }
24  }
```

A7. MovimientoY.cs

```
using UnityEngine;

public class MovimientoY : MonoBehaviour
{
    public float distanciaMovimiento = 0.1f; // Distancia a mover en Y
    public float velocidad = 2f; // Velocidad del movimiento

    private Vector3 posicionInicial;
    private bool enMovimiento = false;
    private bool regresando = false;

    void Start()
    {
        posicionInicial = transform.position;  // Guardamos la posici n inicial
    }

    void Update()
    {
        if (enMovimiento)
        {
            // Ahora va a comenzar movi ndose hacia abajo primero (Vector3.down)
            Vector3 direccion = regresando ? Vector3.up : Vector3.down;

            // Mueve la pieza
            transform.position += direccion * velocidad * Time.deltaTime;

            // Comprobar si ha alcanzado la posici n deseada (cuando se mueva hacia
                abajo)
            if (!regresando && transform.position.y <= posicionInicial.y -
                distanciaMovimiento)
            {
                transform.position = new Vector3(transform.position.x, posicionInicial.y
                    - distanciaMovimiento, transform.position.z);
                regresando = true; // Ahora va a regresar hacia arriba
            }
            else if (regresando && transform.position.y >= posicionInicial.y)
            {
                transform.position = new Vector3(transform.position.x, posicionInicial.y,
                     transform.position.z);
                regresando = false; // Ahora va hacia abajo nuevamente
            }
        }
    }

```

```
41      // M todo que se activa cuando pulsas el bot n VR
42      public void ActivarMovimiento()
43      {
44          this.enabled = true;
45          enMovimiento = true; // Comienza el movimiento
46      }
47
48      // M todo para resetear la posici n de la pieza
49      public void ResetearPosicion()
50      {
51          enMovimiento = false;
52          transform.position = posicionInicial; // Vuelve a la posici n inicial
53      }
54  }
```

A8. RotationZLimitedVR.cs

```
1   using System.Collections;
2   using UnityEngine;
3
4
5   public class RotationZLimitedVR : MonoBehaviour
6   {
7       public float velocidadRotacion = 30f; // Velocidad en grados por segundo
8       private float anguloInicial;
9       private float anguloObjetivo = -45f; // L mite de rotaci n en Z
10      private bool rotando = false;  // Para activar la rotaci n hacia -45
11      private bool regresando = false; // Para activar el regreso a la posici n inicial
12
13      void Start()
14      {
15          // Guardamos el  ngulo  inicial en Z
16          anguloInicial = transform.eulerAngles.z;
17      }
18
19      void Update()
20      {
21          if (rotando)
22          {
23              RotarHacia(anguloObjetivo, ref rotando);
24          }
25          else if (regresando)
26          {
27              RotarHacia(anguloInicial, ref regresando);
28          }
```

```
29      }
30
31      // Activa la rotaci n hacia -45
32      public void ActivarRotacion()
33      {
34          this.enabled = true;
35          rotando = true;
36          regresando = false; // Detenemos el regreso si estaba en curso
37      }
38
39      // Activa el regreso a la posici n inicial
40      public void ResetearRotacion()
41      {
42          regresando = true;
43          rotando = false; // Detenemos la rotaci n si estaba en curso
44      }
45
46      // Funci n auxiliar para rotar suavemente hacia un  ngulo  objetivo
47      private void RotarHacia(float anguloDestino, ref bool estado)
48      {
49          float anguloActual = transform.eulerAngles.z;
50
51          // Ajuste para evitar problemas con  ngulos  mayores a 180
52          if (anguloActual > 180) anguloActual -= 360;
53
54          // Verificamos si a n no llegamos al destino
55          if (Mathf.Abs(anguloActual - anguloDestino) > 0.1f)
56          {
57              float nuevoAngulo = Mathf.MoveTowards(anguloActual, anguloDestino,
                    velocidadRotacion * Time.deltaTime);
58              transform.rotation = Quaternion.Euler(transform.eulerAngles.x, transform.
                    eulerAngles.y, nuevoAngulo);
59          }
60          else
61          {
62              estado = false; // Detener la rotaci n cuando llega al destino
63          }
64      }
65  }
```

A9. RotacionEjeY.cs

```
1   using UnityEngine;
2   using System.Collections;
3
```

```
4  public class RotacionEjeY : MonoBehaviour
5  {
6      public float velocidadRotacion = 100f; // Velocidad configurable en el inspector
7
8      private bool girando = false;
9      private Quaternion rotacionInicial;
10     private Coroutine resetCoroutine;
11
12     void Start()
13     {
14         rotacionInicial = transform.rotation; // Guarda la rotaci n inicial
15     }
16
17     void Update()
18     {
19         if (girando)
20         {
21             // Gira continuamente sobre el eje Y
22             transform.Rotate(Vector3.up * velocidadRotacion * Time.deltaTime, Space.Self)
                   ;
23         }
24     }
25
26     // Activar giro desde el bot n VR
27     public void ActivarGiro()
28     {
29         if (resetCoroutine != null)
30         {
31             StopCoroutine(resetCoroutine); // Para el reseteo si est  en proceso
32         }
33
34         girando = true;
35         Debug.Log("Giro ACTIVADO.");
36     }
37
38     // Detener giro y volver a la rotaci n inicial desde otro bot n VR
39     public void ResetearRotacion()
40     {
41         if (girando)
42         {
43             girando = false;
44             Debug.Log("Giro detenido. Volviendo a la rotaci n inicial...");
45             resetCoroutine = StartCoroutine(VolverARotacionInicial());
46         }
47     }
48
49     private IEnumerator VolverARotacionInicial()
```

