UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

TRABAJO FIN DE MÁSTER

CURSO 2018/19

DESARROLLO DE UN SIMULADOR DE EVENTOS DISCRETOS EXPERIMENTAL BASADO EN REALIDAD VIRTUAL

Máster en Ingeniería Industrial

ALUMNO

Javier Pernas Álvarez

TUTOR

Diego Crespo Pereira

FECHA

JULIO 2019

RESUMEN

Desarrollo de un simulador de eventos discretos experimental basado en realidad virtual

En el presente trabajo se lleva a cabo el desarrollo de un simulador experimental de eventos discretos en un entorno de realidad virtual, empleando para ello el motor de videojuegos Unity3D. En este sentido, inicialmente se crea una serie de librerías que permiten modelar procesos de fabricación. A continuación, tras una revisión del estado del arte, se diseñó una interfaz gráfica de usuario con algunas de las principales formas de interacción con el mundo virtual empleadas en la actualidad. Tras una fase de conexión entre el simulador y la interfaz, se llevó a cabo una fase de ensayos con varias personas con conocimientos en los softwares actuales de simulación de eventos discretos. En la última parte del proyecto, se desarrolló un caso de demostración y se implementó un algoritmo de programación de tareas que permitió validar el simulador y analizar su potencial como herramienta de simulación de eventos discretos.

RESUMO

Desenvolvemento de un simulador de eventos discretos experimental baseado en realidade virtual

No presente traballo lévase a cabo o desenvolvemento dun simulador experimental de eventos discretos nun entorno de realidade virtual, empregando o motor de videoxogos Unity3D. Para iso, primeiramente creáronse unha serie de librarías que permiten modelar procesos de fabricación. Deseguido, tras una revisión do estado da arte, diseñouse unha interface gráfica de usuario con algunhas das principais formas de interacción entre có mundo virtual que se empregan na actualidade. Tras una fase de conexión do simulador coa interfaz, varias persoas con experiencia en simulación de eventos discretos levaron a cabo ensaios có simulador, probando a súas funcionalidades. Na derradeira parte do proxecto, desenvolveuse un caso de demostración e implementouse un algoritmo de programación de tarefas que permitiu validar o simulador e analizar o seu potencial como ferramenta de simulación de eventos discretos.

ABSTRACT

Development of an experimental discrete-event simulation simulator based on virtual reality

Within the following project, a discrete event experimental simulator based on virtual reality has been developed by using the video game engine Unity3D. Initially, a set of programming libraries were developed with the aim of modelling manufacturing processes as well as taking advantage of the capabilities of games engines. Afterwards, a VR Graphical User Interface (VR GUI) was designed with a view to the state-of-the-art ways of VR interaction which main video games include nowadays. After coupling the interface and the simulator with each other, some people, experienced with discrete-event simulation, were selected to test the aforementioned simulator and to respond to a brief survey. Eventually, a demonstration case was built by developing a task sorting algorithm which allowed to validate the simulator and to deeply analyse the future potential of such a DES tool.

Escola Politécnica Superior

TRABAJO FIN DE GRADO/MÁSTER

CURSO 2018/19

DESARROLLO DE UN SIMULADOR DE EVENTOS DISCRETOS EXPERIMENTAL BASADO EN REALIDAD VIRTUAL

Máster en Ingeniería Industrial

Documento

MEMORIA

Índice

Capítulo 1 13
1.1 Introducción 13
1.2 Objetivo del proyecto 14
1.3 Recursos del proyecto 14
Capítulo 2 17
2.1 Contexto actual 17
2.2 Fabricación inteligente (*Smart manufacturing)* 18
2.2.1 Fábrica digital y gemelo digital 19
2.2.2 Fábrica inteligente 20
2.2.3 Fábrica virtual 22
Capítulo 3 23
3.1 Software de eventos discretos 23
3.1.1 Retos actuales de la simulación 24
3.1.2 Evolución del software de simulación 25
3.2 Realidad Virtual 26
3.2.1 La realidad virtual en la industria 28
3.2.2 Realidad virtual en simulación 29
Capítulo 4 31
4.1 Trabajo de partida 31
4.2 Desarrollo del código 33
4.2.1 Depuración y mejora del código de partida 33
4.2.2 Ampliación del código de partida 35
4.2.2.1 Taller Virtual 35
4.2.2.2 Fábrica Virtual 38
4.2.2.3 Fábrica Virtual 2 41
4.3 Desarrollo de la interfaz 44
4.3.1 Readaptación de recursos existentes 44
4.3.2 Creación de los menús 47
4.3.3 Ensayos 51
4.4 Caso de demostración 54
4.4.1 Descripción del caso 54
4.4.2 Objetivos y recursos 3D 57
4.4.3 Desarrollo del algoritmo de programación de tareas 59
4.4.4 Resultados y experimentación 60

Capítulo 5 65
5.1 Conclusiones 65
5.2 Trabajos futuros 66
Capítulo 6 69
6.1 Bibliografía 69

Índice de Figuras

Figura 1. Logo del motor de videojuegos Unity3D. Fuente: www.unity3d.com. 15

Figura 2. Sistema de realidad virtual Oculus Rift. Fuente: www.amazon.es. 15

Figura 3. Transformación de la cadena de producción. Fuente: Deloitte analysis. 19

Figura 4. Características fundamentales de la fábrica inteligente según Deloitte. Fuente: Deloitte analysis. 21

Figuras 5 y 6. Máscara HMD multisensorial como emisor de olores (izquierda) y guantes hápticos de realidad virtual. Fuentes (respectivamente): www.uploadvr.com y www.feelreal.com. 27

Figura 7. Sistema tipo CAVE. Fuente: www.vrfitnessinsider.com 27

Figura 8. Diagrama explicativo antiguo de la interacción nivel lógico – nivel virtual – Unity3D del proyecto de partida (Pernas Álvarez, 2017). Fuente: (Pernas Álvarez, 2017, p. 37). 31

Figura 9. Diagrama explicativo modificado de la interacción nivel lógico – nivel virtual – Unity3D con las clases nuevas y los distintos cambios realizados en el código. Se han incluido las clases principales con los que se han desarrollado los distintos casos, si bien no se incluyen en el diagrama todas las librerías existentes en el proyecto. Fuente: Elaboración propia. 32

Figura 10. Detalle de colocación de los ítems sobre el modelo de la cola acorde a las dimensiones de estas. Fuente: Elaboración propia 34

Figura 11. Detalle de avance de los ítems en la estación de trabajo. Fuente: Elaboración propia. 34

Figura 12. Detalle de panel de configuración de la clase «UnityScheduleSource», con posibilidad de selección del modo de llegada. Fuente: Elaboración propia. 35

Figura 13. Vista general del modelo completo del «Taller Virtual». Fuente: Elaboración propia. 36

Figura 14. Diagrama de flujo del proceso del proyecto «Taller Virtual». Fuente: Elaboración propia. 36

Figura 15. La cola «UnityGateQueue» esperando por la orden de fabricación del usuario. Fuente: Elaboración propia. 37

Figura 16. Vista en detalle del funcionamiento de la cinta transportadora y el ensamblador múltiple. Fuente: Elaboración propia. 38

Figura 17. Vista general del modelo completo del «Fábrica Virtual». Fuente: Elaboración propia. 39

Figura 18. Diagrama de flujo de producción del proyecto «Fábrica Virtual». Fuente: Elaboración propia. 39

Figura 19. Detalle de la malla de navegación para la implementación del algoritmo A*. Fuente: Elaboración propia. 40

Figura 20. Detalle del enrutamiento de la carretilla elevadora empleando el algoritmo A*. Fuente: Elaboración propia. 41

Figura 21. Diagrama de flujo del proceso del proyecto «Fábrica Virtual 2», con la inclusión del operador. Fuente: Elaboración propia. 41

Figura 22. Modelo 3D de humanoide del paquete de Unity «Characters» empleado en el proyecto. Fuente: Elaboración propia. 42

Figura 23. Escena específica donde es posible configurar el *avatar* del operador humanoide. Fuente: Elaboración propia. 42

Figura 24. Parámetros (izquierda) y diagrama de estados (derecha) de la animación del operador. Fuente: Elaboración propia. 43

Figura 25. Vista en primera persona de los operadores en «Fábrica Virtual 2». Fuente: Elaboración propia. 43

Figura 26. Detalle de las manos virtuales desde la perspectiva del usuario. Fuente: Elaboración propia. 45

Figura 27. Detalle del menú principal desde el punto de vista del usuario. El aspecto es el mismo que el del recurso de Oculus Sample Framework. Fuente: Elaboración propia. 46

Figura 28. Detalle del teclado virtual desde la perspectiva del usuario. Fuente: Elaboración propia. 46

Figura 29. Detalle de la creación de una carretilla elevadora desde la perspectiva del usuario. Fuente: Elaboración propia. 47

Figura 30. Detalle de la selección de elementos durante la creación de las conexiones. El recuadro de selección verde representa el *input* de la conexión, y el amarillo el *output*. Fuente: Elaboración propia. 48

Figura 31. Detalle del panel informativo sobre el éxito de destrucción de un elemento. Fuente: Eliminación propia. 48

Figura 32. Detalle del panel de configuración de la carretilla elevadora. Fuente: Elaboración propia 49

Figura 33. Detalle del panel de configuración del tiempo del modelo. Fuente: Elaboración propia 49

Figura 34. Estructura jerárquica de las distintas clases de la librería «UnityFunctions». Fuente: Elaboración propia. 50

Figura 35.Desglose del menú principal en submenús y botones. Fuente: Elaboración propia 50

Figura 36. Caso propuesto de ensayos con el simulador. Fuente: Elaboración propia…51

Figura 37. Imágenes de dos de los participantes en la fase de pruebas del simulador, durante las pruebas con el mismo. Fuente: Elaboración propia. 52

Figura 38. Diagrama de flujo simplificado del caso de demostración. Recuadrada en verde, la tarea que se emplea para el caso de demostración. Fuente: Elaboración propia. 55

Figura 39. Diagrama de Gantt con la planificación *real* del caso de demostración. 55

Figura 40. Modelos 3D de las columnas del caso de demostración proporcionadas por la UMI. Fuente: Elaboración propia. 57

Figura 41. Modelo 3D del astillero del caso de demostración proporcionado por la UMI. Fuente: Elaboración propia. 58

Figura 42. Modelo 3D empleado para representar las celdas de fabricación de las columnas, sobre la zona del taller donde se hayan en la realidad. Fuente: Elaboración propia. 58

Figura 43. Detalle de la perspectiva del usuario en el mundo virtual mientras se fabrican las semicolumnas. Fuente: elaboración propia. 60

Figura 44. Detalle de panel final con la duración de la fabricación de las semicolumnas. Fuente: Elaboración propia. 60

Figura 45. Diagrama de Gantt con los tiempos de planificación generados por el algoritmo. Fuente: Elaboración propia. 61

Figura 46. Panel de configuración de un elemento tipo «UnityScheduleSource» con los nuevos parámetros del algoritmo. Fuente: Elaboración propia. 62

Figura 47. Diagrama de Gantt con los tiempos de planificación cambiando las prioridades del algoritmo. Fuente: Elaboración propia. 62

Figura 48. Detalle del panel de configuración de la estación de trabajo, con el botón «Block Station» que permite bloquear la estación seleccionada. Fuente: Elaboración propia. 63

Figura 49. Diagrama de Gantt con los tiempos de planificación bloqueando las estaciones 8 y 9 a la semana 7. Fuente: Elaboración propia. 63

Índice de Tablas

Tabla 1. Tabla con las valoraciones medias de los participantes en la encuesta. Fuente: Elaboración propia. 53

Tabla 2. Fechas de llegada de las semicolumnas según el número de barco. Fuente: Elaboración propia. 56

Capítulo 1

1.1 Introducción

En la actualidad, no existe duda alguna de que la simulación de eventos discretos (DES, por sus siglas en inglés y, en adelante, simulación) se ha convertido en la herramienta esencial sobre la cual se asienta el concepto de fábrica inteligente («*Smart Factory*»), uno de los pilares de la *Industrial 4.0*. Considerada por muchos no sin controversia la "4ª Revolución Industrial", lo que sí es cierto es que hoy en día nos encontramos en un entorno sumamente cambiante y de una muy alta competitividad donde términos como *Big Data*, *la nube*, *IOT (Internet de las cosas*, por sus siglas en inglés), o *fabricación aditiva* entre otros, forman ya desde hace tiempo parte de la jerga profesional relativa a todo aquello que concierne las tecnologías de la información.

Junto con el avance incesante de la tecnología, se consolidan y actualizan paralelamente técnicas de organización y gestión de proyectos, desde el *Lean Manufacturing* de los años 90, con los conceptos de *Just in Time* o *Kanban*, hasta la filosofía *Agile* de los 2000, que recoge muchos conceptos del mundo *Lean*. En cualquier caso, dentro de la organización de proyectos, todo sistema de gestión de la producción debe saber aprovechar al máximo los recursos que tiene a su alcance, de manera que la toma de decisiones (el conocido como *decision-making*) y la comunicación de las mismas sea lo más clara, rápida y eficiente posible.

Entre estos recursos, se extiende cada vez con más fuerza la ya mencionada simulación de eventos discretos. La posibilidad de obtener un modelo virtual de lo que sucede en la realidad supone un avance muy importante tanto en la optimización y diseño de procesos industriales como en el análisis de riesgos y, de manera más general, la gestión de proyectos. Sin embargo, aunque cuenta con más de medio sigo de desarrollo, la incorporación del 3D y los grandes esfuerzos por parte de algunas compañías privadas en su mejora han supuesto el punto de inflexión para que su credibilidad y aceptación como herramienta de apoyo en la toma de decisiones vaya en aumento.

Precisamente, en relación con esto último y desde el punto de vista global, nos hallamos en un punto intermedio de esta *revolución* donde los mínimos exigibles en la creación y desarrollo de cualquier aplicación tecnológica se incrementan año a año a una velocidad inimaginable tan solo 10 años atrás. Cabe decir también que paralelamente sucede lo mismo con los medios que tenemos a nuestra disposición para acometer tal tarea. No obstante, en relación con lo mencionado previamente, los requerimientos mínimos relativos a la parte gráfica de cualquier *software* que contenga algún componente visual, como es nuestro caso, son cada vez más elevados; incluso en situaciones donde esta rigurosidad obedece exclusivamente a fines de *marketing*.

Junto con el aspecto gráfico y estrechamente relacionado con él se encuentra también la revolución en términos de interactividad, de comunicación hombre-máquina, técnicamente denominado como interfaz gráfica de usuario. Esta debe ser lo más intuitiva y comprensible posible − lo que se conoce como manejabilidad, especialmente en el mundo de los videojuegos − de manera que el programa en cuestión pueda ser manejado de manera sencilla y rápida. Entran en juego aquí dos tecnologías con muchos puntos en común como son la realidad virtual (VR, por sus siglas en inglés) y la realidad aumentada (VA, por sus siglas en inglés). Con cada vez mayor número de campos de aplicación, se asumen como ya implantadas y sobradamente extendidas en el mundo de los videojuegos, a pesar de que su rentabilidad, así como accesibilidad en términos económicos no haya alcanzado aún los niveles que se le presuponían en un principio.

Particularizando para el amplio campo de la ingeniería, están llamadas a suponer una revolución en los términos ya mencionados, desde el diseño CAD en tres dimensiones hasta la validación de modelos de simulación. El objetivo final evidentemente es el de facilitar, simplificar y, por tanto, acelerar la realización de operaciones y tareas y, en consecuencia,

mejorar la productividad de cualquier empresa. Todo ello sin ignorar el componente atractivo y el papel comercial que poseen estas tecnologías, tanto por la novedad que aún hoy en día siguen constituyendo como por el salto cualitativo que suponen.

Con todo lo anterior, en este trabajo se pretende dar un paso más en lo que se refiere a las aplicaciones y usos de la realidad virtual en la simulación de una manera que, como se explicará más adelante, aún no ha sido explorada hasta el momento.

1.2 Objetivo del proyecto

Este proyecto se sitúa como una continuación del Trabajo Fin de Grado (Pernas Álvarez, 2017) donde se estudia la posibilidad de aprovechar las ventajosas capacidades de los motores de videojuegos en la creación de modelos de simulación. Aunque no contemplado entre los trabajos futuros planteados, en el trabajo mencionado ya se había experimentado brevemente con entornos de realidad virtual, si bien se había hecho una readaptación de la interfaz gráfica de usuario en 3D para poder ejecutar la aplicación con un visor y un teléfono móvil.

Por tanto, partiendo de esta base, el **objetivo principal** de este Trabajo Fin de Máster es el de desarrollar un simulador virtual de eventos discretos experimental a través de un motor de videojuegos que permita desarrollar y simular un modelo con tres condiciones:

- **exclusivamente** dentro del mundo virtual;
- de manera autónoma;
- y sin necesidad de recurrir a otros periféricos de salida que no sean los del sistema de realidad virtual empleado.

Este objetivo se puede desglosar brevemente en los siguientes hitos:

- Ampliación de las librerías de simulación en C# desarrolladas en el Trabajo Fin de Grado del que parte este trabajo, incorporando otros elementos comunes existentes en muchos procesos industriales como pueden ser el ensamblador o la carretilla elevadora, con el fin de mejorar las funcionalidades del simulador. Este objetivo ya se incluía entre los trabajos futuros del trabajo de partida.
- Desarrollo de una interfaz gráfica de usuario simple e intuitiva controlada exclusivamente a través de los controladores y las gafas de realidad virtual empleadas en el proyecto.
- Aplicación de los recursos desarrollados a un caso de demostración que permita ilustrar adecuadamente las funcionalidades desarrolladas. Este nos permitirá obtener las conclusiones acerca de las ventajas o inconvenientes de este uso de la realidad virtual en el campo de los eventos discretos.

1.3 Recursos del proyecto

Para llevar a cabo el proyecto, se emplea el mismo software de videojuegos utilizado en el trabajo de partida; esto es, Unity3D (en adelante, Unity). Como lenguaje de programación, se continúa asimismo ampliando el código en C#, un lenguaje orientado a objetos perteneciente a la plataforma .NET y, como entorno de programación, Visual Studio. Cabe destacar que se emplea la versión gratuita sin fines comerciales de Unity3D que proporciona Unity Technologies ApS, junto con la versión también gratuita de Visual Studio que se descarga automáticamente con el *software*.

Figura 1. Logo del motor de videojuegos Unity3D. Fuente: www.unity3d.com.

Por otra parte, como sistema de realidad virtual se emplea Oculus Rift, constituido por las gafas de realidad virtual y los dos controladores manuales. La interfaz Oculus-Unity ya viene proporcionada por el propio software de Unity3D. Por otra parte, tanto el sistema de realidad virtual como el ordenador en el cual se desarrolla el simulador son proporcionados por el Grupo Integrado de Ingeniería de la Universidade da Coruña.

Figura 2. Sistema de realidad virtual Oculus Rift. Fuente: www.amazon.es.

En lo que se refiere a la interfaz gráfica de usuario, se ha partido en algunos casos de recursos de muestra del paquete Oculus Sample Framework desarrollado por Facebook Technologies, LLC a través de la empresa Oculus VR. Estos recursos son accesibles y descargables en su sitio web dedicado a los desarrolladores (https://developer.oculus.com). Junto con ellos, también se ha empleado el recurso gratuito VRKeys disponible en el almacén de recursos de Unity (Unity Asset Store). Todos estos contenidos están sujetos a los términos de uso explicados en dicho sitio web, a los cuales también se ajusta los usados en el presente trabajo, el cual no tiene carácter comercial.

Finalmente, los recursos gráficos y y toda la información relativa al caso de demostración son aportados por la Unidad Mixta de Investigación (UMI) de la Universidade da Coruña – Navantia y no tienen carácter confidencial.

.

Capítulo 2

En este segundo capítulo se establece el marco en el que se sitúa el presente trabajo y las dos tecnologías en las que se basa, como son la simulación de eventos discretos y la realidad virtual. En esta contextualización es requisito indispensable establecer qué se entiende por fabricación inteligente y las nuevas visiones del concepto de fábrica, todo ello dentro de la ya mencionada *Industria 4.0.* Esto nos permitirá situar específicamente el proyecto actual dentro de esta revolución tecnología que vivimos hoy en día.

2.1 Contexto actual

Como ya se mencionó en la 1.1 Introducción, es innegable que en la actualidad nos encontramos en medio de una extraordinaria ola de avance tecnológico que muchos no dudan en denominar «La Cuarta Revolución Industrial», término no exento de controversia por ser para muchos demasiado ambicioso. Con origen en Alemania, concebido por el gobierno central en el año 2011 como plan estratégico a medio-largo plazo, recibe inicialmente el nombre de *Industria 4.0*, término con el que se ha extendido al resto de países del entorno. No obstante, frente a cualquier posible etiqueta, es más que evidente que de 10 años a esta parte estamos viviendo una auténtica transformación tanto en el qué como en el cómo producimos y, además, a un ritmo exponencial, frente a la evolución lineal de las tres grandes revoluciones industriales (Rodic, 2017).

Entre otros modelos descriptivos que pretenden definir este fenómeno, en (Pernas Álvarez, 2017) se adoptada el recogido en (Rüßmann, Lorenz, Gerbert, & Waldner, 2015) con fuente en el grupo consultor BCG, donde se identifican nueve pilares fundacionales entre los que se encontraba la simulación.

Sin embargo, otros como McKinsey & Company en (Wee, 2015) son menos ambiciosos y prefieren hablar de *Industria 4.0* como la siguiente fase en la digitalización del sector productivo, impulsado por cuatro principales fenómenos: el extraordinario incremento en el volumen de datos (*Big Data*), el poder computacional y la conectividad; la aparición del *Business Intelligence* y *Business Analytics*; las nuevas formas de interacción hombre-máquina como la realidad aumentada; y los mejoras en materia de transmisión y traducción de órdenes o instrucciones digitales a la realidad del mundo físico, en referencia a tecnologías como la impresión 3D y la robótica avanzada – lo que podemos definir también como la convergencia entre el mundo digital y el mundo real. En otras palabras, la era modera de la fabricación tiene sus raíces en el último medio siglo, con especial importancia de la expansión de la automatización de procesos, en diferentes niveles según sector industrial (Kusiak, 2018).

Frente a todo, si bien es cierto que algunas de las tecnologías mencionadas aún no están suficiente preparadas para su aplicación a gran escala − mismamente (Wee, 2015) recoge una encuesta en la que, a enero del 2015, entre 300 grandes fabricantes, solamente el 48% se consideraban preparados para la *Industria 4.0* − , la *Industria 4.0* está llamada a resolver algunos de los problemas y desafíos que el mundo enfrenta hoy en día, como el uso eficiente de la energía y los recursos, la producción urbana o el cambio demográfico (Kagermann, Wahlster, & Helbig, 2013). Y de forma más específica, en lo concerniente a la industria[1], este mismo informe cita una serie de áreas en las que se le asume un gran potencial. Destacamos entre ellas tres en particular:

- la incorporación de los requisitos del cliente en el proceso de fabricación, desde la fase inicial de diseño hasta los posibles cambios de última hora: es decir, una completa personalización tanto del producto como del servicio al cliente final;

[1] El informe (Kagermann et al., 2013) hace referencia a la industria alemana, pero sus conclusiones son extrapolables a la industria en general, por el carácter del tema tratado.

- la flexibilidad y agilización de los procesos productivos, con configuraciones dinámicas que permitan hacer frente en tiempo y coste a posibles cambios repentinos de la demanda o problemas en algún punto de la cadena. Todo ello manteniendo lo los niveles de calidad, e incurriendo en los mínimos riesgos posibles;
- y la optimización del proceso de toma de decisiones, tanto en tiempo como en calidad; esto es, tomar correctamente las decisiones correctas, mediante la recopilación y tratamiento adecuado de los datos, la verificación temprana de los efectos de las decisiones tomadas y una apropiada metodología de trabajo.

Precisamente estas componen un resumen de los objetivos de la simulación de la que hablaremos en profundidad más adelante y que, por tanto, adquiere un papel muy relevante en la transformación actual de la industria.

2.2 Fabricación inteligente (*Smart manufacturing)*

Entre los fenómenos mencionados, el incremento exponencial en la cantidad de datos tanto en volumen, velocidad como complejidad (Shao, Shin, & Jain, 2015), el aumento del poder computacional y las nuevas posibilidades de interacción hombre-máquina, facilitan el camino para lo que se empieza a conocer como fábrica virtual (*Virtual Factory*). Sin embargo, son varios los términos que entrecruzan definiciones con este último, como son la fábrica digital (*Digital Factory*) inteligente (*Smart Factory*) o el gemelo digital (*Digital Twin*). La celeridad con la que aparecen en escena nuevos conceptos da lugar a que se produzcan ambigüedades en la nomenclatura técnica. A esto hay que añadir el uso con fines únicamente comerciales que muchas veces se hace de ellos, que añade más confusión al asunto. Por tanto, lo que se pretende aquí es aclarar qué se entiende por cada uno de estos vocablos y cuáles son las diferencias entre ellos.

Para ello, es primordial hablar previamente de un término bastante extendido y que surge a consecuencia de la *Industria 4.0*, como es el de fabricación inteligente o *Smart Manufacturing.* Si bien (Kusiak, 2018) insiste en el hecho de que no existe una deficinión generalmente aceptada, son varios los artículos como (Jain, Lechevalier, Woo, & Shin, 2015) o (IT-Enterprise, 2019) que toman como referencia la adoptada por *The National Institute of Standrards and Technology (NIST)*. Este define *Smart Manufacturing* como “sistemas de fabricación colaborativos y completamente integrados que responden en tiempo real a cambios en las condiciones o requerimientos de la producción, tanto en la totalidad de la cadena productiva como en lo referente a las exigencias y peticiones del cliente”[2]. Es importante remarcar que la expresión *en tiempo real* resulta hoy aún muy ambiciciosa, con lo que normalmente suele interpretear como “en tiempo lo más cercano posible al real”.

Por tanto, la fabricación inteligente tiene su base en los sistemas ciber-físicos (CPS, por sus siglas en inglés), sistemas embebidos que recopilan y transportan datos específicos, y se interconectan entre ellos gracias a *IOT* y la nube, formando a su vez otro sistema ciber-físico más amplio y acercando cada vez más los mundos real y virtual (Rodic, 2017). En lo relativo a la fabricación, se habla concretamente de sistemas ciber-físicos de producción (CPPS, por sus siglas en inglés), lo que comprende máquinas inteligentes, sistemas de almacenamiento y medios de producción capaces de, autónomamente o con la mínima intervención humana posible, intercambiar datos, ejecutar instrucciones y tener cierto control unos sobre otros. Esto supone el paso de una cadena de producción lineal y secuencial hacia un sistema abierto, circular e interconectado denominado por (Deloitte, 2017) como la red de producción y distribución digital (*Digital Supply Network*).

[2]Traducción realizada por el autor de la definición en inglés: *fully-integrated, collaborative manufacturing systems that respond in real time to meet changing demands and conditions in the factory, in the supply network, and in customer needs.*

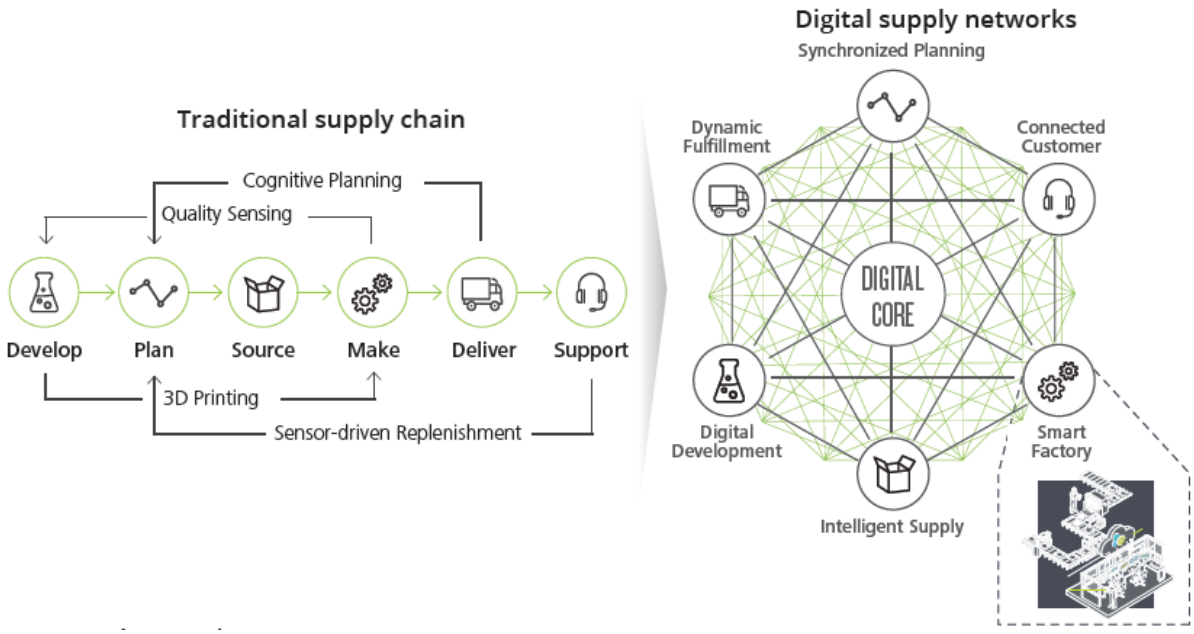

Figura 3. Transformación de la cadena de producción. Fuente: Deloitte analysis.

Esta red circular abre un abanico nuevo de posibilidades con eje central en la información y la interconexión de sistemas. Existe una integración total de toda la cadena de valor, desde el proveedor hasta el cliente final, con realimentación de información entre todas las partes, algo impensable en la cadena de producción tradicional, donde hay elementos aislados entre ellos. En consecuencia, se posibilita la optimización del proceso de forma mucho más ágil de inicio a fin y como un conjunto.

No obstante, también es necesario señalar una serie de capacidades que la industria debe desbloquear para realizar una fabricación inteligente: la integración horizontal de todos los sistemas operacionales y organizativos, tanto en términos de recursos humanos como materiales; la integración vertical a través de todos los sistemas de fabricación; y, finalmente la integración de toda la red de producción (Kagermann et al., 2013). Esto puede traducirse más concretamente en una serie de cambios fundamentales como la consolidación o creación de una arquitectura estándar de referencia, la disposición de medios de transmisión de información adecuados – algunos como (Qi & Tao, 2018) mencionan cantidades cercanas a 40 zettabytes de datos para el año 2020 – y la creación en paralelo de un marco legal adecuado (Kagermann et al., 2013):

Con todo lo anterior, no hay lugar a dudas de que la fabricación inteligente cambiará a su vez el concepto que tenemos de fábrica, que deja de ser ese edificio donde se producen determinados productos, para convertirse en una realidad mucho más amplia que va más allá del mundo real.

2.2.1 Fábrica digital y gemelo digital

Cuando pensamos en lo que es una fábrica, imaginamos un edificio medianamente grande plagado de máquinas, cintas transportadoras, carretillas elevadoras, oficinas y, cada vez en menor cantidad, trabajadores. Sin embargo, en estos últimos años han surgido nuevos conceptos gracias al desarrollo tecnológico que, como se ha remarcado previamente, no tienen una definición generalmente aceptada y resultan ambiguos en la mayoría de los casos.

Si bien el término natural que deriva de fabricación inteligente es el de fábrica inteligente, el primero en aparecer en escena es la “fábrica digital”. Como propone (IT-Enterprise, 2019), esta hace referencia al de desarrollo de modelos de productos a través del diseño digital y herramientas de modelado en todas las etapas de producción; desde la de investigación y

desarrollo hasta la creación del *gemelo digital*. Es decir, *Digital Manufacturing* abarca las tecnologías CAD/CAM/CAE, PDM (*Product Data Management*), PLM (*Product Lifecycle Management*), las máquinas CNC y tecnologías de fabricación aditiva como la impresión 3D (IT-Enterprise, 2019).

Surge aquí el concepto de *gemelo digital*, cada vez más extendido en diversos ámbitos de la industria. Aunque cuenta con múltiples definiciones, es interesante la proporcionada en un principio por (Söderberg, Wärmefjord, Carlson, & Lindkvist, 2017) como «una copia digital de un sistema físico que permite llevar a cabo en tiempo real la optimización de productos y procesos de producción»[3] Frente a otras, se trata de una acepción bastante amplia que pone el foco en la optimización del recurso del cual se hace una copia digital, pero parece no tener en cuenta otros aspectos sumamente importantes como la conectividad o la información. En este sentido, (Rodic, 2017) puntualiza que el *gemelo digital* «altera sus propiedades y su comportamiento a través de los datos y la información». Finalmente (Söderberg et al., 2017) recalca que, con la llegada del *Internet de las cosas*, el *gemelo digital* se ha convertido en «una amplia descripción tanto física como funcional de un componente, producto o sistema, que incluye más o menos toda la información que pudiese resultar útil en la actual y la siguiente fase del ciclo de vida del mismo»[4]

Por tanto, con todo lo anterior, podemos considerar la *fábrica digital* como un primer paso ya dado en la consecución del *gemelo digital* aplicado al conjunto de una fábrica.

2.2.2 Fábrica inteligente

Al haber realizado una descripción bastante completa de lo que significa la fabricación inteligente, podemos ya hacernos una idea de lo que comporta la *fábrica inteligente* o *Smart Factory*. Mismamente (Kagermann et al., 2013) ya lo cita como «un componente clave en las infraestructuras inteligentes del mañana» lo que, como se ha mencionado, transformará la cadena de valor y abrirá nuevas oportunidades de negocio. De la misma manera, (Deloitte, 2017) trata la fábrica inteligente como un gran salto adelante desde la automatización tradicional a un sistema flexible capaz de:

- optimizarse a sí mismo a lo largo de los diferentes subsistemas que lo componen,
- aprender de los nuevos cambios y condiciones en tiempo lo más cercano posible al real,
- y ejecutar de manera autónoma el o los procesos de fabricación que lo integran.

Para ser más concretos, se mencionan a continuación las características fundamentales de la fábrica inteligente (Deloitte, 2017):

- Conectividad. Esta, a su vez, puede ser: a nivel interno desde los sensores y dispositivos de monitorización tanto de materiales como máquinas inteligentes (*IoT)* hasta los sistemas de toma de decisiones mediante inteligencia artificial; o global, donde la fábrica como sistema se puede conectar a una red global conformada por más fábricas inteligentes distintas y, más ampliamente, con la red de producción y distribución digital (véase Figura 3).
- Optimización: tanto en el uso de los recursos y la energía, como en la gestión de la información y la productividad. Una fábrica continuamente optimizada será requisito indispensable para poder competir en el futuro, eliminando los desperdicios (filosofía *Lean*) con el mínimo coste y máxima calidad posibles. En

[3] Textualmente: «*The concept of using a digital copy of the physical system to perform real-time optimization is often referred to as a Digital Twin»* (pág. 2).

[4] Textualmente: «*a comprehensive physical and functional description of a component, product or system, which includes more or less all information which could be useful in the current and subsequent lifecycle phases»* (pág. 2).

este sentido, como se explicará más adelante, **la simulación** se erige como herramienta indispensable en la consecución de este objetivo.

- Transparencia: tanto en la toma de decisiones como el funcionamiento general de la fábrica. También juega un papel importante en este sentido la ciberseguridad, de manera que los datos solamente sean accesibles por el personal autorizado.
- Proactividad: en referencia a la predicción, identificación y solución de problemas antes de que aparezcan. Esto permitirá mejorar la productividad y la calidad del producto fabricado, además de prevenir costes y contratiempos que deriven de una mala y tardía detección.
- Agilidad.Todo lo anterior generará en consecuencia una fábrica ágil, capaz de adaptarse rápidamente a los posibles cambios de la demanda o en la producción con la mínima intervención y coste posibles.

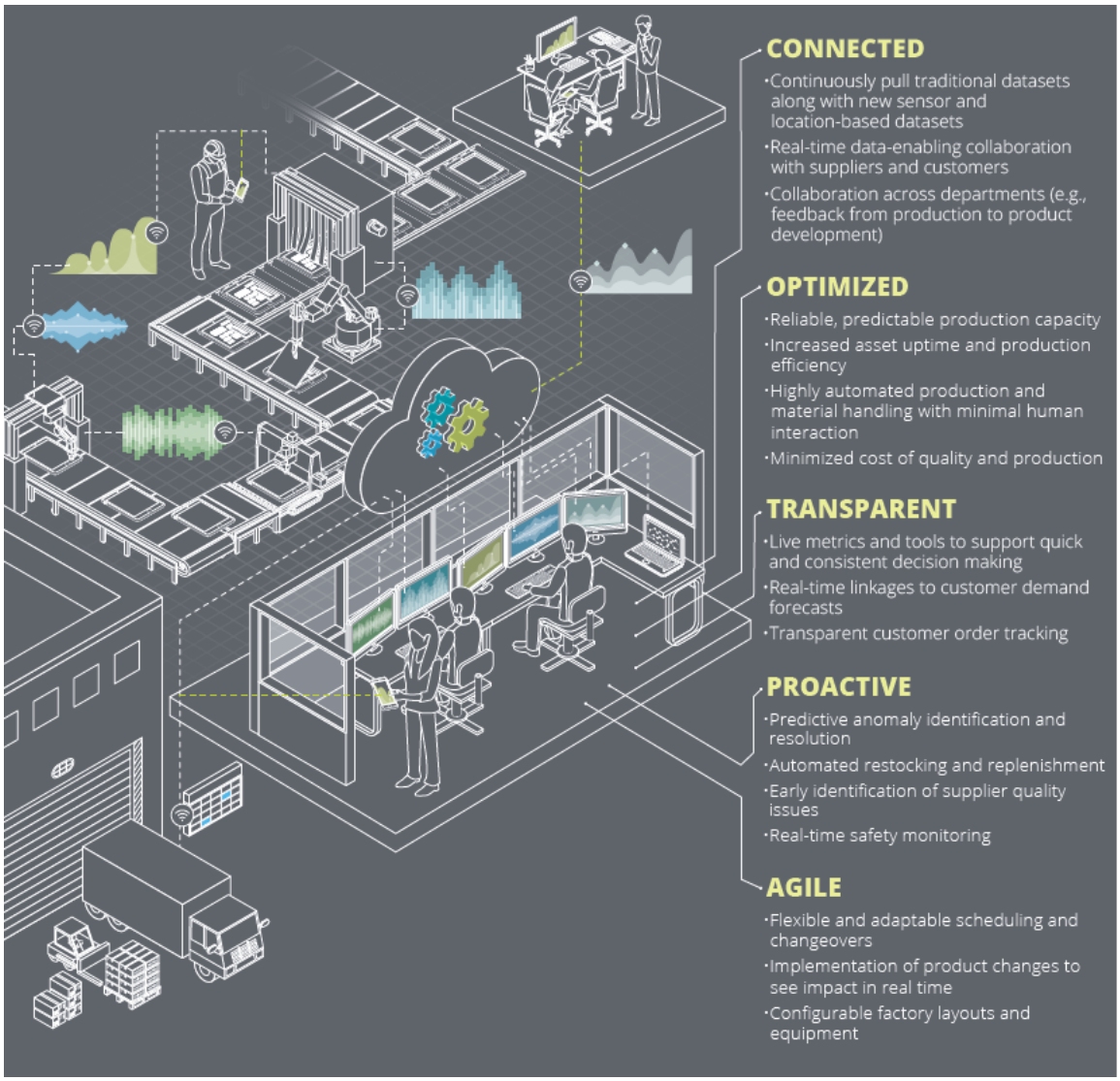

Figura 4. Características fundamentales de la fábrica inteligente según Deloitte. Fuente: Deloitte analysis.