```
50      {
51          float t = 0f;
52          Quaternion rotacionActual = transform.rotation;
53
54          while (t < 1f)
55          {
56              t += Time.deltaTime;
57              transform.rotation = Quaternion.Slerp(rotacionActual, rotacionInicial, t);
58              yield return null;
59          }
60
61          transform.rotation = rotacionInicial;
62      }
63  }
```

A10. AvanceY.cs

```
1   using System.Collections;
2   using UnityEngine;
3
4   public class AvanceY : MonoBehaviour
5   {
6       public float velocidad = 2f; // Velocidad del movimiento
7       public float distanciaObjetivo = 0.05f; // Distancia a recorrer
8
9       private Vector3 posicionInicial;
10      private bool enMovimiento = false;
11      private bool regresando = false;
12
13      void Start()
14      {
15          posicionInicial = transform.position;
16      }
17
18      void Update()
19      {
20          if (enMovimiento)
21          {
22              // Direcci n invertida: hacia la izquierda primero
23              Vector3 direccion = regresando ? Vector3.right : Vector3.left;
24
25              // Movimiento
26              transform.position += direccion * velocidad * Time.deltaTime;
27
```

```
28              float distanciaRecorrida = Mathf.Abs(transform.position.x - posicionInicial.x
                    );
29
30              if (!regresando && distanciaRecorrida >= distanciaObjetivo)
31              {
32                  transform.position = new Vector3(posicionInicial.x - distanciaObjetivo,
                        transform.position.y, transform.position.z);
33                  regresando = true;
34              }
35              else if (regresando && transform.position.x >= posicionInicial.x)
36              {
37                  transform.position = posicionInicial;
38                  regresando = false;
39              }
40          }
41      }
42
43      public void ActivarMovimiento()
44      {
45          this.enabled = true;
46          enMovimiento = true;
47      }
48
49      public void ReiniarPosicion()
50      {
51          enMovimiento = false;
52          transform.position = posicionInicial;
53          regresando = false;
54      }
55  }
```

A11. MovimientoVaivenZ.cs

```
1   using UnityEngine;
2
3   public class MovimientoVaivenZ : MonoBehaviour
4   {
5       public float distancia = 1f;          // Distancia del vaiv n sobre el eje Z
6       public float velocidad = 1f;          // Velocidad del movimiento
7       private Vector3 posicionInicial;
8       private Vector3 posicionDestinoAdelante;
9       private Vector3 posicionDestinoAtras;
10
11      private bool enMovimiento = false;
12      private bool yendoAdelante = true;
```

```
13
14      void Start()
15      {
16          // Guardamos la posici n inicial
17          posicionInicial = transform.position;
18
19          // Calculamos los extremos del vaiv n en Z
20          posicionDestinoAdelante = posicionInicial + Vector3.forward * distancia;
21          posicionDestinoAtras = posicionInicial - Vector3.forward * distancia;
22      }
23
24      void Update()
25      {
26          if (enMovimiento)
27          {
28              // Determina el destino actual
29              Vector3 destinoActual = yendoAdelante ? posicionDestinoAdelante :
                    posicionDestinoAtras;
30
31              // Movimiento suave hacia el destino
32              transform.position = Vector3.MoveTowards(transform.position, destinoActual,
                    velocidad * Time.deltaTime);
33
34              // Cambia de direcci n al llegar al destino
35              if (Vector3.Distance(transform.position, destinoActual) < 0.001f)
36              {
37                  yendoAdelante = !yendoAdelante;
38              }
39          }
40      }
41
42      // Activar el movimiento desde un bot n VR
43      public void ActivarMovimiento()
44      {
45          enMovimiento = true;
46          Debug.Log("Movimiento de vaiv n en Z activado.");
47      }
48
49      // Detener el movimiento desde un bot n VR
50      public void DetenerMovimiento()
51      {
52          enMovimiento = false;
53          Debug.Log("Movimiento de vaiv n detenido.");
54      }
55
56      // Resetear la posici n desde un bot n VR y detener el movimiento
57      public void ResetearPosicion()
```

```
58      {
59          enMovimiento = false; // Detiene el movimiento
60          transform.position = posicionInicial; // Vuelve a la posici n inicial
61          yendoAdelante = true; // Reinicia la direcci n
62          Debug.Log("Objeto reseteado a la posici n inicial y movimiento detenido.");
63      }
64  }
```

A12. RotaciónControladaX.cs

```
1   using UnityEngine;
2
3   public class RotacionControladaX : MonoBehaviour
4   {
5       public float velocidad = 45f;         // Velocidad de rotaci n continua
6       public float velocidadRetorno = 2f;   // Qu  tan r pido vuelve a la posici n
            inicial
7
8       private bool rotando = false;
9       private bool regresando = false;
10
11      private Quaternion rotacionInicial;
12
13      void Start()
14      {
15          rotacionInicial = transform.localRotation;
16      }
17
18      void Update()
19      {
20          if (rotando)
21          {
22              transform.Rotate(Vector3.right * velocidad * Time.deltaTime);
23          }
24          else if (regresando)
25          {
26              // Suavemente vuelve a la rotaci n inicial
27              transform.localRotation = Quaternion.Lerp(transform.localRotation,
                    rotacionInicial, Time.deltaTime * velocidadRetorno);
28
29              // Cuando est  suficientemente cerca, termina el regreso
30              if (Quaternion.Angle(transform.localRotation, rotacionInicial) < 0.1f)
31              {
32                  transform.localRotation = rotacionInicial;
33                  regresando = false;
```

```
            }
        }
    }

    public void ActivarRotacion()
    {
        rotando = true;
        regresando = false;
    }

    public void DesactivarRotacion()
    {
        rotando = false;
        regresando = true;
    }
}
```

A13. SutilVaiven.cs

```
using UnityEngine;

public class SutilVaiven : MonoBehaviour
{
    [Header("Ajustes del vaiv n")]
    public float anguloMaximo = 5f;     // M ximo  ngulo  de oscilaci n (muy peque o)
    public float velocidad = 3f;        // Qu  tan r pido oscila

    private Quaternion rotacionInicial;
    private bool vaivenActivo = false;

    void Start()
    {
        rotacionInicial = transform.localRotation;
    }
    void Update()
    {
        if (!vaivenActivo) return;

        float zOffset = Mathf.Sin(Time.time * velocidad) * anguloMaximo;
        Quaternion rotacionVaiven = Quaternion.Euler(0f, 0f, zOffset);
        transform.localRotation = rotacionInicial * rotacionVaiven;
    }
    public void ActivarVaiven()
    {
        vaivenActivo = true;
```

```
27      }
28
29      public void DesactivarVaiven()
30      {
31          vaivenActivo = false;
32          transform.localRotation = rotacionInicial; // Vuelve a su orientaci n original
33      }
34  }
```

A14. RotacionControladaZ.cs

```
1   using UnityEngine;
2
3   public class RotacionControladaZ : MonoBehaviour
4   {
5       public float velocidad = 45f;         // Velocidad de rotaci n sobre Z (grados por
            segundo)
6       public float velocidadRetorno = 2f;  // Qu  tan r pido vuelve a la rotaci n
            inicial
7
8       private bool rotando = false;
9       private bool regresando = false;
10
11      private Quaternion rotacionInicial;
12
13      void Start()
14      {
15          rotacionInicial = transform.localRotation;
16      }
17
18      void Update()
19      {
20          if (rotando)
21          {
22              transform.Rotate(Vector3.forward * velocidad * Time.deltaTime);
23          }
24          else if (regresando)
25          {
26              transform.localRotation = Quaternion.Lerp(transform.localRotation,
                    rotacionInicial, Time.deltaTime * velocidadRetorno);
27
28              if (Quaternion.Angle(transform.localRotation, rotacionInicial) < 0.1f)
29              {
30                  transform.localRotation = rotacionInicial;
31                  regresando = false;
```

```
32              }
33          }
34      }
35
36
37      public void ActivarRotacion()
38      {
39          rotando = true;
40          regresando = false;
41      }
42
43
44      public void ResetearRotacion()
45      {
46          rotando = false;
47          regresando = true;
48      }
49  }
```

A15. botonVR.cs

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.Events;
5
6   public class botonVR : MonoBehaviour
7   {
8       public GameObject button;
9       public UnityEvent onPress;
10      public UnityEvent onRelease;
11      GameObject presser;
12      AudioSource sound;
13      bool isPressed;
14
15      void Start()
16      {
17          sound = GetComponent<AudioSource>();
18          isPressed = false;
19      }
20      private void OnTriggerEnter(Collider other)
21      {
22          if (!isPressed)
23          {
24              button.transform.localPosition = new Vector3(0, 0.003f, 0);
```

```
25              presser = other.gameObject;
26              onPress.Invoke();
27              sound.Play();
28              isPressed = true;
29          }
30      }
31      private void OnTriggerExit(Collider other)
32      {
33          if(other.gameObject == presser)
34          {
35              button.transform.localPosition = new Vector3(0, 0.3f, 0);
36              onRelease.Invoke();
37              isPressed = false;
38          }
39      }
40      public void PruebaBoton()
41      {
42          Debug.Log(" El  bot n VR ha sido pulsado!");
43      }
44  }
```

B. Pósteres

En el presente Anexo se muestran los distintos pósteres informativos que han sido incorporados junto a cada máquina-herramienta en el entorno de realidad virtual. Estos recursos tienen como objetivo proporcionar al usuario una experiencia más completa. Cada póster ha sido ubicado dentro del museo, junto a la máquina-herramienta correspondiente.

MUSEO VIRTUAL DE LA ESCUELA DE INGENIERÍAS INDUSTRIALES

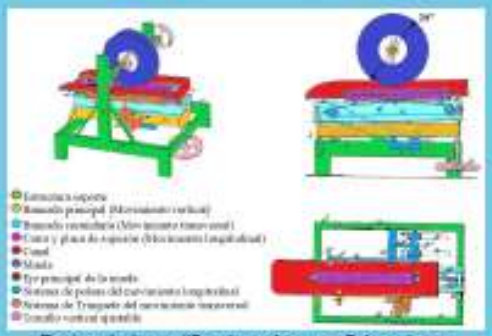
RECTIFICADORA SUPERF. PLANAS DARLING (1853)

VALOR HISTÓRICO

- **Primera rectificadora de superficies planas.**
- Atribuida a **Samuel L. Darling y patentada en 1853** (EE. UU.).
- Aunque ya **existían otras máquinas herramientas** para el mecanizado de este tipo de superficies, tales como la **cepilladora** de Richard Roberts (1817) o la **fresadora** de Eli Whitney (1818), ninguna de ellas era capaz de proporcionar las **tolerancias necesarias** en superficies planas para componentes de máquinas con **requerimientos de precisión** cada vez mayores.
- No se conservan **ninguno** de estos primeros modelos de rectificadora Darling, siendo la patente la única fuente de información de sus características y funcionamiento.