No obstante, esta fábrica conectada, continuamente optimizada, transparente, proactiva y ágil no puede asumirse como el estado o punto a final de la evolución de los sistemas productivos, pues su propia idiosincrasia la lleva a constituirse como un sistema en constante evolución y aprendizaje con capacidad de adaptarse a las nuevas necesidades de la organización en cuestión, como la expansión a nuevos mercados o el desarrollo de nuevos

productos o servicios (Deloitte, 2017). Todo ello frente al modelo tradicional de “llave en mano” que aún sigue muy extendido hoy en día.

2.2.3 Fábrica virtual

Finalmente, ya se empieza a hablar de lo que se conoce como la *fábrica virtual*, un término que si bien surge años atrás (Jain & Shao, 2015), en la literatura se solapa en cierta medida con la fábrica inteligente no quedando muchas veces claro la línea que los separa. La definición más ampliamente aceptada es la aportada por (Jain & Shao, 2015) como «un modelo de simulación integrado por los principales subsistemas de una fábrica que considera la misma como un todo y sirve como apoyo avanzado en la toma de decisiones»[5]. Aparentemente puede resultar una definición semejante a la fábrica inteligente. No obstante, va mucho más allá, pues se trata de un modelo que permite analizar el sistema de producción a cualquier nivel de detalle, desde un paso determinado de fabricación, una línea concreta de producción, la propia fábrica como sistema o incluso las diferentes fábricas vinculadas y conectadas con esta (Jain et al., 2015).

En otras palabras, la fábrica virtual constituye un gemelo digital de nuestra fábrica o red de fábricas al máximo nivel de detalle, algo que resulta hoy por hoy, muy difícilmente ejecutable a gran escala. Si bien (Jain & Shao, 2015) aporta varios ejemplos como el de Ford Motor Company, donde ya empleaban un sistema similar en 2013, con la el avance de la tecnología el concepto de fábrica virtual va redefiniéndose, volviéndose más y más compleja, algo que a la vez forma parte de su idiosincrasia.

En conclusión, podemos asemejar la fábrica virtual a la red de producción y distribución digital (véase Figura 3) de la que nos hablaba Deloitte, entendiéndola como un todo. Aunque su visión difiere de la presentada en este trabajo, (Turner, Hutabarat, Oyekan, & Tiwari, 2016) no duda en situar la simulación de eventos discretos como uno de los pilares de la fábrica virtual y, por tanto, de la fábrica inteligente ya que permite el análisis de la productividad de la fábrica en su conjunto y el de las distintas máquinas o procesos que la componen y, en consecuencia, se erige como herramienta de apoyo básica en la toma de decisiones.

[5] Textualmente: «an integrated simulation model of major subsystems in a factory that considers the factory as a whole an provides an advanced decision support capability».

Capítulo 3

Definido el marco contextual, en el presente capítulo se realiza una revisión de la evolución y el estado del arte tanto del *software* comercial de eventos discretos, así como de los sistemas de realidad virtual y los antecedentes en cuanto a su aplicación en el campo de la simulación. Finalmente, se concreta el alcance real del proyecto y se exponen las nuevas posibilidades que se han pretendido explorar con el trabajo realizado.

3.1 Software de eventos discretos

El origen de la simulación de eventos discretos como tal se remonta a los años 50 o 60 donde, debido a la baja capacidad computacional de la época, su uso quedaba reducido a pequeñas simulaciones muy simples, prácticamente en código máquina, que tenían como objeto determinar si ciertos sistemas cumplían con determinados objetivos operacionales (Xu et al., 2016). No obstante, ya quedaba patente el potencial que tiene esta herramienta en el diseño o validación de sistemas productivos que se traduce en un gran crecimiento a partir de los 90, de la mano de la expansión del ordenador personal y la caída del precio del mismo (Robinson, 2005).

Todo ello sin olvidarnos de que multitud de problemas en cuanto a, por ejemplo, una óptima disposición en planta, son prácticamente irresolubles sin la necesidad de adoptar simplificaciones que a veces resultan difíciles de aceptar. De esta manera, el *software* simulación permite ajustarnos en la medida de lo posible a la realidad – siempre manteniendo el compromiso entre coste, tiempo y esfuerzo –obteniendo resultados con muy altos niveles de confianza. La simulación se sitúa por tanto dentro de lo que (Kusiak, 2018) denomina como «ingeniería predictiva», uno de los pilares de la fabricación inteligente, llamada a restructurar la industria manufacturera.

Sin embargo, como apunta (Xu et al., 2016), todo lo anterior también deja entrever dos exigencias ineludibles que existen desde su nacimiento, pero que se siguen presentando como prerrequisitos en la actualidad:

- Por un lado, la necesidad constante de un mayor poder computacional, debido a que la creciente complejidad de los procesos de fabricación, junto con los cada vez más altos niveles de productividad y eficiencia, exigen modelos de simulación más enredados y fieles a la realidad, todo ello en el marco de la fabricación inteligente. Cabe decir que gran parte del esfuerzo pasa por la optimización del *software* comercial de simulación. Mismamente en (Pernas Álvarez, 2017) se proponían los motores de videojuegos como modelo a seguir, debido al extraordinario avance que ha tenido en estos últimos años, algo, en lo que se ahondará en este trabajo.
- Y por otro, datos cada vez en mayor cantidad y calidad así como herramientas para transmitirlos y manejarlos, lo que constituye de las bases de la *Smart Factory* (Jain & Shao, 2015). Afincados aún en la producción tradicional, muchas empresas manufactureras siguen infravalorando la importancia de los datos de producción relativos a tiempos de operaciones y tareas, con lo que estos son de muy mala calidad y desvirtúan completamente la realidad que se miden. En estos casos, una posible optimización de su proceso productivo queda muy lejos de los resultados esperados además de encarecer el proyecto.

Con la explosión actual en términos de desarrollo computacional, parece ser que la atención hoy en día está puesta en el segundo de los requisitos mencionados. La ausencia de datos de calidad se presenta como uno de los grandes retos de este sector, algo que también deriva en que se siga dudando de su eficacia como técnica de optimización de procesos, si bien con la inclusión del 3D ha ganado muchos más adeptos.

3.1.1 Retos actuales de la simulación

El presente proyecto pretende centrarse en uno de los retos actuales del *software* de simulación como es la incorporación de la realidad virtual y su posible papel en construcción y simulación de modelos. Por tanto, no es objeto del mismo profundizar en la metodología de simulación ni en los objetivos de la misma, los cuales son descritos con detalle en (Pernas Álvarez, 2017) y se dan por asumidos aquí.

Por consiguiente, se explican a continuación los retos principales que afronta hoy en día la simulación, algunos de ellos aportados desde el punto de vista del autor:

- Mejoras en el proceso de desarrollo de modelos. Es interesante la perspectiva aportada por (Robinson, 2005) donde se comentan diversos aspectos, quizás algunos de ellos ya superados en estos últimos años. Entre ellos destaca especialmente la manejabilidad – facilidad por parte del modelador a la hora de desarrollar un modelo −, donde queda camino por recorrer, pues los actuales programas comerciales de ya simulación 3D cuenta con una curva de aprendizaje con mucha pendiente; y, por otra parte, la capacidad de reciclaje o reutilización de modelos, aspecto que ya se exploraba en (Pernas Álvarez, 2017) y que representa una de las funcionalidades más interesantes de los motores de videojuegos.
- Interacción operario-sistema productivo. El modelado del comportamiento de los humanos, así como el alcance de sus acciones sobre el sistema productivo sigue requiriendo más profundización dado la complejidad que atañe, pues se aleja mucho de la linealidad y simplicidad que posee el de un robot en su comportamiento.
- Optimización de procesos con intervención humana. Relacionado con el punto anterior, cabe decir que, tras la optimización – en el *software* de simulación − de algún proceso productivo con intervención humana, existe también una problemática «pos-simulación» asociada a la aplicación de las medidas que se pretenden adoptar, debido a los hábitos, sesgos, tendencias y rutinas que tenemos como humanos. Por tanto, resultaría también interesante investigar hasta qué punto una simulación debe modelar el comportamiento del operario desde el punto de vista de que si esas medidas se harán finalmente realidad.
- Integración con otros tipos de simulación. En (Pernas Álvarez, 2017) se trató de aprovechar el motor físico de Unity3D para crear ciertos movimientos como el de la grúa, una línea en la que se profundiza en este proyecto. También se proponía combinar la simulación de eventos discretos con simulaciones continuas como puede ser la caída de cierto producto por acción de la gravedad. En este sentido, el futuro de la simulación de eventos discretos pasa por aprovechar las funcionalidades de otros tipos de simulaciones, y más teniendo en cuenta que las nuevas generaciones de modeladores que se adentran en este mundo están acostumbrados al nivel de gran realismo de los videojuegos (Robinson, 2005).
- Simulación distribuida. Si bien es algo extendido en el mundo militar, existe aún mucho potencial por explotar en el desarrollo de modelos y la simulación de casos a través de la nube, aprovechando el poder computacional y las ventajas que proporciona una simulación distribuida entre distintos ordenadores en distintas partes del mundo. En este sentido, conviene reivindicar de nuevo la necesidad de una serie de estándares de simulación que hagan esto posible en un futuro, pues la tecnología necesaria está ya alcance de todos.
- Mejora de los algoritmos de optimización. La mayor parte de los *softwares* comerciales emplean algoritmos metaheurísticos comunes que obtienen buenas soluciones a base de réplicas, partiendo de muestras aleatorias que siguen una determinada distribución muestral sin nunca una solución óptima – aunque en multitud de casos esto último no es necesario, pues el propio resultado es de carácter estocástico. En cualquier caso, (Xu et al., 2016) apunta a que los algoritmos actuales de optimización no tienen la eficiencia computacional

requerida ni son adecuados para que, con la explosión actual del *Big Data*, se haga posible toma de decisiones en tiempo real, dentro del contexto de la fábrica inteligente.

- Realidad virtual. Finalmente, en estos últimos años solo unos pocos de los principales programas comerciales como FlexSim o Simio han ido incorporando una de las tecnologías de las que sigue diciéndose revolucionará la industria − (Robinson, 2005) ya citaba FlexSim entre los primeros que incluían esta tecnología, si bien por aquel entonces más que realidad virtual se trataba animaciones 3D desde diversas perspectivas. Dado el objetivo del proyecto, se profundizará sobre este aspecto más adelante, donde se pretende describir hasta qué punto y con qué fines se ha incluido hoy por hoy la realidad virtual en la simulación.

3.1.2 Evolución del software de simulación

En el mercado actual de *software* profesional de simulación, la mayor parte de los programas ya cuentan con, al menos, la visualización 3D del modelo (Swain, 2015), donde en el propio modelado obtenemos una versión en tres dimensiones del proceso. Sin embargo, en el presente trabajo se parte de la posibilidad de construir el modelo directamente en 3 dimensiones, funcionalidad que poseen los considerados como los mejores *software* de simulación del mercado actual (Descreye Solutions, 2018). Entre ellos situamos principalmente AnyLogic, FlexSim, Arena o Simio, algunos mencionados ya previamente.

Como breve introducción histórica, la idea de lo que se denomina como simulación visual interactiva (VIS, por sus siglas en inglés) surge ya en los años 70 (Robinson, 2005), cuando aún no existía poder computacional para realmente llevarla a cabo. Este concepto evoluciona y se hace realidad al final de los 80, donde se empiezan a comercializar los primeros paquetes de simulación interactiva aplicada la fabricación (VIMS, por sus siglas en inglés) como QUEST, Arena o mismamente FlexSim. Algunos de ellos evolucionan rápidamente hacia el desarrollo de modelos en el entorno 3D, lo que también tiene un efecto visible en sus licencias de uso, superando los 10.000€, una cantidad no asumible por muchas empresas. Esto mismo señala (Robinson, 2005) el cual afirma que se hubiese acabado por convertir en un obstáculo para la expansión de la simulación sino llegan a aparecer opciones alternativas y mucho más accesibles en términos económicos – con licencias alrededor de los 1000€ – como Siml8 o Extend, paquetes VIMS más simples donde la creación del modelo se realiza en 2D – (Robinson, 2005) los denomina como «VIMS *low cost*» –, mientras que la visualización 3D se genera automáticamente, y queda reducida a una mera representación. Esto también tiene un efecto en el resto de VIMS de alta calidad, suavizando los precios de esos paquetes de simulación.

A pesar de seguir teniendo cierto éxito hoy en día, debido en muchos casos a su solidez y su simplicidad que facilita la construcción de determinados modelos, a partir de los 2000 la revolución computacional y gráfica empieza a cambiar el arquetipo de *software* de simulación Esto se traduce en la exigencia de mejores prestaciones y funcionalidades tanto gráficas como en lo relativo a la interfaz de usuario. Además, se empieza prestar más atención a otro tipo de perfil de modelador: un usuario principiante, generalmente con pocos conocimientos de programación, que quiere iniciarse en el mundo de la simulación y que, como se ha mencionado previamente, cada vez está más acostumbrado a los altos niveles de calidad visual que se encuentran en otros sectores como el de los videojuegos. Por lo tanto, es en esta época cuando el 3D empieza a adquirir cada vez mayor importancia y comienza a hablar de lo que denominamos ya en (Pernas Álvarez, 2017) como simulación 3D.

La simulación 3D hace referencia a la posibilidad de construir el modelo directamente en 3 dimensiones de manera que, al final del proceso, se obtiene un gemelo digital similar tanto en su aspecto como en su comportamiento a la planta, fábrica, establecimiento o, en general, caso de simulación. Al simular este modelo, podemos ver cuál es el comportamiento de nuestro gemelo «en tiempo real», el cual obedece en primera instancia al motor de eventos

discretos que ejecuta los pasos de tiempo de la simulación, pero que se sincroniza a su vez con un motor de tiempo continuo o casi-continuo que realiza la visualización 3D de los eventos ejecutados, si hubiere. No obstante, no es objeto del presente trabajo analizar tanto el funcionamiento como las ventajas o inconvenientes de este tipo de simulación, que ya habían sido exploradas en (Pernas Álvarez, 2017) y de las que hay multitud de análisis en la literatura. Aquí se pretende dar un paso más allá, como es la introducción de un tipo de interacción distinta con el modelo, a través de la realidad virtual.

3.2 Realidad Virtual

En origen la realidad virtual (también denominada VR, por sus siglas en inglés), entre la multitud de definiciones más o menos técnicas que existen en la literatura, surge como «una interfaz hombre-máquina que permite al usuario sumergirse en una simulación gráfica 3D generada por ordenador y navegar e interactuar en el en tiempo real, desde una perspectiva centrada en el usuario» − esta definición ya había sido aportada en (Pernas Álvarez, 2017), lo cual, en primera instancia, es un concepto bastante amplio. De hecho, (Mujber, Szecsi, & Hashmi, 2004) aporta 3 sistemas distintos en función del grado de inmersión de una tecnología que nace ya hace 30 años:

1. Sistemas VR no inmersivos: muestran el mundo virtual en una pantalla o monitor en alta resolución, e interactúan con él a través de periféricos como ratones, teclados o joysticks. No generan sensación de inmersión y el nivel de interacción es bajo.
2. Sistemas VR semi-inmersivos: un término medio donde sistemas de proyección o monitores de gran tamaño, incluso con sistemas múltiples, y donde los periféricos de entrada siguen siendo los mismos que los del nivel anterior. El mayor tamaño como complejidad del monitor aumenta la sensación de inmersión hasta un nivel medio-bajo.
3. Sistemas de Inmersión de VR: una serie de periféricos de entrada y salida que el usuario se coloca previamente crean en él la ilusión de introducirse dentro de la pantalla de los dos niveles anteriores, dentro un mundo virtual donde puede moverse e interactuar con los elementos que existen dentro de él. Se incluyen en este nivel los sistemas CAVE y HMD que se explicarán más adelante.

No hay duda de que son realmente los últimos los que se identifican en la actualidad con lo que llamamos realidad virtual, mientras que los dos primeros se suelen denominar más ben como animación 3D. No obstante, se ha decidido adoptar esta caracterización pues en algunas publicaciones antiguas existen multitud de referencias a los dos primeros como VR, como por ejemplo en (Justice I. Akpan & Brooks, 2005), en (J.I. Akpan & Brooks, 2005) o en (Rekapalli & Martinez, 2010). Como en (Pernas Álvarez, 2017) ya se profundizaba en la diferencia entre ambos tipos de interfaz, nos centraremos aquí directamente en los sistemas de inmersión de VR.

En consecuencia, según los periféricos de entrada y salida, podemos situar fundamentalmente dos tipos de sistemas:

- *Head-mounted display* (HMD) con controladores: este sistema, ilustrado en la Figura 2, es el pack básico comercializado hoy por los principales fabricantes, cuyo mercado más lucrativo es el de los videojuegos. Podemos definirlos como sistemas dinámicos, pues son fácilmente transportables lo cual, a su vez, ha posibilitado su comercialización al pequeño consumidor. Por otra parte, cada vez surgen nuevos añadidos como los guantes VR o hápticos, o HMD multisensoriales con tecnología olfativa que emiten pequeños aromas, de manera que la experiencia sea lo más inmersiva posible. No obstante, en este trabajo, dado el objeto del proyecto, se emplea lo que podemos denominar como el kit básico de realidad virtual.

Figuras 5 y 6. Máscara HMD multisensorial como emisor de olores (izquierda) y guantes hápticos de realidad virtual. Fuentes (respectivamente): www.uploadvr.com y www.feelreal.com.

- CAVE (*Computer Automatic Virtual Environment*): es posiblemente el sistema inmersivo de VR más efectivo entre todas las interfaces de visualización. Creado por investigadores de la Universidad de Illinois en 1996, consiste en un sistema de pantallas en forma cúbica, donde el usuario se sitúa en el interior del cubo. Existen CAVEs desde 3 hasta 6 pantallas, consiguiendo cada vez un mayor grado de inmersión. No obstante, se trata de un sistema que es utilizado especialmente con fines de investigación, o en eventos culturales o de entretenimiento abiertos al público en general, concretamente por dos motivos: su precio, pues es un sistema complejo y caro, que requiere de bastante espacio físico; y su carácter estático, aunque ya existen empresas que lo comercializan asegurando tiempos de montaje de pocas horas (Pérez Martínez, 2011).

Figura 7. Sistema tipo CAVE. Fuente: www.vrfitnessinsider.com

3.2.1 La realidad virtual en la industria

Aunque ya desde su nacimiento se empezó a investigar las posibles aplicaciones de la realidad virtual en distintos ámbitos, en numerosas ocasiones la experimentación no era todavía viable debido a la falta de medios, especialmente en lo referido a poder computacional (Justice I. Akpan & Brooks, 2005). Esta fuerte apuesta por la realidad virtual sigue hoy en día, pues es con la *Industria 4.0* cuando la tecnología alcanza un nivel de madurez suficiente como para expandir su uso a gran escala.