Rectificadora de superficies planas "Darling" (patentada en 1853)

Patente (1853)



Partes de la rectificadora de superficies planas

1. **Movimiento principal de corte** rotativo ejecutado por la muela.
2. **Movimiento de avance** longitudinal alternativo ejecutado por la pieza
3. Movimiento de ajuste vertical de la **profundidad de corte**, ejecutado por la pieza

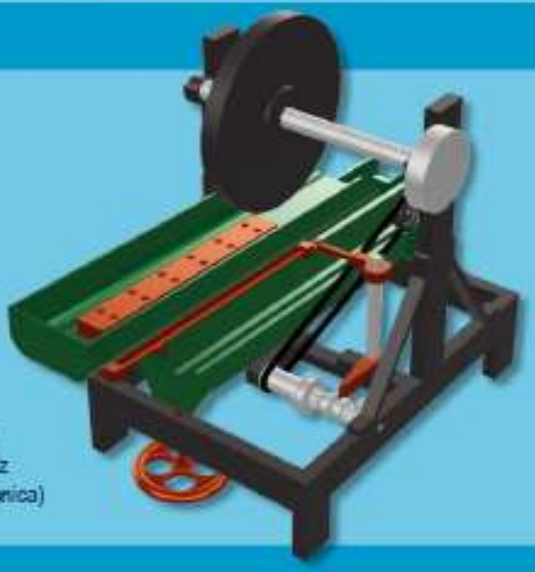
Sistema de poleas para desplazamiento longitudinal alternativo de la pieza

Sistema de trinquete para movimiento lateral de la pieza, coordinado con el movimiento alternativo longitudinal

Sistema de tornillo para ajuste vertical

MODELO VIRTUAL

Modelo realizado por
Manuel Martín Vázquez
(Grado en Ingeniería Mecánica)



Iniciativa desarrollada en el marco del Proyecto de Innovación Educativa (PIE22-094)
Desarrollo y empleo de herramientas docentes a través de la creación del Museo Virtual de la EII de la UMA

MUSEO VIRTUAL DE LA ESCUELA DE INGENIERÍAS INDUSTRIALES

PRENSA DE HUSILLO A FRICCIÓN "DELALANDE" (1808)

VALOR HISTÓRICO

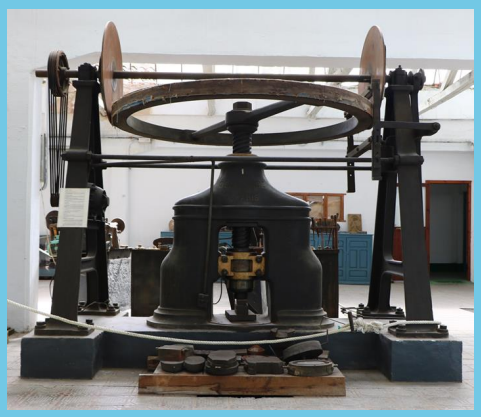
Fabricada en **París en 1808** por la empresa **Delalande**. Se encuentra ubicada en el **Museo de las Reales Fábricas de San Juan de Alcaraz en Riópar** (Albacete), en excelente estado de conservación.

Adquirida en **1826**, fue utilizada inicialmente como ***prensa manual de balancín*** para la fabricación de bajorrelieves de latón mediante **acuñado**.

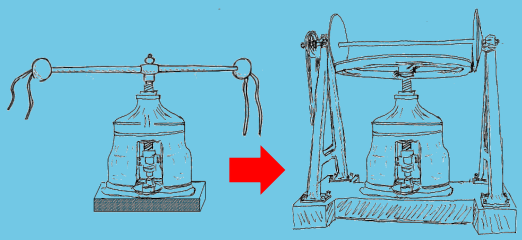
Fue modificada, incorporándole una estructura autoportante externa para el soporte de los discos de fricción y del volante de inercia, movida inicialmente por energía hidráulica, y posteriormente mediante un motor eléctrico, en su fisonomía actual (***husillo a fricción***).

Este hecho la dota de un **excepcional valor** histórico y patrimonial e **icono** del Museo.

Ha sido utilizada para la fabricación de cubertería de alpaca, plata y acero inoxidable mediante **estampación** hasta el año **1996**, cuando la fábrica cesa su actividad.

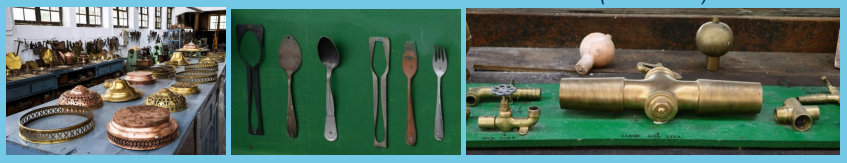


Prensa de husillo a fricción Delalande (1808)
Museo de las Reales Fábricas de
San Juan de Alcaraz de Riópar (Albacete)

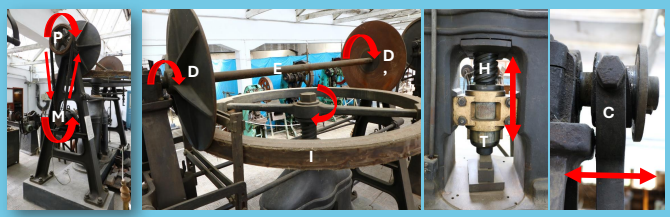


Evolución: *prensa manual de balancín (acuñado) a prensa de husillo a fricción (estampación)*

Fábricas de San Juan de Alcaraz de Riópar (1772-1996)



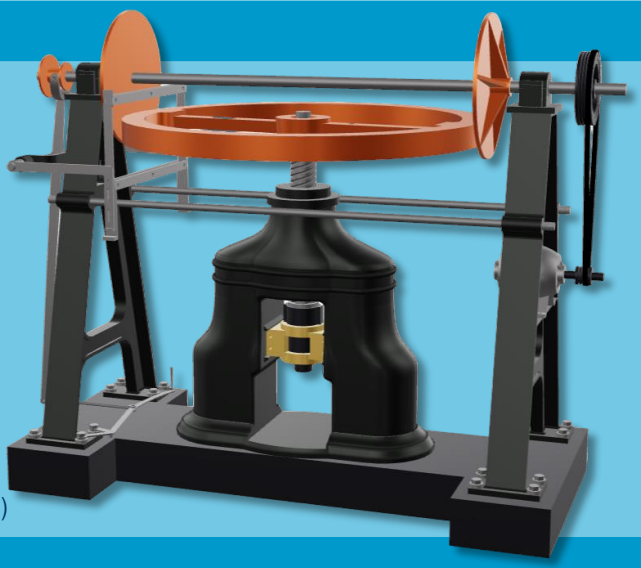
Piezas estampadas de latón, alpaca, plata y acero inoxidable



- El motor (M) acciona la polea (P) → giro del eje (E) y de los discos de fricción (D).
- Solo uno de los discos (D) está en contacto con el volante de inercia (I) → giro en un sentido.
- El husillo (H) gira de forma solidaria al volante (I) → movimiento lineal (descenso) de la herramienta (T).
- Impacto de la herramienta (T) en la pieza → transformación de la energía inercial en deformación del material.
- El embrague (C) manual → cambio de sentido de giro del volante (I) por contacto con el disco de fricción (D') opuesto → Subida de la herramienta (T)

MODELO VIRTUAL

Modelo realizado por
Carlos Sánchez Linares
(Máster en Ingeniería Industrial)



Iniciativa desarrollada en el marco del Proyecto de Innovación Educativa (PIE22-094)
Desarrollo y empleo de herramientas docentes a través de la creación del Museo Virtual de la EII de la UMA

MUSEO VIRTUAL DE LA ESCUELA DE INGENIERÍAS INDUSTRIALES

LIMADORA (1960)

VALOR HISTÓRICO

Limadora de fabricación española (**Insuna S.A.**)

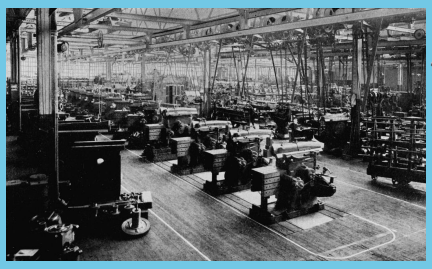
Data de los años 60 y fue donada a la **Escuela de Ingenierías Industriales** de la **Universidad de Málaga** tras el cierre de la factoría jiennense **Santana Motor S.L.**

Remodelada en 2018, en la actualidad se utiliza como recurso educativo en asignaturas de Ingeniería de los Procesos de fabricación de diferentes grados para explicar **máquinas herramienta de movimiento rectilíneo**.

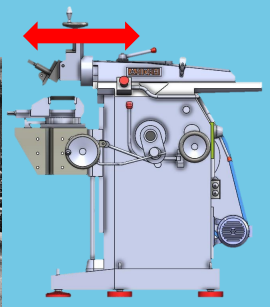
Limadora de la EII, 1960

James Nasmyth (1808-1890)

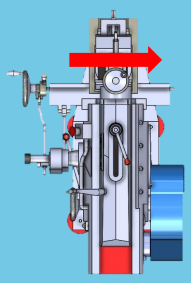
Joseph R. Whitworth (1803-1887)



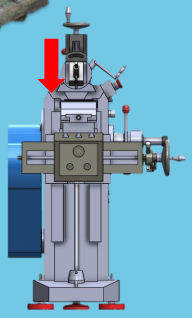
Hendey Machine Co., 1921 (1870-1954)



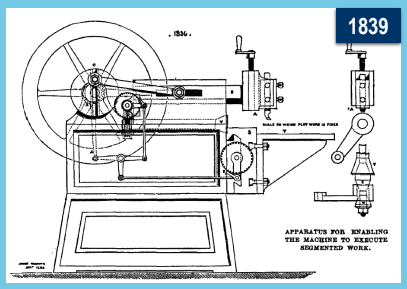
Movimiento ppal. de corte



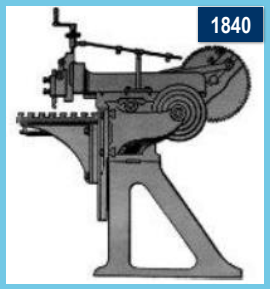
Mov. avance



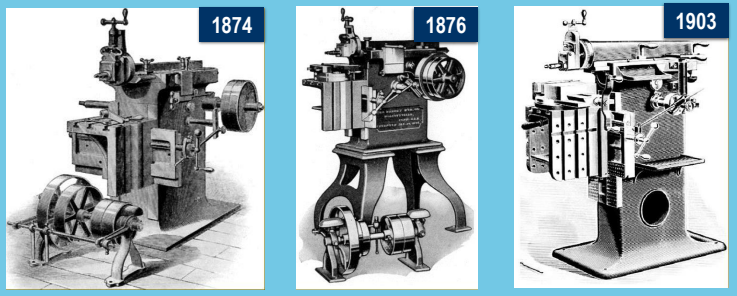
Mov. profundidad de corte



Limadora de J. Nasmyth (1839)