De hecho, un sector donde desde hace tiempo ha supuesto una mejora trascendental es el de la medicina y, por tanto, a su vez en la salud de los pacientes. Algunos de los ejemplos que pone (Pérez Martínez, 2011) son las autopsias virtuales, las interfaces neuronales para el movimiento de brazos, el tratamiento de la paraplejia y el autismo o el tratamiento de fobias. También se emplea la realidad virtual para practicar o mejorar la técnica de los cirujanos en una determinada operación quirúrgica sin necesidad de llevarla cabo, lo que supone una reducción importante de los riesgos asociados.

Otro sector pionero en el uso de esta tecnología es el militar, donde la realidad virtual sirve para formar y entrenar a soldados en operaciones tácticas de tierra mar y aire (Pérez Martínez, 2011). Todo ello a través de kits de *software* que permiten la inmersión en escenarios virtuales escalables similares a los que podrían existir en la realidad. Junto con ello, los simuladores de vuelo o de conducción de vehículos militares, donde las sensaciones que tiene el usuario son prácticamente las mismas que tendría en la realidad, permiten practicar determinadas técnicas sin necesidad de poner en riesgo su vida. Precisamente simuladores parecidos tienen cada vez más popularidad en el mundo del entretenimiento, pues desde hace tiempo se pueden ver en salas recreativas.

Existen también otro tipo de aplicaciones más culturales como recorridos virtuales por museos o ciudades – Google Earth – o la creación de escenarios virtuales en televisión. No obstante, en lo que se refiere a la industria manufacturera, podemos seleccionar una serie de áreas donde se viene utilizando la realidad virtual desde hace unos años, pero que sigue mejorando y evolucionando:

- Diseño. Con la posibilidad de explorar distintas dimensiones y aspectos o acceder a librerías de formas o plantillas, la realidad virtual se traduce en un salto muy importante en el diseño de producto en términos de visualización e interacción (Nee & Ong, 2013). El usuario realiza ahora los bosquejos directamente en 3 dimensiones, evaluando diseños alternativos y realizando un prototipado virtual. La tecnología VR supone un avance a la hora realizar análisis técnicos tanto de fabricación, mantenimiento o comercialización, como también para evaluar la recepción por parte del cliente de forma más fiable (Mujber et al., 2004).
- Gestión de operaciones. En este campo podemos diferenciar a su vez tres áreas como son las de planificación, simulación y entrenamiento (Mujber et al., 2004). En el caso de la simulación de procesos – a través de eventos discretos – son precisamente las ventajas le proporciona la realidad virtual las que trataremos de discutir a lo largo del presente trabajo. No obstante, podemos apuntar algunas aplicaciones en el campo de la gestión de operaciones en general como: la obtención diseño óptimo del *layout* del proceso de fabricación; una mejora en las etapas de validación y verificación de un proceso existente o un modelo; soporte en la interpretación los resultados obtenidos; acceso virtual instantáneo por parte de cualquier empleado a determinado punto o parte de la fábrica; o el entrenamiento virtual para, de la misma manera que en otros ámbitos mencionados previamente, aprender, mejorar o perfeccionar la realización de una determinada tarea sin coste o riesgo alguno asociado.
- Procesos de fabricación: mecanizado, ensamblado e inspección. Entre las aplicaciones más destacadas en este campo podemos mencionar: soporte en el estudio de factores que afectan a la calidad, tiempo y coste del mecanizado;

reducción de tiempos de ciclo y reprocesados gracias a los prototipos virtuales y la comparación con el producto real; verificación de los procesados de ensamblado; o simular el proceso de inspección tanto de los aspectos físicos como mecánicos (Mujber et al., 2004).

Todo lo anterior contribuye a la interconexión entre la fábrica real y física con la fábrica virtual, que acabará siendo el *gemelo digital* y la vez el cerebro de la primera. Sin embargo, cada vez juega un papel más importante la conocida como realidad aumentada, donde la literatura progresivamente nos aporta más y más ejemplos de aplicación de la misma, afirmando incluso que su potencial a corto plazo es incluso mayor que el de la virtual (Nee & Ong, 2013). Desde el punto de vista del autor, a la velocidad de avance de la tecnología, en el futuro se acabará hablando únicamente de realidad mixta, confluyendo por tanto ambas tecnologías en el mismo producto final.

3.2.2 Realidad virtual en simulación

En lo que se refiere al mundo de la simulación, atendiendo a la encuesta realizada por (AnyLogic, 2017), no muchos de los principales desarrolladores de *software* han incorporado soporte para realidad virtual en sus productos. Algunos como FlexSim (AnyLogic, 2017) o Simio (Simio LLC, 2016) han sido los primeros en sumergirse en este mundo; pero, en contraposición a la tecnología VR en sí misma, no existe un gran desarrollo por el momento debido principalmente a dos factores: el precio de los dispositivos necesarios para implementarla sigue siendo alto, lo que provoca un bajo retorno sobre la inversión, aunque a esperas de que disminuya considerablemente en los próximos años con su difusión en los sectores de entretenimiento; y las dudas sobre su utilidad y acerca del beneficio que puede aportar en la optimización de procesos mediante simulación. En este contexto, este proyecto intentará explorar otras posibles vías de aplicación hasta ahora no investigadas, aportando su grano de arena con respecto a este segundo aspecto.

Pese a lo anterior, existen multitud de ejemplos en la literatura del uso de la realidad virtual en la simulación de eventos discretos. Tras una revisión del estado del arte, se han seleccionado algunos de los artículos más interesantes respecto al objeto de este proyecto.

En primer lugar, (Turner et al., 2016) enfoca en su caso la simulación asistida por realidad virtual (lo que denominan como «VR DES») como «vehículo de decisión para análisis complejos de datos a través de modelos de simulación interactivos», dentro del marco de la fabricación inteligente que ha potenciado una ola de avances y cambios en el mundo manufacturero. De hecho, sitúa la simulación como elemento clave tanto en la fábrica inteligente como en la virtual, aunque su visión con respecto a estas difiera con la adoptada en este trabajo. En su propia revisión bibliográfica, apuntan una serie de conclusiones de otros artículos entre las que destacan el uso de la realidad virtual en simulación:

- como un medio para facilitar y posibilitar la exploración de modelos de simulación, de manera que sirva de apoyo clave para la toma de decisiones a nivel directivo o corporativo, por un gabinete de directivos cuyos conocimientos en simulación son más bien limitados. Esta apreciación ya se mencionaba en (Pernas Álvarez, 2017);
- como una herramienta que mejora el proceso de identificación y depuración de errores y omisiones en un modelo de simulación como, por ejemplo, los cuellos de botella;
- como modo de involucrar al cliente en las fases de diseño, prototipado y producción, lo que resulta muy beneficioso al aumentar su confianza en el proyecto, cimentar relaciones y reducir los retrabajos;
- y para la exploración de modelos y toma de decisiones en tiempo real por usuarios que están geográficamente separados, lo cual resulta especialmente interesante. Sin embargo, esto lidia a su vez con algunos de los retos actuales de la simulación (véase 3.1.1 Retos actuales de la simulación), como es la simulación distribuida

a través de la nube y la necesidad de unos estándares de referencia de comunicación DES-VR, apuntado mismamente por (Turner et al., 2016).

(Turner et al., 2016) realiza una revisión bibliográfica muy exhaustiva de los diferentes aspectos de la «VR DES» que existen en la literatura, desde la creación de los modelos virtuales de los diversos elementos, hasta los distintos sistemas de *tracking*. No obstante, resulta especialmente atractiva la idea de que el usuario interactúe en tiempo real con la simulación de un modelo.

En esta línea ya habían investigado (Kelsick, Vance, Buhr, & Moller, 2003) a través del desarrollo de un simulador de ingeniería virtual o VE (por sus siglas en inglés de *Virtual Engineering*») denominado VRFactory. Este superpone una interfaz VR para CAVE sobre un motor de eventos discretos comercial. A través del uso de guantes hápticos, el usuario es capaz de elegir el modo de simulación predefinido que desea, pasando a ser posteriormente un mero espectador dentro del mundo virtual.

Por otra parte, algunas de las conclusiones anteriores eran también extraídas por (Abidi et al., 2015) en su trabajo, donde realiza dos casos de simulación basados en la misma arquitectura. De forma simplificada, se basa en sincronizar un entorno 3D virtual con el simulador SIMAN de Arena a través del *framework opensource* MASCARET, que además hace las veces de interfaz VR con el usuario. Es en el segundo de los casos donde propone la interacción entre este último y el mundo virtual, de manera que después de cada interacción la simulación con Arena cambia en consecuencia a la acción realizada – si bien no especifica exactamente qué tipo de acciones contemplan. Finalmente, también afirma que su objetivo conseguir que las diferentes partes de un proyecto se involucren en el análisis y mejora de los procesos industriales facilitando la toma de decisiones.

Finalmente, (Dorozhkin, Vance, Rehn, & Lemessi, 2012) va un poco más allá con un caso en que el usuario puede pausar la simulación en cualquier punto y cambiar ciertos parámetros como el tiempo medio de ciclo de una máquina. Posteriormente, este puede decidir si desea que el efecto de esta acción sea inmediato en el modelo, o bien que el parámetro cambie una vez la máquina finalice el trabajo previamente en cola. Si bien su estudio se centra también en otros aspectos como la sincronización del motor de eventos discretos con la visualización 3D, supone uno de los avances más notorios encontrados en la literatura. De hecho, hoy en día los *softwares* comerciales requieren reiniciar completamente la simulación para que el cambio de un parámetro de la simulación tenga efecto. Además, aquellos que han incorporado la realidad virtual no permiten modelar en VR, sino que la interacción se habilita una vez simulación ya está corriendo.

Aunque existen más ejemplos en la literatura, estos se han considerados los más cercanos al objetivo del presente trabajo. En este sentido, la meta propuesta aquí da un paso más, configurando un simulador de realidad virtual que permita llevar a cabo la simulación de un proceso prácticamente desde el inicio. Lo único que no se ha contemplado es la generación del entorno 3D – sea una fábrica, un astillero, o una zona de una planta, por ejemplo – donde se desarrolla el proceso pues, como se verá en el caso de demostración, este se introducirá en el motor de videojuegos antes de generar la aplicación. Por lo demás, el resto de recursos se han generado atendiendo a cómo funcionan los programas de simulación en la actualidad, pero esta vez será el modelador dentro del propio mundo virtual el que cree y configure la simulación.

Capítulo 4

En el presente capítulo se describe todo el trabajo realizado en el proyecto, comenzando por la ampliación del código de partida. Tanto en esta fase como durante el posterior desarrollo de la interfaz, se han ido completando distintos casos de prueba mediante los cuales se ha ido depurando y verificando el trabajo ejecutado hasta el momento. Estos serán explicados a lo largo del capítulo, ilustrando así las diferentes etapas de desarrollo del proyecto. Finalmente, se describirá brevemente el caso de demostración seleccionado, así como el resultado final conseguido.

4.1 Trabajo de partida

En primer lugar, antes de abordar el trabajo realizado, es necesario explicar el punto de partida. Ya mencionado en «1.2 Objetivo del proyecto», se parte del código desarrollado en C# en el proyecto fin de grado (Pernas Álvarez, 2017), donde se había modelado el comportamiento de una serie de elementos básicos de cualquier modelo de simulación, a saber: fuente, estación de trabajo, estación de ensamblaje, y sumidero Todo ello sobre la base de unas librerías en JAVA proporcionadas por Grupo Integrado de Ingeniería de la Universidade da Coruña, que posteriormente son traducidas a C#, depuradas, y ampliadas.

Aunque no es propósito del presente proyecto entrar en un mayor nivel de detalle – para ello, véase (Pernas Álvarez, 2017, pp. 35-42) −, para explicar el código desarrollado, en (Pernas Álvarez, 2017) se habla de dos niveles de abstracción: el nivel base o lógico – «SimLink», «SimElements» y «SimClock» en la Figura 8 − que corresponde al funcionamiento del motor de eventos discretos; y el nivel virtual o de Unity − «SimSElements» en la Figura 8 −, que se superpone al anterior, ejerce de interfaz entre este y el motor de Unity3D − «MonoBehaviour» en la Figura 8 −, y cuyo comportamiento obedece asimismo al nivel lógico.

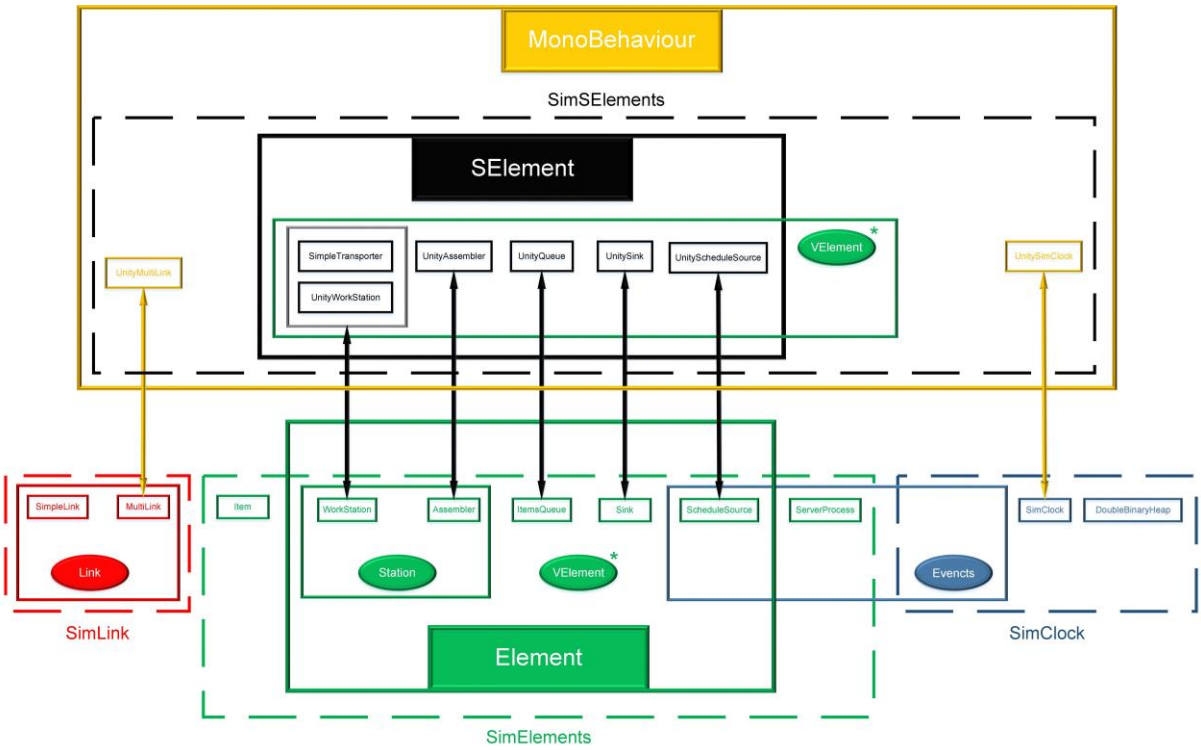

Figura 8. Diagrama explicativo antiguo de la interacción nivel lógico – nivel virtual – Unity3D del proyecto de partida (Pernas Álvarez, 2017). Fuente: (Pernas Álvarez, 2017, p. 37).

En este sentido, en este proyecto se modifica y amplía tanto a nivel lógico como virtual. A nivel lógico, podemos hablar de tres modificaciones distintas:

- En primer lugar, se crean dos elementos que suponen un cambio importante en la programación que se había desarrollado hasta el momento: el operador, y la carretilla elevadora. Como se explicará posteriormente, en estos dos elementos es la clase virtual la que en ciertas ocasiones gobierna la clase lógica, algo que no se había contemplado en un principio, pero que se termina implementando por necesidad. No obstante, este cambio no supone una modificación en el comportamiento normal del motor de eventos discretos.
- En segundo lugar, se mejoran algunas clases entre las que destaca el ensamblador, de manera que sea capaz de ensamblar productos distintos, una capacidad con la que no contaba previamente. Si bien más adelante se mencionarán otras clases, en líneas generales estas suponen modificaciones parciales de otras ya desarrolladas en (Pernas Álvarez, 2017), para que cumplan una determinada función. Por tanto, no se contempla aquí su descripción en profundidad. La razón es que muchas de ellas no tienen relevancia en la consecución del mismo, ya que su creación fue realizada específicamente para algún caso intermedio de demostración.
- Finalmente, todo lo anterior da lugar a una *reestructuración* parcial del código que se vuelve ahora más complejo con la creación de estas nuevas clases (véase Figura 9).

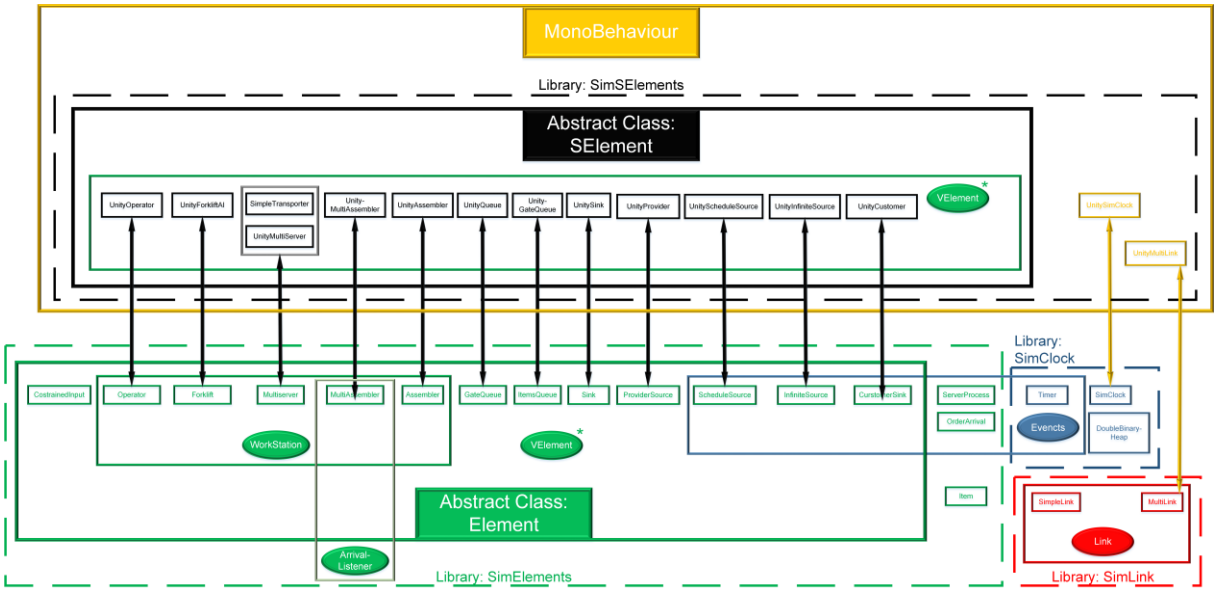

Figura 9. Diagrama explicativo modificado de la interacción nivel lógico – nivel virtual – Unity3D con las clases nuevas y los distintos cambios realizados en el código. Se han incluido las clases principales con los que se han desarrollado los distintos casos, si bien no se incluyen en el diagrama todas las librerías existentes en el proyecto. Fuente: Elaboración propia.

Como también es observable en la Figura 9, a nivel virtual se desarrollan las clases que actúan de interfaz entre aquellas nuevas de nivel lógico y el motor de Unity, y que derivan asimismo de «MonoBehaviour». No obstante, en el caso del operador y la carretilla, también se integran respectivamente animaciones 3D y un algoritmo de enrutamiento, algo que no se había contemplado en (Pernas Álvarez, 2017).