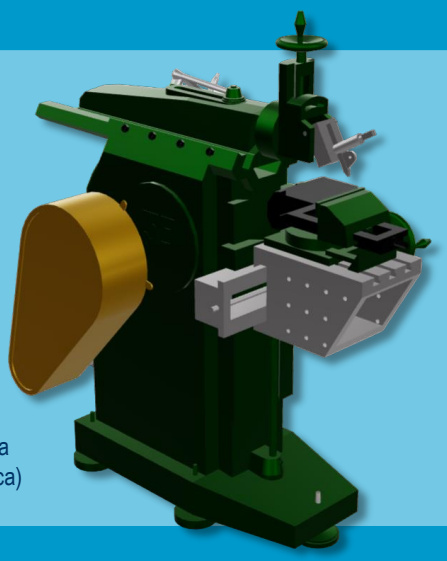
Limadora de R. Whitworth(1840)



Evolución histórica de la limadora (1839-1903)

MODELO VIRTUAL

Modelo realizado por
Eduardo José Botella Ortega
(Grado en Ingeniería Mecánica)



Iniciativa desarrollada en el marco del
Proyecto de Innovación Educativa (PIE22-094)
Desarrollo y empleo de herramientas docentes a través de la creación del Museo Virtual de la EII de la UMA

MUSEO VIRTUAL DE LA ESCUELA DE INGENIERÍAS INDUSTRIALES

FRESADORA UNIVERSAL DE BROWN (1865)

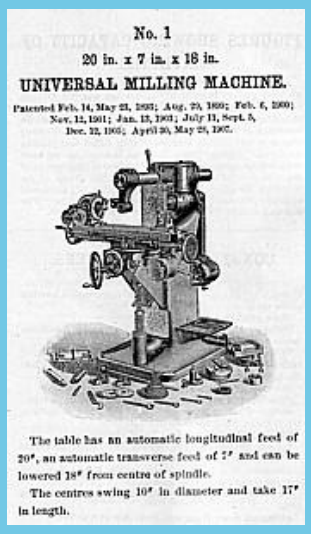
VALOR HISTÓRICO

El desarrollo de la **primera fresadora universal** en **1848**, de **Frederick W. Howe**, permitió realizar operaciones de mecanizado en **superficies planas y curvas**, así como el **giro de la pieza** en diferentes ángulos.

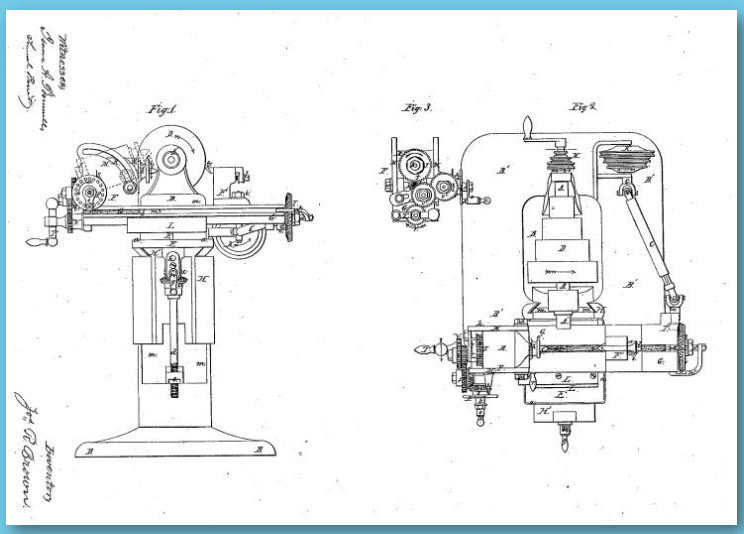
En **1853**, **Brown & Sharpe Co**. incorpora el **plato divisor** a la fresadora universal, a instancias de Howe, permitiendo la fabricación de engranajes.

La fresadora de **J. R. Brown**, patentada en **1865**, presenta **mejoras** en los dispositivos de sujeción de la pieza, versatilidad y comodidad de funcionamiento. Esta máquina se encuentra actualmente en el American Precisión Museum (Vermont, EE.UU.).

Joseph R. Brown
(1810-1876)



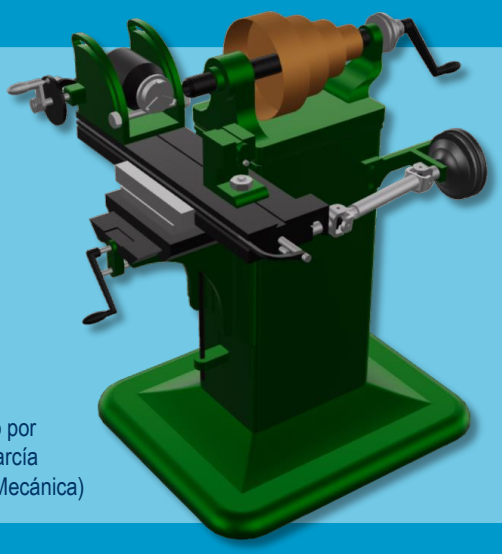
Catálogo (1906, EE.UU.)



Patente (1865, EE.UU.)

Modelo original de 1865
(American Precisión Museum, EE.UU.)

MODELO VIRTUAL



Modelo realizado por
Marina Porras García
(Grado en Ingeniería Mecánica)

Iniciativa desarrollada en el marco del
Proyecto de Innovación Educativa (PIE22-094)
Desarrollo y empleo de herramientas docentes a través de la creación del Museo Virtual de la EII de la UMA

MUSEO VIRTUAL DE LA ESCUELA DE INGENIERÍAS INDUSTRIALES

RECTIFICADORA UNIVERSAL BROWN (1877)

VALOR HISTÓRICO

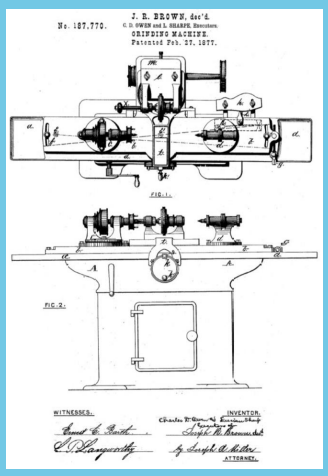
Primera rectificadora capaz de mecanizar **cualquier tipo de geometría**: superficies planas, cilindros, superficies cónicas, curvas y geometrías irregulares, tanto exteriores como interiores.