Finalmente, en lo que se refiere a la interfaz, es necesario realizarla de nuevo por completo, incluyendo la capacidad de modelar y configurar los elementos directamente en el mundo virtual. En este sentido, no se recicla el trabajo desarrollado en el proyecto previo, pues tampoco la interacción es la misma, al basarse ahora en los periféricos de realidad virtual.

4.2 Desarrollo del código

Tal y como se ha mencionado en el punto anterior, la primera fase antes de programar nuevos elementos es la de depuración y mejora del código desarrollado en (Pernas Álvarez, 2017). Es importante tener en cuenta las dos ideas básicas sobre los cuales se ejecuta esta parte del trabajo:

- Por un lado, la programación se enfoca siempre desde el punto de vista del reciclaje y la reutilización del código, generando este de la forma más genérica y versátil posible; esto es, que el comportamiento lógico que modela un elemento permita un cambio en el modelo virtual que lo representa, sin mucha necesidad de reprogramación; o que la creación de un nuevo elemento pueda realizarse ágilmente a partir de un código de un elemento similar ya programado previamente.
- Por otro, siempre manteniendo la máxima anterior, se intenta seguir el modelo de funcionamiento de los distintos elementos de una simulación que se sigue en los *softwares* comerciales, teniendo en este caso como referencia el *software* FlexSim de eventos discretos.

4.2.1 Depuración y mejora del código de partida

Con esta base, en lo que se refiere a la depuración del código, se simplifica eliminando variables innecesarias en algunas clases, y realizando una simplificación en algunos aspectos. Aunque se parte de un código ya de por sí genérico, en su revisión se localizan algunas variables innecesarias – que se habían creado especialmente para el caso de demostración anterior − o métodos demasiado extensos, que se dividen para hacer más claro el código. Tal es el caso de los métodos ahora separados de «UnitySimClock» que se encargan de configurar el escenario al inicio, y de resetearlo cuando es necesario. También se incluye una clase nueva que permite actualizar el tiempo cada décima de segundo, para mostrar un tiempo más continuo al usuario.

Respecto a la fase de mejora, a el nivel lógico, el único cambio significativo se produce en la antigua clase «Workstation», que permite ahora procesar el número de ítems que se desee. Por esto mismo se rebautiza como «MultiServer», y «Workstation» pasa a ser una interfaz implementada por todos los procesadores − «MultiServer», «Assembler» y el nuevo «MultiAssembler». Esta interfaz ya existía en el código de partida como «Station».

No obstante, esta fase se centra especialmente en el nivel virtual, con modificaciones en la respuesta a nivel visual de un evento ejecutado. Entre otros, por ejemplo, se modifica la manera en que se configura la disposición de los ítems una vez son recibidos por la cola «UnityQueue» − Figura 10 − de manera que, si se cambian las dimensiones de la cola, no es necesario volver a modificar el código.

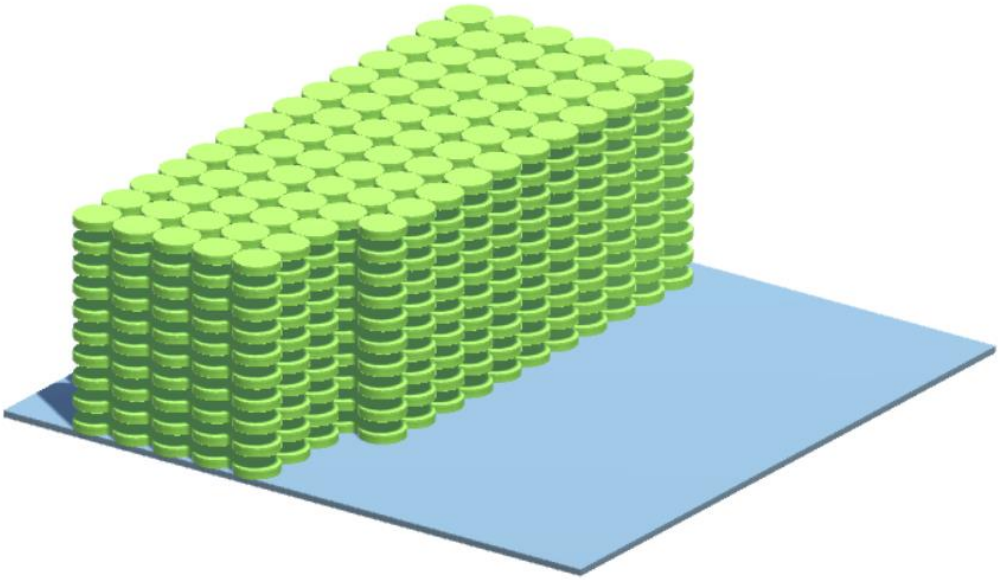

Figura 10. Detalle de colocación de los ítems sobre el modelo de la cola acorde a las dimensiones de estas. Fuente: Elaboración propia

También se realizan las primeras pruebas con un modelo 3D de procesador −«UnityWorkStation» −, frente a las estaciones de trabajo estáticas del astillero que habían sido contempladas en el trabajo anterior. Ahora, se modela un procesador similar al que encontramos en el programa comercial, donde el ítem avanza desde un punto inicial a un punto final según una velocidad específica obtenida en el momento en que el ítem es recibido. Esta velocidad se calcula según el tiempo de evento que ejecuta el procesamiento del ítem, y la distancia que separa ambos puntos de entrada y salida − Figura 11. Cabe decir que el mismo modelo se seguirá en la cinta transportadora, aunque esta será explicada más adelante al constituir una clase nueva.

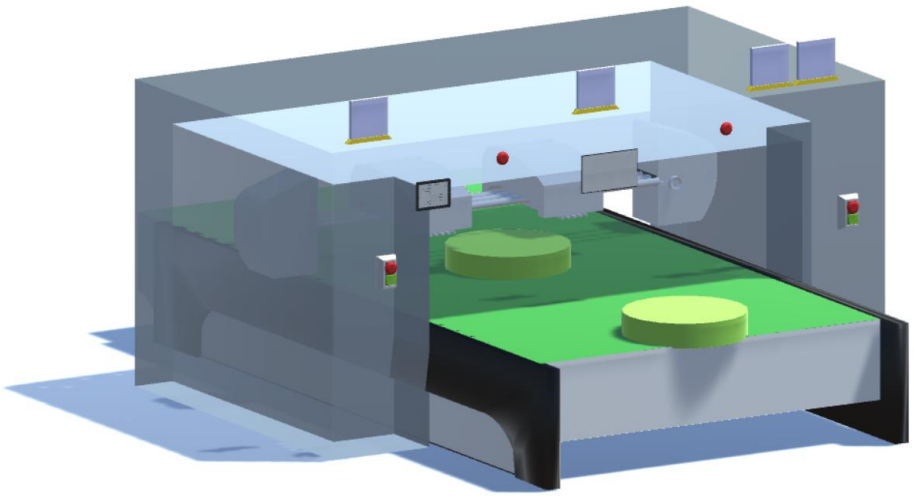

Figura 11. Detalle de avance de los ítems en la estación de trabajo. Fuente: Elaboración propia.

Otra mejora reside en lo relativo al comportamiento lógico de algunos elementos como la fuente programada− «ScheduleSource» − donde los ítems llegaban según una lista predefinida por el usuario. Ahora también se incluye la posibilidad de que los ítems lleguen según una distribución exponencial, definiendo el usuario la media de dicha distribución. Lo

interesante de esta mejora reside en la manera en que se configura, pues el usuario puede elegir directamente en Unity el tipo de llegada a través de una casilla – véase Figura 12. Es decir, se explora la posibilidad de configurar la propia interfaz de Unity3D, algo que evidentemente también permite el propio programa, y que no había sido contemplado en el proyecto anterior.

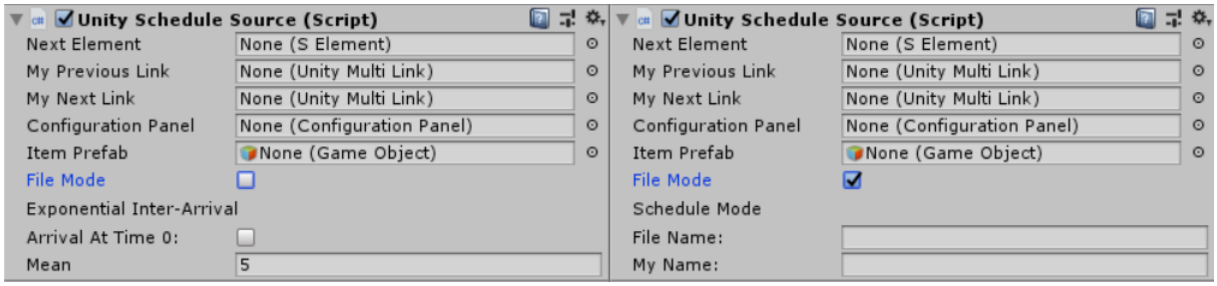

Figura 12. Detalle de panel de configuración de la clase «UnityScheduleSource», con posibilidad de selección del modo de llegada. Fuente: Elaboración propia.

Finalmente, todo lo anterior se acompaña de una renovación de los diferentes modelos 3D de los distintos elementos, de manera que la visualización gane en credibilidad. Estos modelos se mostrarán a continuación en la explicación de los distintos proyectos realizados.

4.2.2 Ampliación del código de partida

Tras la primera fase de mejora, se aborda una segunda parte de ampliación del código para poder modelar otros elementos. El objetivo es explorar así otras funcionalidades que nos aportan los motores de videojuegos, como es la creación manual de animaciones 3D. Parte de estos desarrollos corresponden a pequeños proyectos paralelos que se realizan durante la realización de este trabajo, con la idea de realizar «juegos serios» con fines educativos, en colaboración con el Grupo Integrado de Ingeniería de la Universidade da Coruña. Por tanto, emplearemos estos proyectos para realizar la descripción de esta fase.

4.2.2.1 Taller Virtual

El primero de ellos es el «Taller Virtual», donde la idea es que el usuario en primera persona se sumerja en el entorno de una fábrica y tenga control sobre cuánto pedir, cuándo pedir, cuándo emitir una orden de fabricación, cuánto fabricar y cuándo satisfacer la demanda. La idea es que a través de la toma de tiempos de la llegada de ítems y pedidos y de procesamiento de las distintas máquinas, el usuario pueda ajustar diversas distribuciones estadísticas estos tiempos con un determinado nivel de confianza. Una vez obtenidas estas distribuciones, estará en disposición de modelar el proceso con un *software* comercial y, a través de simulación, obtener una estrategia de pedidos y emisión de órdenes de fabricación que reduzca los costes y genere los mayores beneficios posibles en el tiempo de juego – ambos indicados al usuario. Es decir, nuestro taller representa la fábrica real que es necesario optimizar – véase Figura 13.

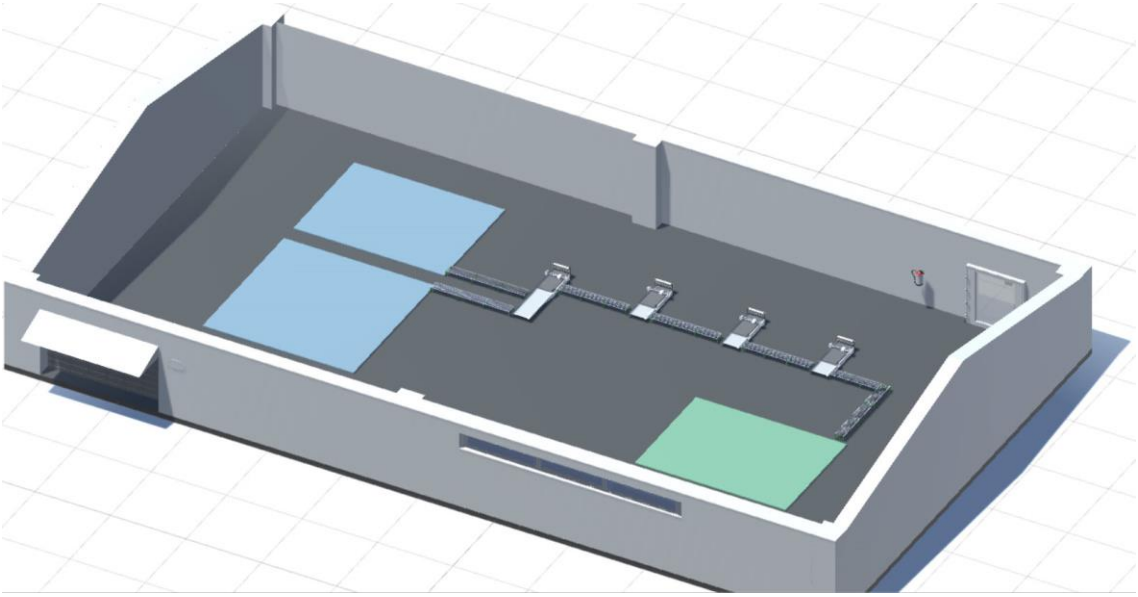

Figura 13. Vista general del modelo completo del «Taller Virtual». Fuente: Elaboración propia.

En este sentido, la primera de las tareas es la de plantear el diagrama de flujo que seguirá el proceso. Este definirá a su vez los elementos cuyo comportamiento será necesario programar. Con el objetivo de no complicar demasiado el caso, se opta por el modelo representado en la siguiente imagen:

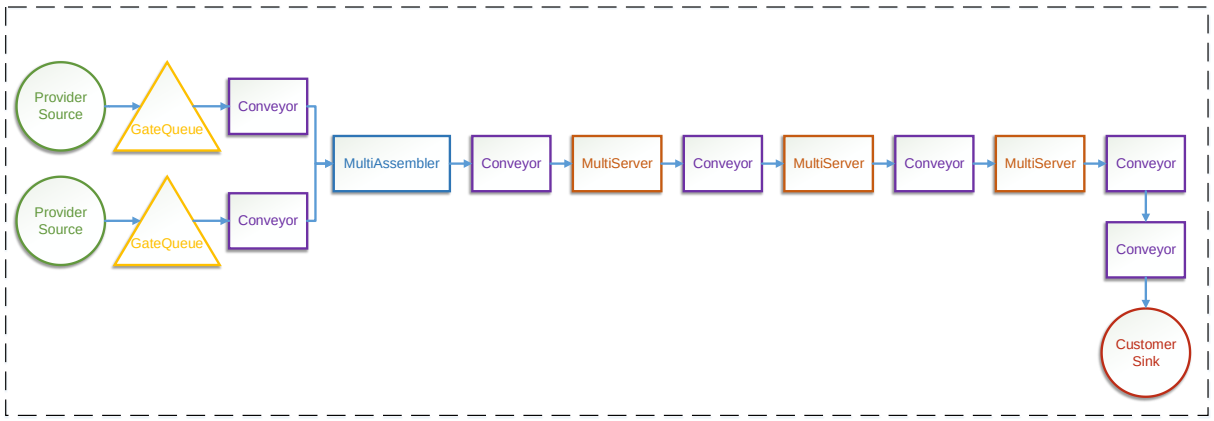

Figura 14. Diagrama de flujo del proceso del proyecto «Taller Virtual». Fuente: Elaboración propia.

Si bien no es relevante para el caso de demostración final del presente trabajo, el primero de esos elementos nuevos respecto a (Pernas Álvarez, 2017) es el tipo de fuente que se denomina ahora «ProviderSource»[6], que no cuenta con un modelo 3D representativo[7]. Se trata de una variación de la clase «InfiniteSource» donde ahora los pedidos los gestiona el usuario a través de la interfaz, indicando cuándo y cuánto pide, apoyándose en la clase «OrderArrival». En el momento en que se realiza el pedido, se incurre automáticamente en un coste prefijado. Por otra parte, con «variación» se hace referencia al reciclaje del código de la primera, pues se introducen los cambios necesarios sobre esta para desarrollarla. Este será el mismo procedimiento que se lleve a cabo para programar el resto de elementos nuevos.

[6] Este es el nombre que recibe la clase del nivel lógico. A nivel de Unity3D, de la misma manera que en (Pernas Álvarez, 2017), la clase se denomina «UnityProviderSource». Esta misma lógica se sigue igualmente para el resto de elementos mencionados aquí, a excepción de la estación de trabajo.

[7] Si bien más adelante se adoptara un modelo 3D simbólico de la fuente, al igual que los *software* comerciales, no se contempló en este primer proyecto.

El segundo de ellos es el elemento «GateQueue», con un funcionamiento similar al de la cola «ItemQueue» de (Pernas Álvarez, 2017), pero programada para enviar ítems en el momento en que el usuario decide emitir una orden de fabricación, incurriendo en el coste correspondiente. En caso contrario, no deja que el ítem comience a procesarse.

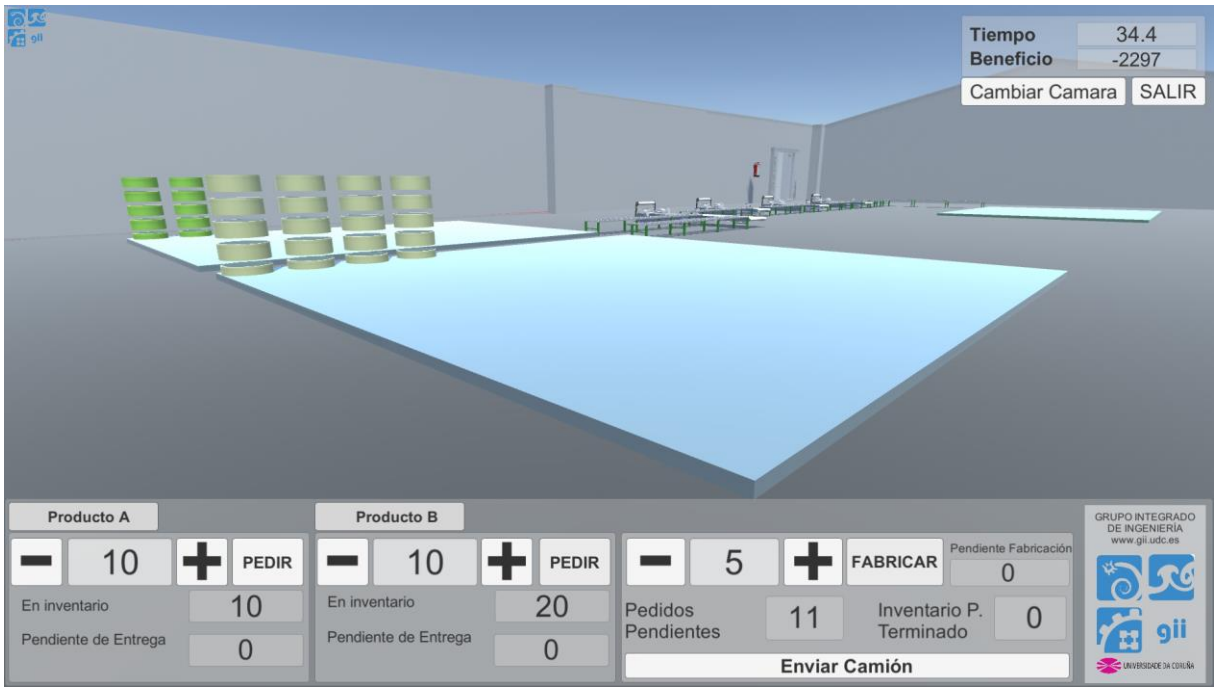

Figura 15. La cola «UnityGateQueue» esperando por la orden de fabricación del usuario. Fuente: Elaboración propia.

A continuación, siguiendo el diagrama de la Figura 14, cada una de las colas alimenta a un elemento denominado «Conveyor», pero que emplea la clase «MultiServer», ya desarrollada en su momento en (Pernas Álvarez, 2017). En dicho trabajo, esta clase lógica no solo sirve de cerebro para la estación de trabajo − «UnityWorkStation», sino que también se empleaba para modelar la grúa a través de la clase «SimpleTransporter» en Unity. Es esta última la que se modifica ahora para generar un comportamiento idéntico al de una cinta transportadora– véase Figura 16. Habiendo definido previamente la velocidad, la capacidad y los puntos de partida y llegada del ítem, el movimiento virtual de esta se genera de la misma manera que en el caso de las estaciones de trabajo en el apartado 4.2.1, sabiendo de antemano el tiempo que dura dicha operación.

Tras las cintas transportadoras, se programa un elemento nuevo como es el ensamblador múltiple – clase «MultiAssembler» −, que permite ensamblar dos o más productos de un tipo, con otro u otros tantos de otros. Se trata de una evolución de la clase «Assembler» desarrollada en (Pernas Álvarez, 2017) que permite ensamblar ítems distintos. Con «UnityMultiAssembler» como clase a nivel de Unity, es necesario especificar previamente cuales son los *inputs* de la máquina, así como los requerimientos de cada tipo de ítem, pudiendo definir tantos como se quieran. Para ello se apoya en la clase «ConstrainedInput» que instancia para definir los *inputs*, así como en la interfaz «ArrivalListener», la cual implementa para realizar la comunicación con estos. En el caso de nuestro taller, se disponen dos entradas que corresponden a cilindros de distintos colores, y se configura un ítem de salida rojo como identificador de la pieza ensamblada – véase Figura 16. En lo que se refiere a la representación visual, tanto el modelo de las cintas transportadoras, el del taller y el del ensamblador son de uso libre y gratuito descargados del sitio web 3D Warehouse. En este caso, el mismo modelo del ensamblador se emplea también para las estaciones de trabajo.

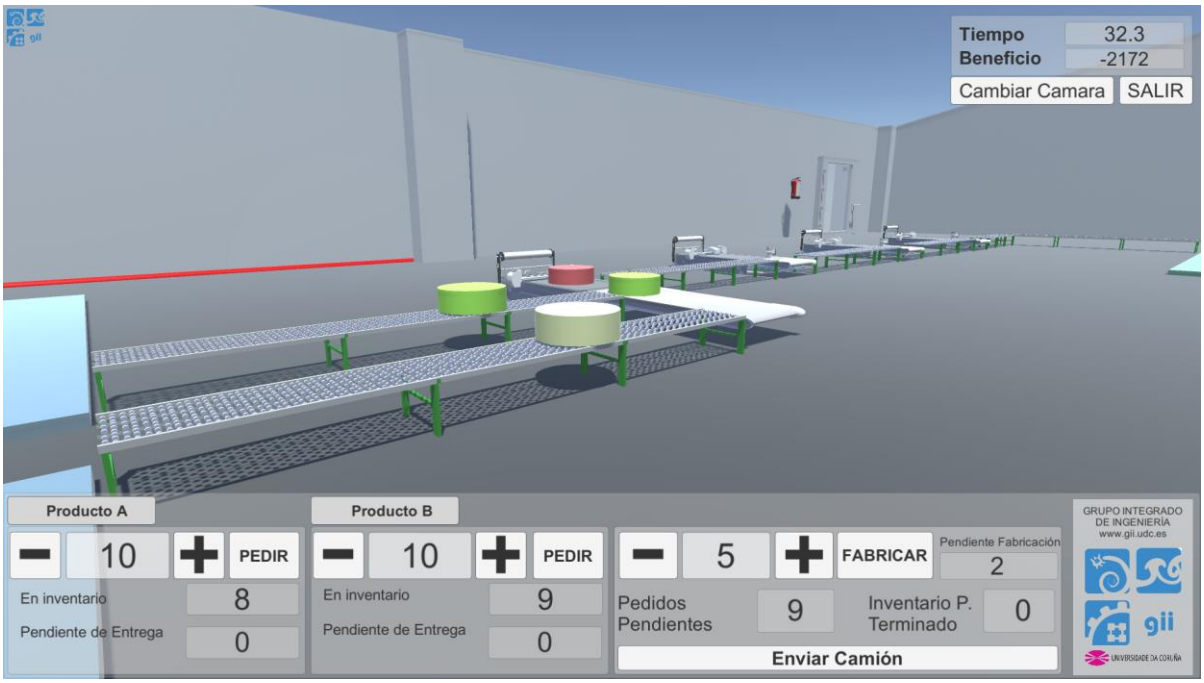

Figura 16. Vista en detalle del funcionamiento de la cinta transportadora y el ensamblador múltiple. Fuente: Elaboración propia.

Por último, de la misma manera que para la fuente, se programa un sumidero −«CustomerSink» en la Figura 14 – que hace las veces de cola mientras el usuario no decida enviar un camión, satisfaciendo así la demanda. Por tanto, con un máximo número de ítems por camión, el usuario genera costes de envío, así como ingresos por ítem, teniendo que optimizar también la gestión de estos envíos.

4.2.2.2 Fábrica Virtual

El segundo proyecto es la «Fábrica Virtual», que sigue la línea del anterior, pero incluyendo el elemento de la carretilla elevadora. El objetivo de la aplicación desarrollada vuelve a ser el mismo, donde la fábrica virtual representa el mundo real, y el usuario o alumno tiene la tarea de optimizar la cantidad de pedido y las órdenes de fabricación. Todo esto en el marco de un «juego serio» donde gana aquel que genere el mayor beneficio. Por otra parte, en este segundo proyecto también se cuida más la visualización, empleando modelos 3D de mayor calidad, que aportan mayor realismo y detalle al modelo. Estos se obtienen de nuevo se obtienen de la plataforma 3D Warehouse, de uso libre y gratuito, y algunos son modificados para ajustarlos a su propósito en el modelo.

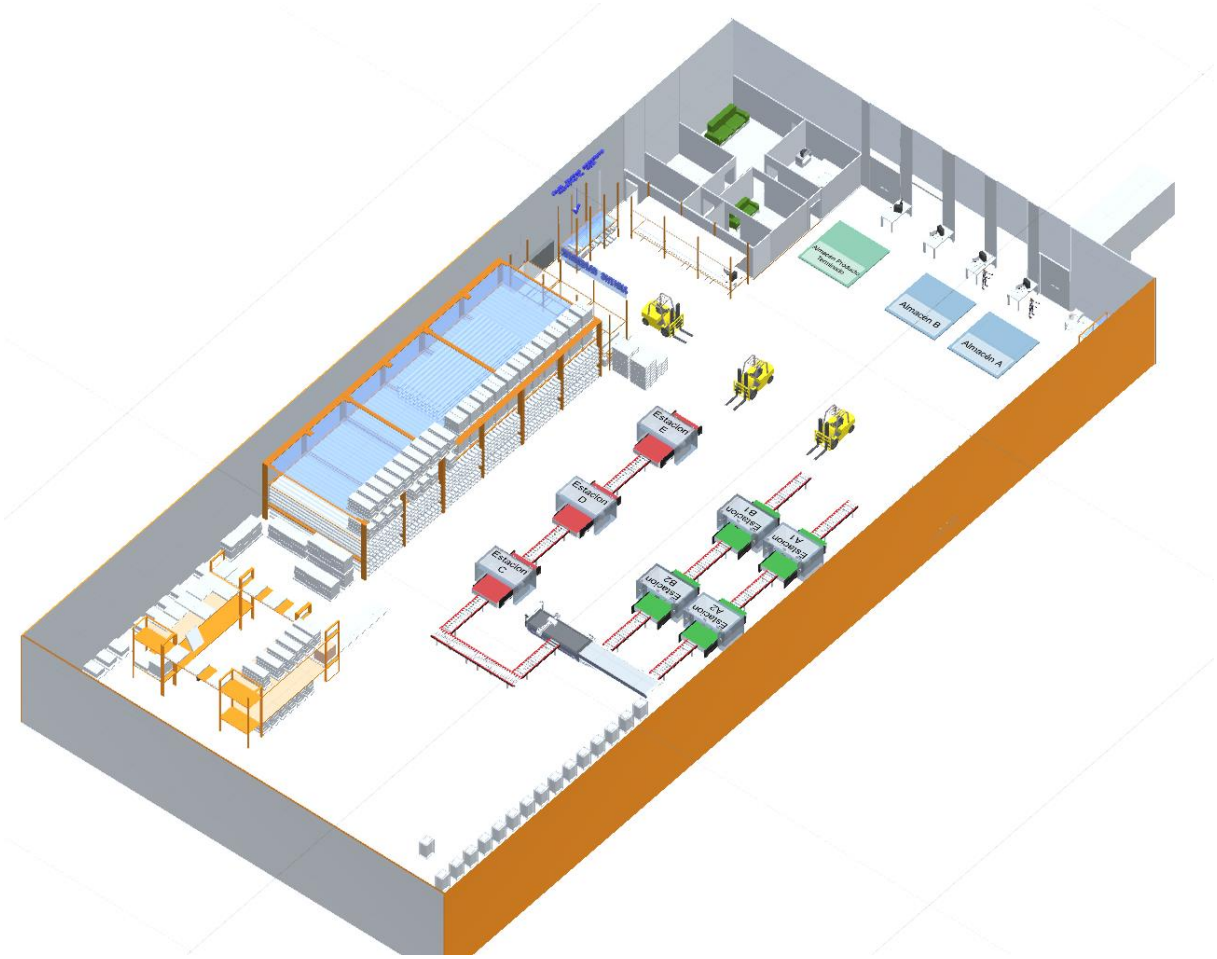

Figura 17. Vista general del modelo completo del «Fábrica Virtual». Fuente: Elaboración propia.

Aunque el proceso, representado en el siguiente diagrama de flujo, es algo más complejo y largo que en el caso anterior, todas las estaciones de trabajo comparten de nuevo la misma distribución de tiempos de procesamiento. Lo mismo sucede con la velocidad de las carretillas elevadoras.

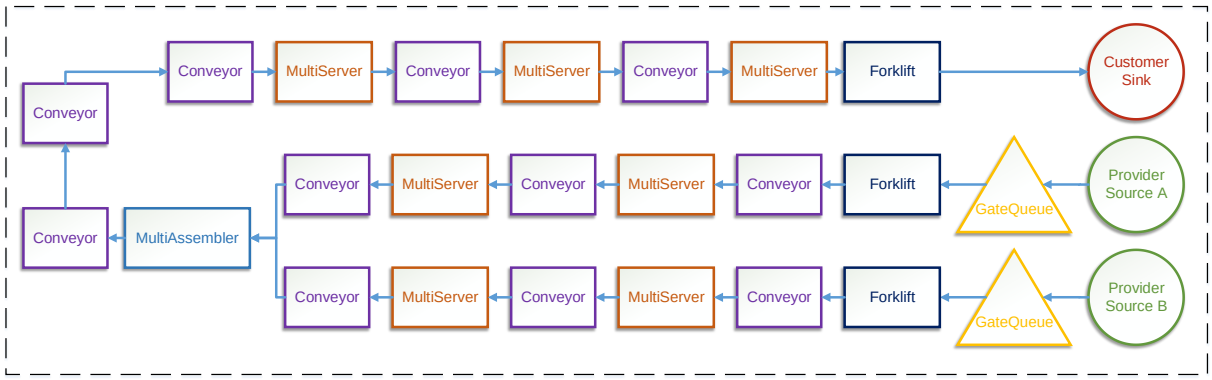

Figura 18. Diagrama de flujo de producción del proyecto «Fábrica Virtual». Fuente: Elaboración propia.

Como es observable, respecto al proceso en sí, el único elemento novedoso es la carretilla elevadora, a través de la clase de nivel lógico «Forklift». Sin embargo, como ya se mencionó previamente, supone un cambio importante en la manera de programar hasta el momento. Aunque en proyectos anteriores la comunicación entre la clase de nivel lógico y la correspondiente en el nivel virtual ya era bidireccional, la primera gestionaba completamente el comportamiento de la segunda, que se limitaba a *informar* y *representar* la visualización

programada. En cambio, ahora es la clase «UnityForkliftUI» la que en ciertos momentos le *dice* – ejecuta un método de la clase lógica que programa un determinado evento − a «Forklift» de que se ha ejecutado un determinado evento en Unity y que, por tanto, puede continuar con su comportamiento normal.

De forma más técnica, hasta ahora la clase de nivel virtual solo ejecutaba el método «Start()» que pone en marcha clase lógica. Sin embargo, ahora también ejecuta los métodos «PickItem()» y «LeaveItem()» una vez se completan estas operaciones, para avisar a «Forklift» de que puede programar el evento de recoger o dejar el ítem. Esto es debido a que no es posible determinar la duración del evento de carga o descarga de los ítems, dado que no se sabe de antemano cuándo llegará la carretilla.

Antes de continuar, es necesario explicar cómo se genera el movimiento de la carretilla. En un primer momento, se plantean distintas formas de modelarlo. Por ejemplo, sin tener en cuenta las colisiones o con movimientos más simples y rectos entre origen y destino. Tras un proceso de prueba y error, donde se valoran distintas soluciones, se acaba implementando el algoritmo de enrutamiento determinista A*, basado en el algoritmo de caminos mínimos o Dijkstra. Entre otros, este es el mismo que emplea FlexSim para generar los movimientos. En este caso, su implementación es relativamente sencilla al contar con la versión gratuita del pack de implantación para Unity desarrollado por www.arongranberg.com. Este solo requiere de la creación previa de una malla de navegación − «NavMesh», una funcionalidad de Unity − para localizar donde existen obstáculos y cuál es la superficie transitable, de acuerdo con unos parámetros determinados.

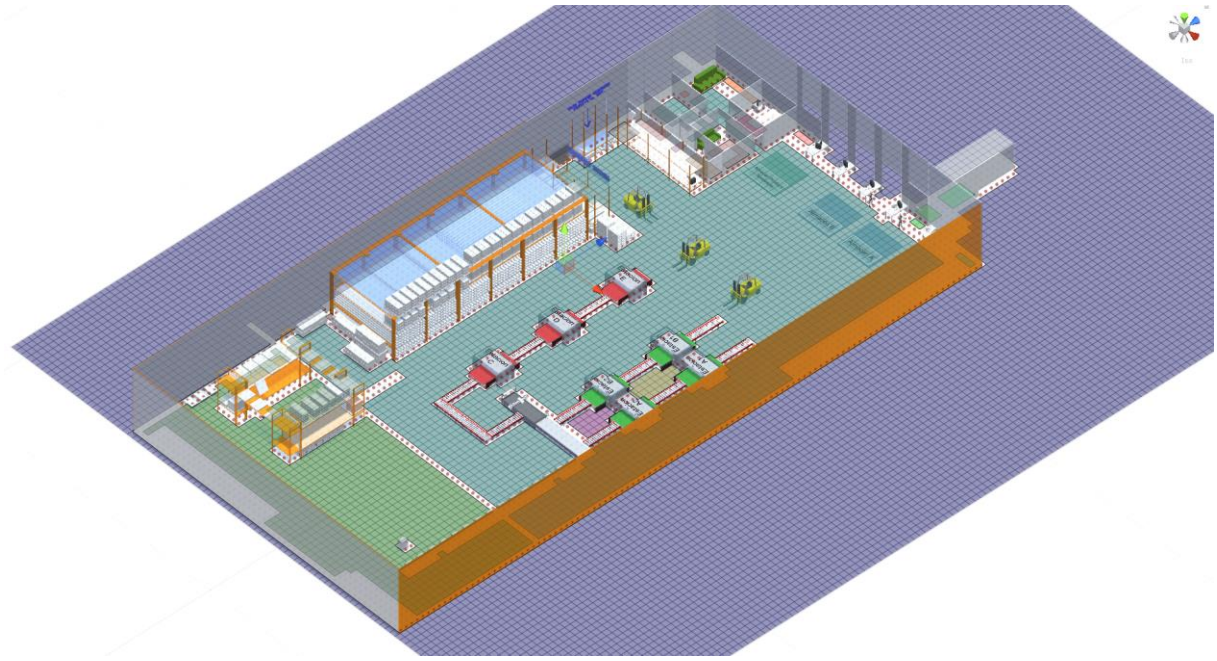

Figura 19. Detalle de la malla de navegación para la implementación del algoritmo A*. Fuente: Elaboración propia.

Una vez creada y asignada la malla junto con los puntos de origen y destino en el elemento que se desea encaminar, la implementación está completa. Sin embargo, en este caso, se adapta el código para que la clase «UnityForkliftUI» pueda asignar dinámicamente los puntos de origen y destino cada vez que sea necesario ejecutar un movimiento en la carretilla. Por tanto, solo es necesario definir al inicio los diferentes puntos entre los que se moverá el elemento. Con todo lo anterior, ahora sí es posible tener en cuenta las colisiones, aportando mayor realidad al modelo. Como ejemplo, es posible ver en la siguiente figura el enrutamiento que programa el algoritmo para esquivar los distintos obstáculos, teniendo en cuenta las dimensiones del elemento que se mueve:

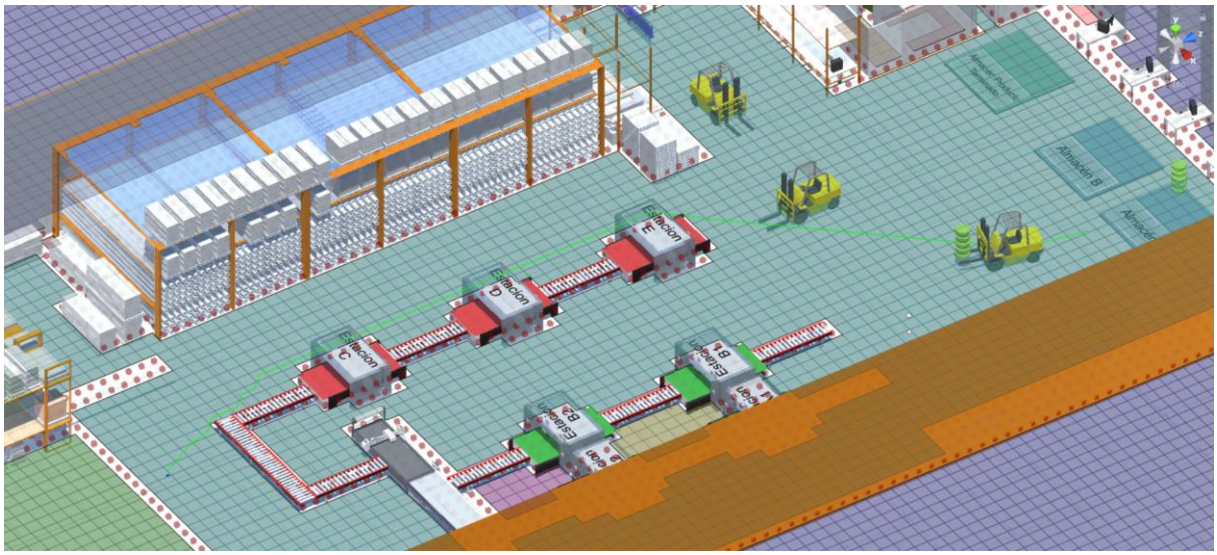

Figura 20. Detalle del enrutamiento de la carretilla elevadora empleando el algoritmo A*. Fuente: Elaboración propia.