Es una de las primeras máquinas-herramienta concebidas para **obtener precisión** en lugar de primar la productividad.

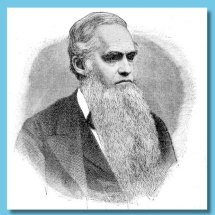
Joseph R. Brown comienza su diseño en **1868**, presentándose en 1876 en la exposición de máquinas de París. Su **patente llegó en 1877**, justo después de la muerte de Brown.

Su desarrollo es fruto del **trabajo de distintos técnicos** (Samuel Darling y Oscar J. Beale, entre otros), siendo Charles H. Norton quien revoluciona su diseño para adaptarla a grandes niveles de producción, manteniendo su precisión.

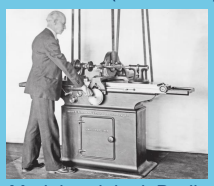
Junto con la rectificadora pesada de Norton, ha sido una máquina-herramienta clave en el desarrollo de **industrias** tan importantes como el **automóvil**, el **ferrocarril** o la **bicicleta**.



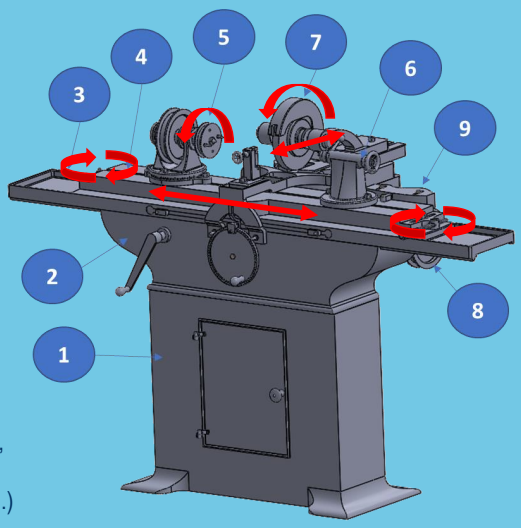
Patente (1877)

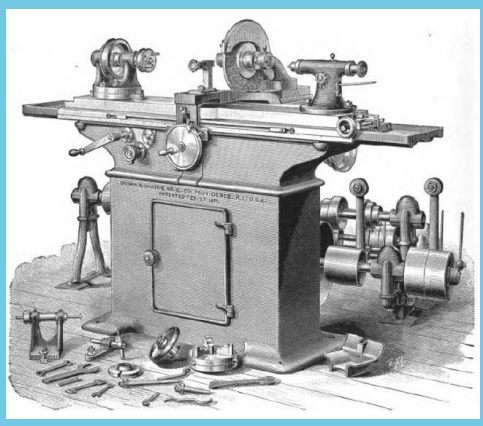


J. R. Brown (1810-1876)



Modelo original. Darling, Brown & Sharpe Co. (Providence, R.I. (EE.UU.)

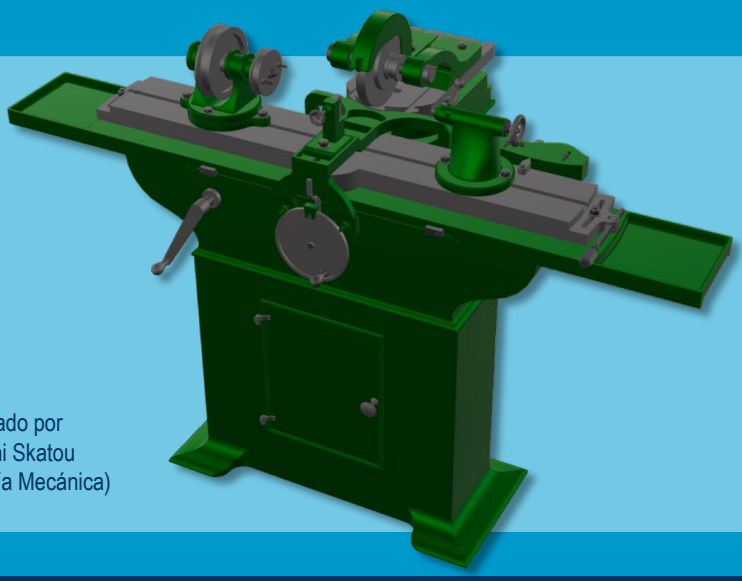




Rectificadora Universal (patentada en 1877)

1. **Bancada inferior**: Soporte del resto de elementos. Fundición de acero. Muy pesada para evitar vibraciones.
2. **Bancada estacionaria**: Alojamiento para los mecanismos de avance y profundidad de corte.
3. **Bancada de movimiento alternativo**: Permite el movimiento longitudinal alternativo de la pieza respecto de la muela.
4. **Mesa ajustable**: Permite la rotación de la pieza respecto de la muela en el plano horizontal.
5. **Mordaza de sujeción de la pieza**. Movimiento rotario de la pieza respecto de su eje.
6. **Contrapunto** para sujeción de piezas esbeltas.
7. **Muela**. Movimiento principal de corte rotatorio.
8. **Sistema de transmisión** del movimiento longitudinal de la muela.
9. **Plantilla** de copiado de contornos.