4.2.2.3 Fábrica Virtual 2

Por último, «Fábrica Virtual 2» constituye una variación de «Fábrica Virtual» con el objetivo de modelar un operador, encargado en este caso de la recepción de los ítems por parte del proveedor. Por tanto, el diagrama de flujo del proceso es similar al anterior con esta pequeña variación:

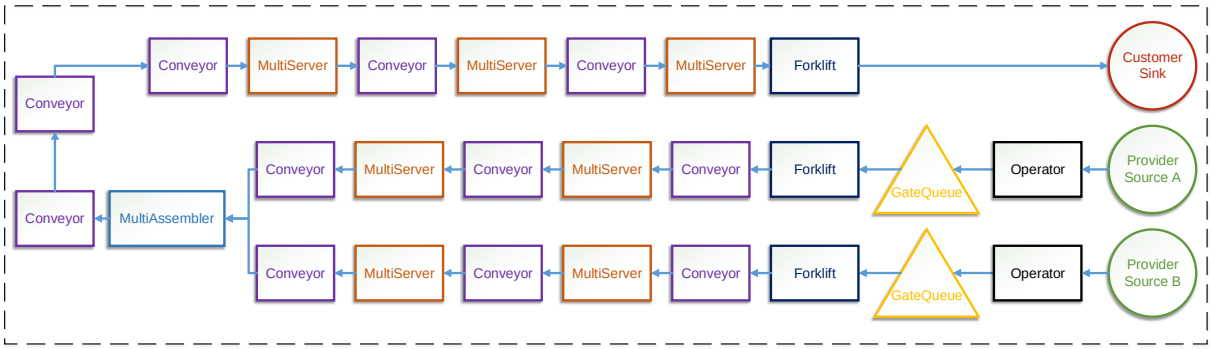

Figura 21. Diagrama de flujo del proceso del proyecto «Fábrica Virtual 2», con la inclusión del operador. Fuente: Elaboración propia.

En lo que se refiere al enrutamiento, la idea es similar a la de la carretilla elevadora, pero en este caso no se empleará un algoritmo externo para generar la ruta, sino una funcionalidad integrada de Unity. Además de las mallas de navegación, Unity también permite definir agentes de malla − «Nav Mesh Agent» −, que serán los objetos o humanoides que se moverán por la malla. En este sentido, se programa la clase «UnityOperator» para que interactúe con la clase «NavMeshAgent», donde el propio objeto operador es una instancia a dicha clase. La comunicación es idéntica a la de la carretilla, cambiando el destino y gestionando los eventos de marcha y paro desde la clase del nivel virtual.

Por otra parte, respecto a la visualización, se emplea el paquete «Characters», el mismo que se había utilizado para la gestión de la cámara en primera persona. Este paquete gratuito contiene un modelo 3D básico de humanoide, con un avatar asignado que define su esqueleto y las diferentes articulaciones, con el fin de generar posteriormente movimientos lo más naturales posibles. Mismamente Unity cuenta con un módulo específico para modificar este el *mapeado* del modelo así como la posición de sus distintas partes – cabeza, brazos, piernas

o hasta dedos – según la pose – véase Figura 23. Por tanto, este modelo 3D es el empleado para realizar la visualización de nuestro operador:

Figura 22. Modelo 3D de humanoide del paquete de Unity «Characters» empleado en el proyecto. Fuente: Elaboración propia.

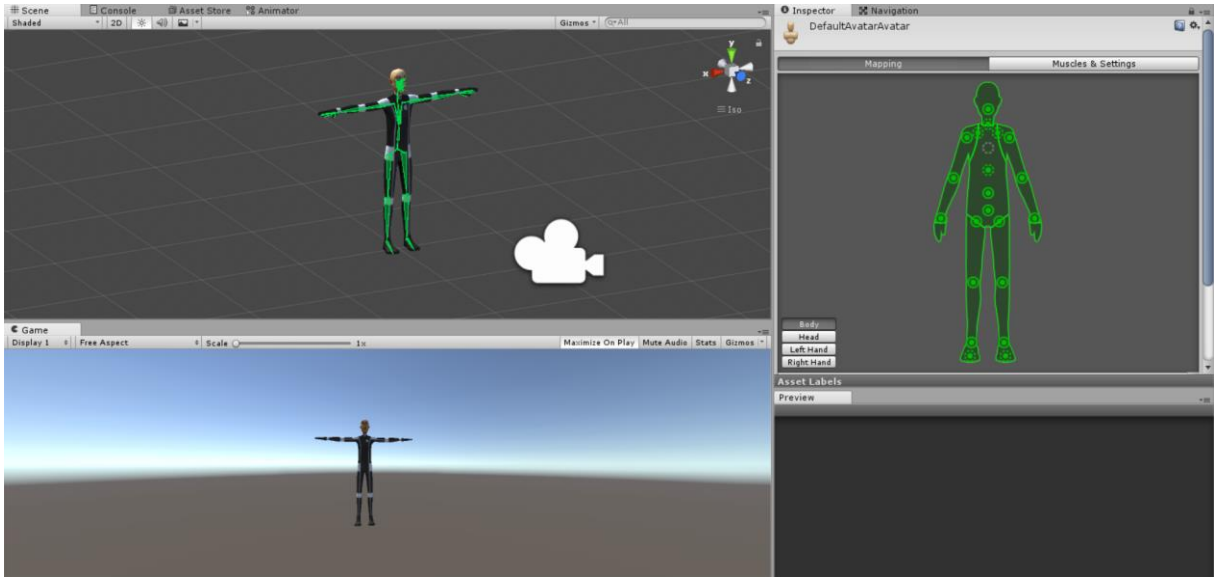

Figura 23. Escena específica donde es posible configurar el *avatar* del operador humanoide. Fuente: Elaboración propia.

Por otra parte, con el fin de aumentar el realismo, también se emplea el módulo de animación «Animator» que tiene integrado el programa. Este módulo permite realizar manualmente clips de animaciones modificando cada cierto número de fotogramas la posición del objeto o de alguna de sus partes. Posteriormente, es posible programar un comportamiento más complejo del objeto – en nuestro caso, del humanoide – definiendo un diagrama de estados que configura una animación completa, compuesta por los clips creados y añadidos. Como ejemplo, en nuestro caso se emplean de base algunas de las animaciones – como caminar, o parar – que trae por defecto el paquete «Characters». A partir de estas, se general nuevos clips con movimientos de levantamiento y bajada de brazos, simulando la recogida del ítem. Finalmente, se configura una animación – véase Figura 24 − a través de un diagrama de estados parametrizado que será instanciado por la clase «UnityOperator».

Esta se encargará de variar los valores de los parámetros de manera que el estado del operador cambie; por ejemplo, cuando tiene que dejar de caminar ya que ha llegado a su destino.

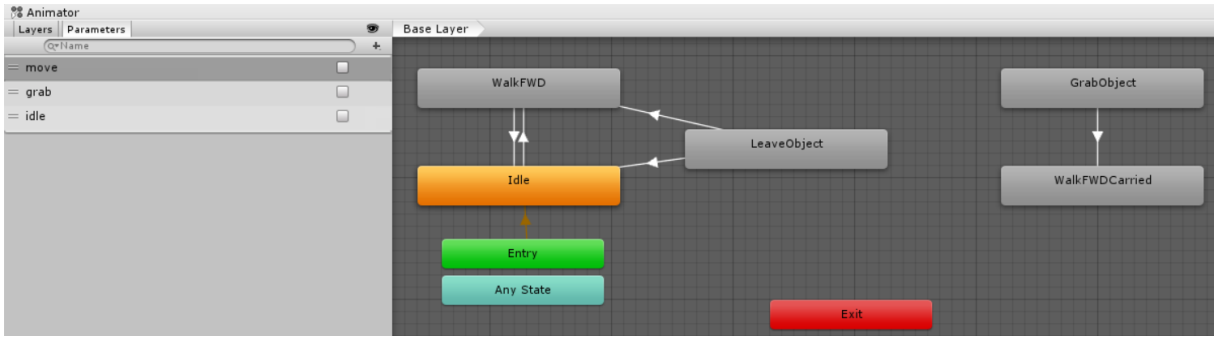

Figura 24. Parámetros (izquierda) y diagrama de estados (derecha) de la animación del operador. Fuente: Elaboración propia.

Se muestra a continuación una captura desde la vista en primera persona del movimiento de transporte de los operadores.

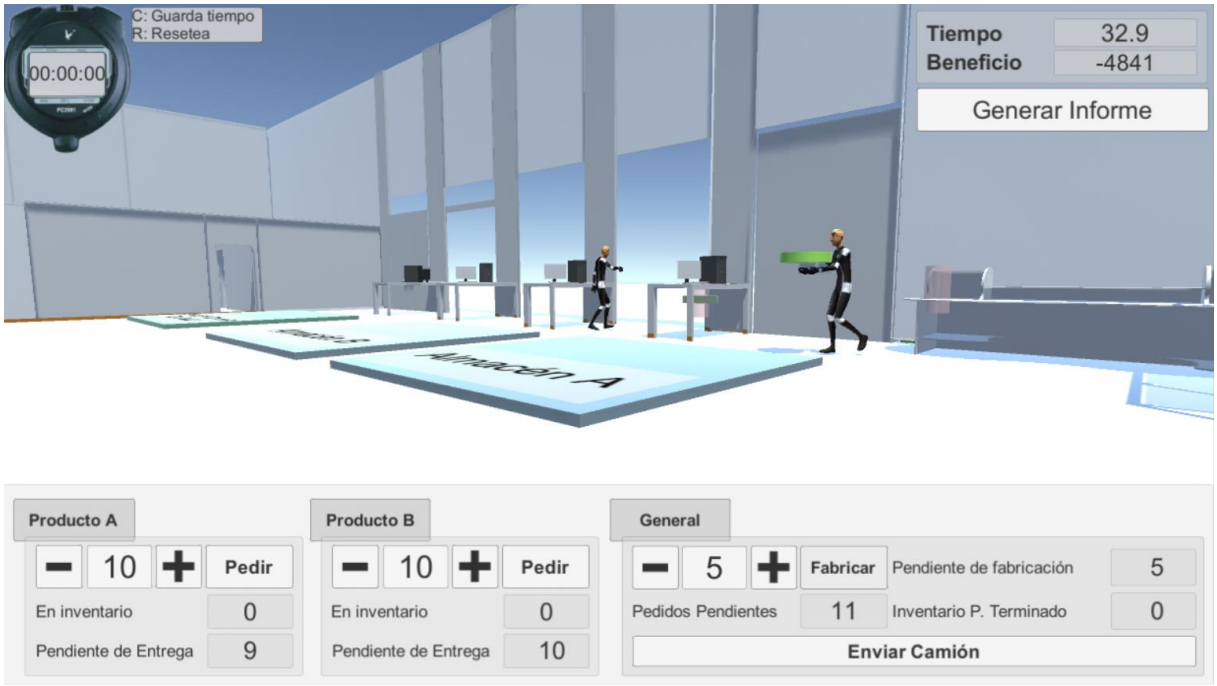

Figura 25. Vista en primera persona de los operadores en «Fábrica Virtual 2». Fuente: Elaboración propia.

Llegados a este punto, no se prosigue más allá con el desarrollo de este elemento, habiendo explorado así las funcionalidades de animación 3D de Unity, así como el control de esta por código. Principalmente esto es debido a que la construcción de animaciones 3D para elementos en concreto se aleja de la idea de versatilidad y reciclaje del código sobre el que tanto se insiste en el presente trabajo. Junto con esto, como se mostrará más adelante, este recurso no se emplea en el caso de demostración, por lo que tampoco resulta de interés una mayor profundización

Con todo, quedaría como posible trabajo el desarrollo de otros funcionamientos posibles del operador, como su participación en la fabricación de un ítem por una estación de trabajo, o en la configuración de la misma antes de comenzar el procesado. Asimismo, cabe decir que un mayor realismo en el modelo y en las animaciones sí tendría cabida en posibles

investigaciones del uso de *softwares* de videojuegos en análisis ergonómicos, donde el nivel de detalle que nos permiten estos programas podría resultar muy útil en estudios de este tipo.

4.3 Desarrollo de la interfaz

Tras la ampliación del código, la siguiente fase del trabajo se centró en el desarrollo de la interfaz de usuario (GUI, por sus siglas en inglés). Esta constituye una de las partes más importantes del presente trabajo en lo relativo a la interacción con el modelo de simulación. En este sentido, su desarrollo también supone un trabajo de investigación y revisión del estado del arte en torno las diferentes formas de interacción usuario-mundo virtual vía VR, tanto en el mundo de la simulación como en el mundo de los videojuegos. Precisamente por el enorme avance de este durante los últimos años, la interfaz de comunicación entre los motores de videojuegos y los dispositivos de realidad virtual está totalmente desarrollada y, por tanto, el trabajo se centra en readaptar los diferentes recursos de interés a los propósitos de este trabajo.

Por consiguiente, dentro de la creación de la interfaz de usuario, podemos diferenciar dos fases principales:

- La readaptación y empleo de recursos relativos al movimiento y la interacción con la interfaz gráfica y el mundo virtual. Entre ellos, la vista en primera persona, el rastreo de la posición del usuario, el teclado virtual o el comportamiento del menú de configuración del modelo son ejemplos de algunos reaprovechados pertenecientes al paquete Oculus Sample Framework de Oculus VR (véase 1.3 Recursos del proyecto). La fase de readaptación ha consistido en la revisión y modificación del funcionamiento y el aspecto de estos recursos, de manera que cumpla con el propósito deseado en el presente trabajo.
- La creación de la estructura de menús, sub-menús y elementos que, a través del desarrollo de una librería, permiten la creación, configuración y modificación del modelo por parte del usuario.

4.3.1 Readaptación de recursos existentes

En el apartado actual se pretende explicar brevemente los diferentes recursos aprovechados de distintas escenas de muestra existentes en el paquete Oculus Sample Framework. Sin entrar en detalle, en todos ellos ha sido necesario revisar el código ya que no se importaron o emplearon todas las funcionalidades que incluían de serie. Esta readaptación consistió en la mayoría de los casos más concretamente en una simplificación del código.

El primero de ellos son los scripts relativos al movimiento y configuración del avatar virtual, así como el rastreo del usuario. Entre ellos, podemos encontrar:

- Scripts que configuran la visión estereoscópica del visor VR, así como el rastreo (*tracking*, en el argot) tanto de las gafas de realidad virtual (*head-mounted display*) como de los controladores (*controllers*). Este recurso viene totalmente configurado para ser arrastrado directamente sobre la escena, una vez eliminada la cámara principal, e incluye asimismo la configuración del puntero o mira para la interacción con una posible interfaz.
- El avatar proporcionado por Oculus en el kit para configurar unas manos virtuales, de manera que se aumente la sensación de inmersión. “OVR Avatar” es el nombre de la clase principal que gestiona este comportamiento, la cual se modifica posteriormente para gestionar la interacción con el teclado virtual.

Figura 26. Detalle de las manos virtuales desde la perspectiva del usuario. Fuente: Elaboración propia.

- Y, finalmente, el movimiento y la gestión de las colisiones con los elementos del mundo virtual se realiza a través de la clase “Sample Player Controller”, que ya había sido empleada en (Pernas Álvarez, 2017) y constituye la clase de referencia para el movimiento en primera persona. Es necesario modificarla para configurar la vista de pájaro, así como para gestionar su movimiento con los controladores VR.

Por otra parte, se emplean dos recursos más de importancia clave para desarrollar las funcionalidades del simulador:

- Por un lado, una serie de scripts que permiten crear el menú desde el cual se configurará la escena. El funcionamiento consiste en generar el menú automáticamente a una distancia y altura predeterminadas respecto al usuario cada vez que se inicia el simulador, permitiendo la recolocación posteriormente mientras se está en la escena. Este recurso contiene por defecto una serie de sub-menús y opciones que permiten configurar algunos parámetros relativos a la cámara o a la escena. La mayoría de ellos son eliminados y sustituidos por otros de manera que se incluyan todas las funcionalidades necesarias para el desarrollo del simulador.

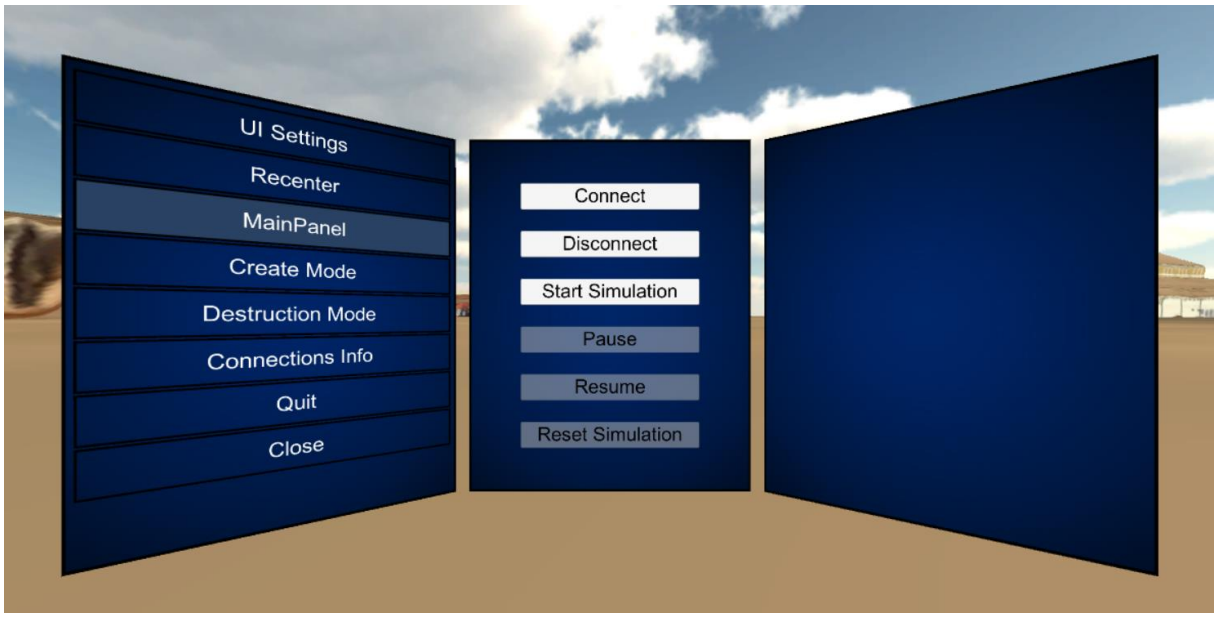

Figura 27. Detalle del menú principal desde el punto de vista del usuario. El aspecto es el mismo que el del recurso de Oculus Sample Framework. Fuente: Elaboración propia.