MODELO VIRTUAL



Modelo realizado por
Omar El Lahiani Skatou
(Grado en Ingeniería Mecánica)

Iniciativa desarrollada en el marco del
Proyecto de Innovación Educativa (PIE22-094)
Desarrollo y empleo de herramientas docentes a través de la creación del Museo Virtual de la EII de la UMA

MUSEO VIRTUAL DE LA ESCUELA DE INGENIERÍAS INDUSTRIALES

TALADRADORA COLUMNA WHITWORTH (1847)

VALOR HISTÓRICO

Sir Joseph Whitworth (1803-1887)

Primera taladradora de columna, patentada por **Sir Joseph Whitworth** en **1847** (UK).

Máquina-herramienta **imprescindible en cualquier fábrica** o taller, que mantiene hoy en día la misma arquitectura que la desarrollada a finales del s. XIX por Whitworth.

Inicialmente concebida para la ejecución del **avance** de la broca de forma **manual**, utilizando un contrapeso para el retorno automático de la broca.

Evoluciona de forma rápida a un sistema de **avance automático**, coordinado con el movimiento de giro de la broca.

Uno de los primeros modelos con avance automático se conserva en el Museo de Ciencias de Londres.

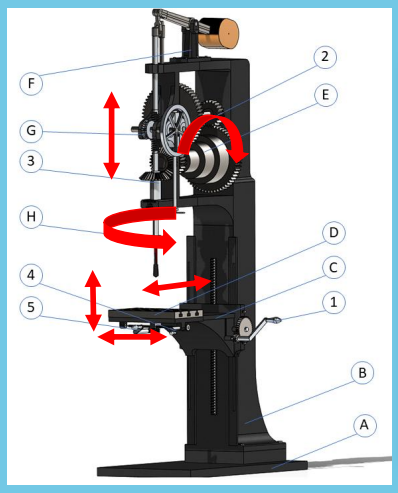
Taladradora de columna de avance automático
J. Whitworth & Co. (1860, Manchester, UK).
Museo de Ciencias de Londres

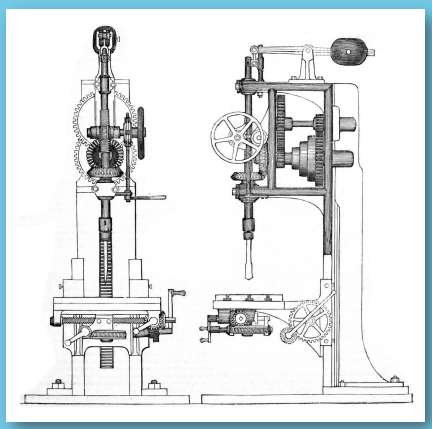
Componentes:

A. Base
B. Columna
C. Soporte mesa de trabajo
D. Mesa de trabajo
E. Transmisión de potencia (4 velocidades)
F. Sistema de contrapeso
G. Sistema de avance de la herramienta
H. Cabezal portaherramientas

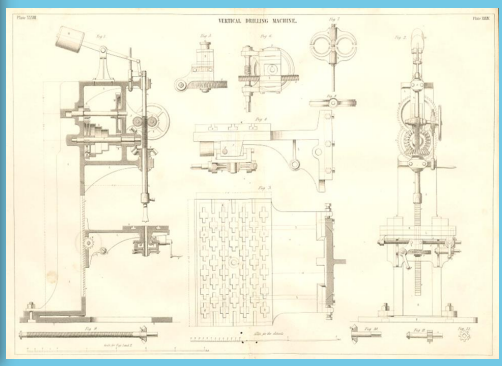
Accionamientos:

1. Manivela avance vertical mesa de trabajo
2. Manivela avance vertical cabezal
3. Engranaje troncocónico transformación giro eje horizontal en giro eje vertical
4. Manivela movimiento horizontal mesa de trabajo
5. Manivela giro mesa de trabajo



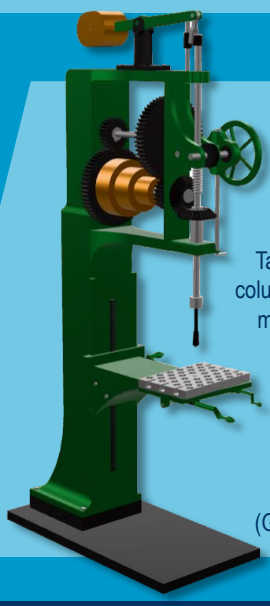


Taladradora de columna de avance manual

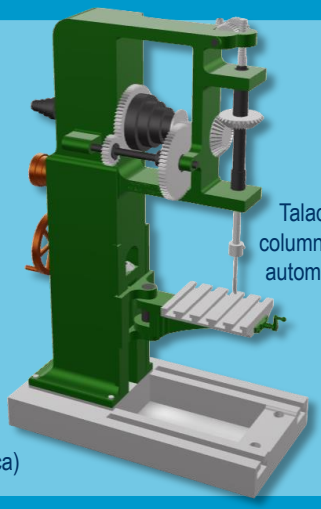


Taladradora de columna manual
(The Engineer and The Machinist's Assistant, 1847)

MODELO VIRTUAL



Taladradora de columna de avance manual (1847)



Taladradora de columna de avance automático (1860)

Modelo realizado por
Antonio Rojas López
(Grado en Ingeniería Mecánica)

Iniciativa desarrollada en el marco del
Proyecto de Innovación Educativa (PIE22-094)
Desarrollo y empleo de herramientas docentes a través de la creación del Museo Virtual de la EII de la UMA