- Por otro, la necesidad de definir parámetros como el tiempo de simulación o la capacidad de las colas hace necesaria la creación de un teclado virtual que permita introducir estos datos. Para ello, como se mencionó en 1.3 Recursos del proyecto, se emplea el recurso VRKeys de uso libre y gratuito. No obstante, su desarrollo es específico para un determinado caso contemplado en dicho proyecto. Por tanto, ha sido necesario readaptar sus funcionalidades, así como el tipo de teclado que ha desplegarse en cada momento. Un ejemplo de ello es mostrar únicamente los números cuando no es posible introducir otro tipo de caracteres, lo que a su vez ahorra trabajo de programación en otras partes del modelo. Respecto a la interacción con el mismo, ha sido necesario readaptar el código para que las manos del avatar se sigan mostrando cuando el usuario está empleando el teclado. Se muestra a continuación una captura de la vista del teclado desde la perspectiva del usuario:

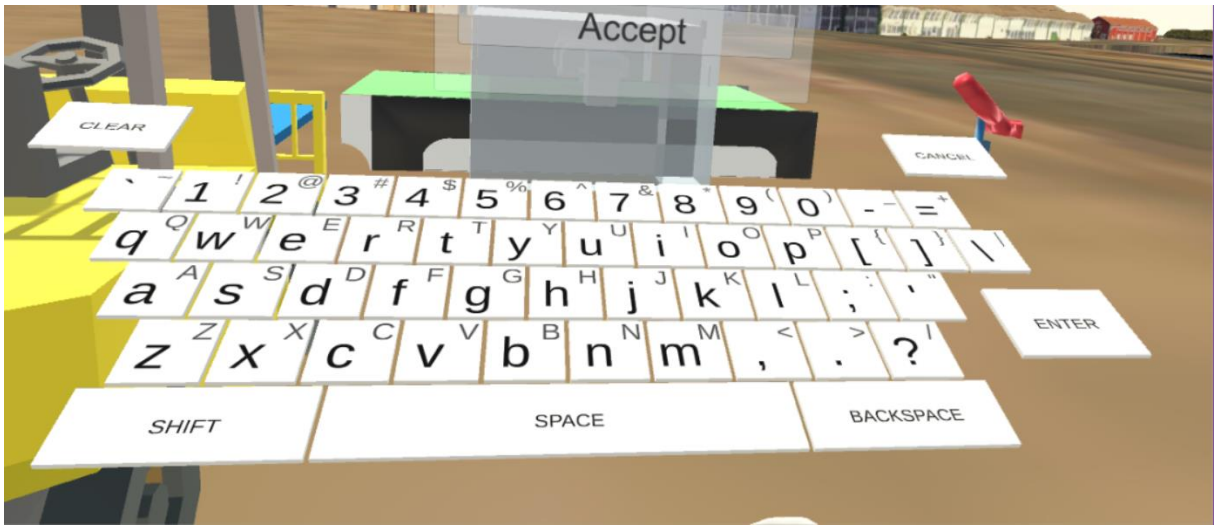

Figura 28. Detalle del teclado virtual desde la perspectiva del usuario. Fuente: Elaboración propia.

Todo lo anterior resumen brevemente los recursos necesarios para generar una comunicación adecuada e inmersiva usuario-mundo virtual. Muchos de estos scripts son

modificados y implementados con un método de prueba y error, hasta conseguir unas determinadas funcionalidades. En este sentido, la necesidad de revisar todo lo configurado haciendo uso de las gafas se traduce en un esfuerzo añadido y unos mayores tiempos de desarrollo, respecto a una depuración normal con la pantalla el ordenador.

4.3.2 Creación de los menús

Con respecto a la fase de creación de los menús, primero fue necesario definir el alcance de las modificaciones que puede realizar el usuario sobre el modelo. Sin excederse en las capacidades que se pretenden desarrollar al ser un proyecto experimental, se proponen principalmente las siguientes:

- Creación elementos de simulación y situación en el espacio con precisión, modificando tanto su posición como su rotación. Estos en elementos son los de la librería SimElements desarrollados en (Pernas Álvarez, 2017) y ampliados en este proyecto. Entre ellos se encuentran: fuente, cola, carretilla elevadora, procesador, ensamblador, cinta transportadora y sumidero. A la hora de crear un elemento en un determinado punto del espacio, es necesario comprobar si su colocación interfiere con algún elemento existente en dicha localización, informando al usuario en dicho caso. Finalmente, se informa mediante un mensaje temporal del éxito de la operación.

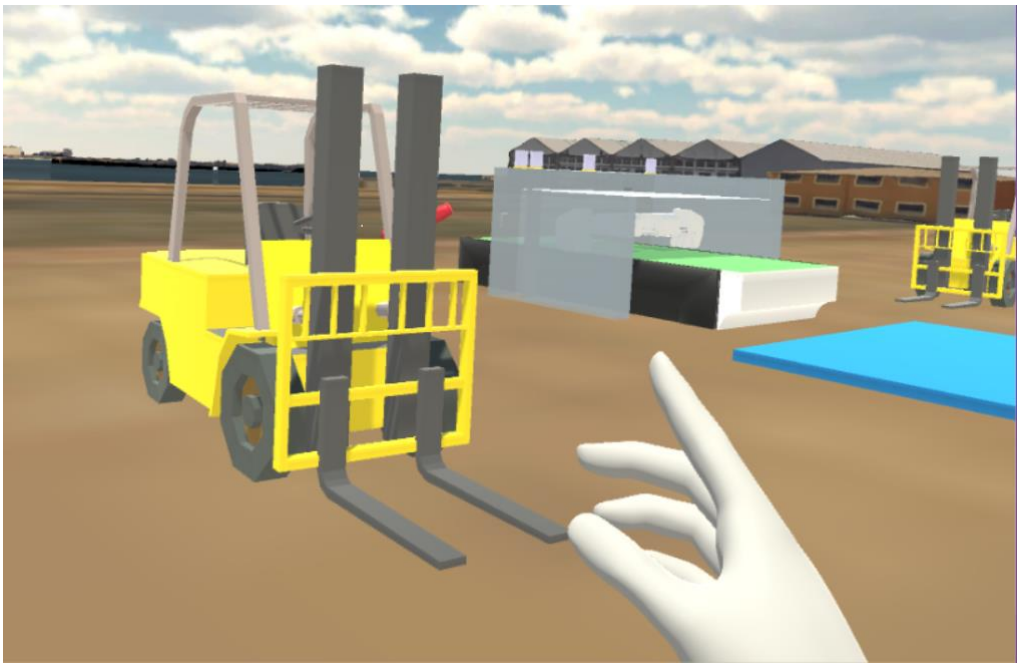

Figura 29. Detalle de la creación de una carretilla elevadora desde la perspectiva del usuario. Fuente: Elaboración propia.

- Creación de conexiones simples y múltiples entre los diferentes elementos de la escena. En este caso, cada vez que el usuario realiza una conexión nueva, es necesario comprobar si existe alguna otra conexión previa que se solape con la nueva, no permitiendo esta última en caso afirmativo. Para proporcionar una mayor intuición, se señalan los elementos seleccionados como *inputs* y *outputs* de la conexión mientras esta está siendo realizada. Finalmente, se muestra un mensaje temporal que informa del resultado de la operación.

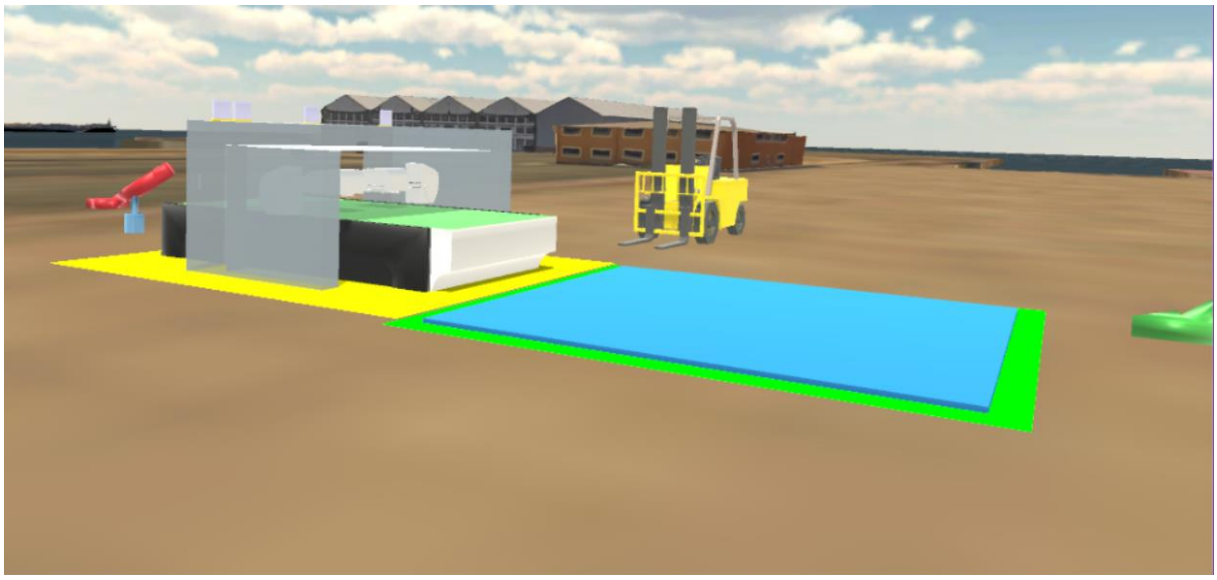

Figura 30. Detalle de la selección de elementos durante la creación de las conexiones. El recuadro de selección verde representa el *input* de la conexión, y el amarillo el *output*. Fuente: Elaboración propia.

- Destrucción de los elementos de simulación que existan en la escena, así como de sus conexiones.

Figura 31. Detalle del panel informativo sobre el éxito de destrucción de un elemento. Fuente: Eliminación propia.

- Inserción y modificación de los parámetros de configuración de los elementos de simulación. Para ello se emplean menús independientes que muestran los valores actuales de los distintos parámetros, permitiendo modificar su valor, en algunos casos haciendo uso del teclado virtual. Como ejemplo, la introducción del tiempo entre llegadas de la fuente, o de las posiciones de inicio y fin en el movimiento de la carretilla elevadora − Figura 32.

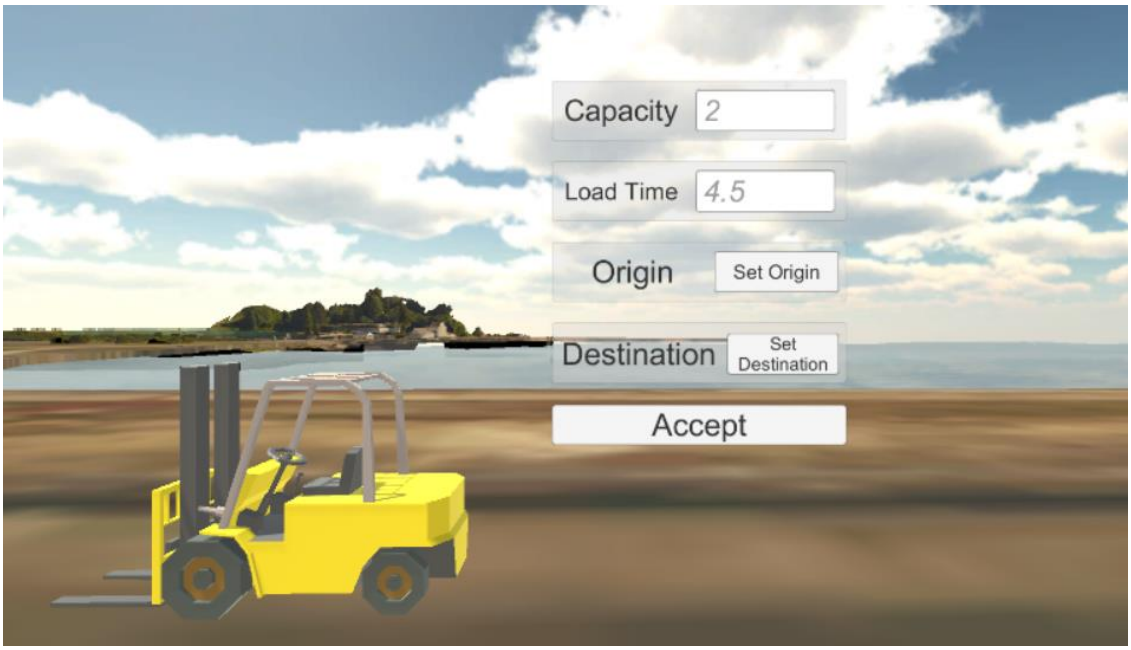

Figura 32. Detalle del panel de configuración de la carretilla elevadora. Fuente: Elaboración propia

- Introducción del tiempo de simulación − Figura 33 − y capacidad de ejecutar, parar y resetear la misma − Figura 27.

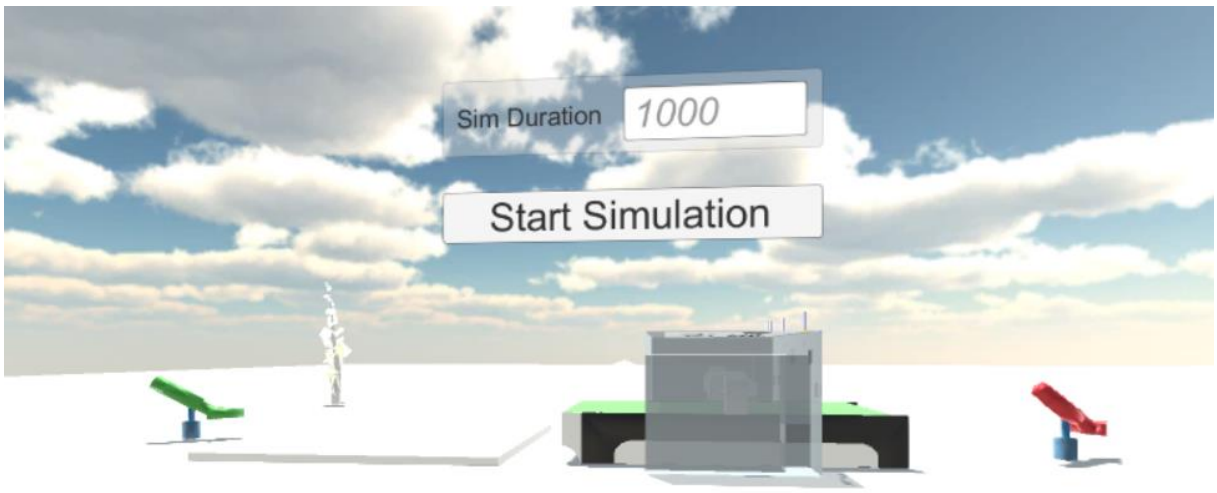

Figura 33. Detalle del panel de configuración del tiempo del modelo. Fuente: Elaboración propia

Con todo lo anterior, el siguiente diagrama resume la estructura jerárquica de las diferentes clases desarrolladas dentro de la librería «UnityFunctions» y su conexión con «SimSElements»:

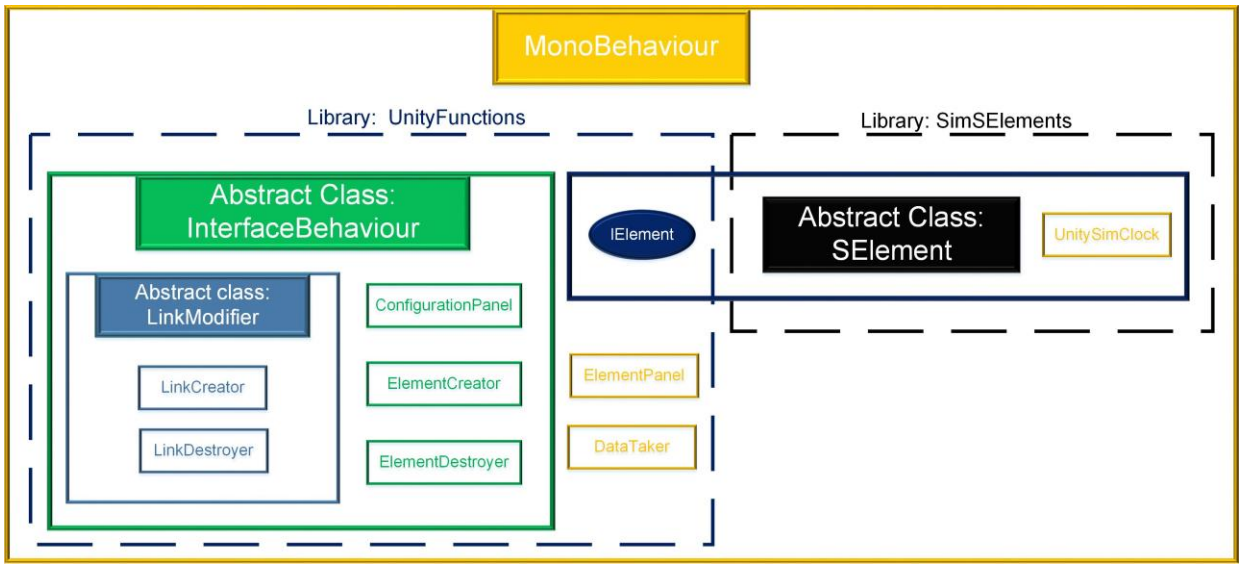

Figura 34. Estructura jerárquica de las distintas clases de la librería «UnityFunctions». Fuente: Elaboración propia.

Finalmente, a continuación se muestra una estructura de los diferentes de menús y submenús de la interfaz. Con ello, queda realizada la parte del simulador relativa a la interacción.

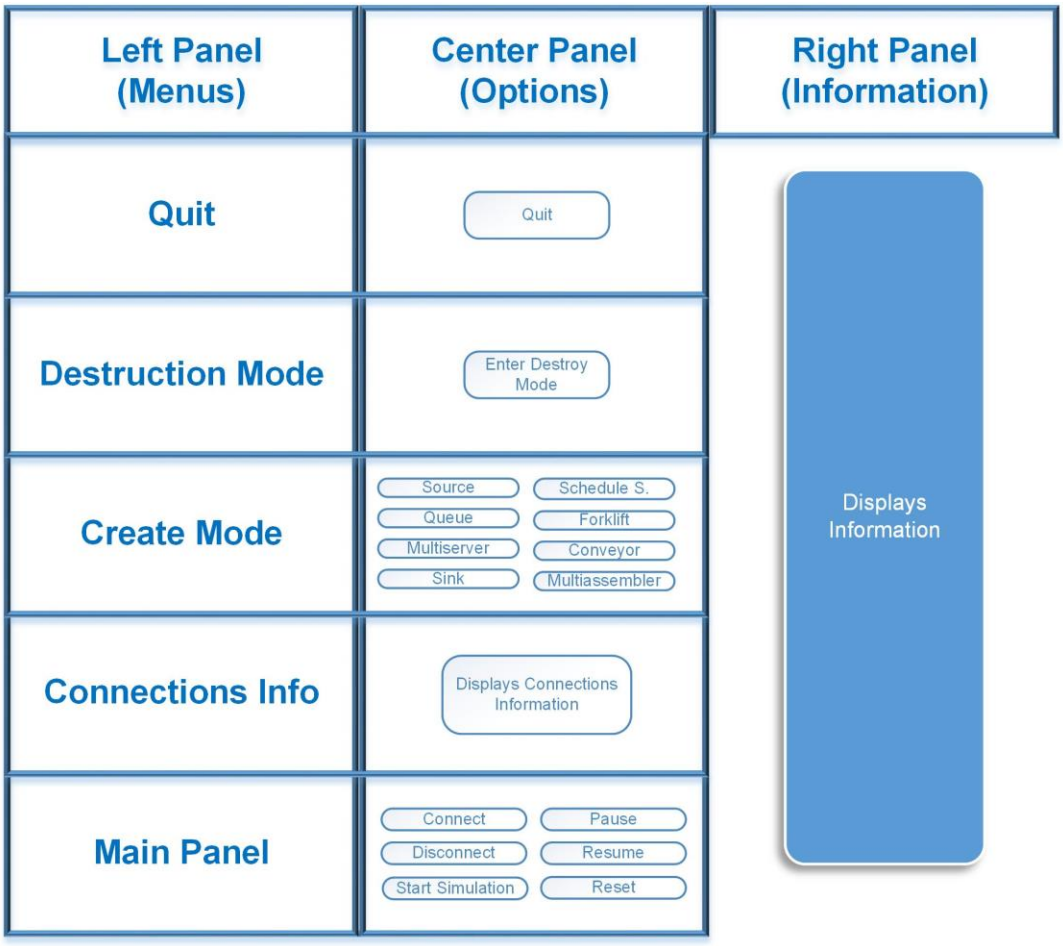

Figura 35.Desglose del menú principal en submenús y botones. Fuente: Elaboración propia

4.3.3 Ensayos

Como apartado añadido, una vez completada la creación del simulador, se escogieron 5 personas con conocimientos y experiencia en simulación de eventos discretos y que, concretamente, conocen de antemano la interfaz de realidad virtual que presenta uno de los *software*s comerciales con más desarrollo en el campo como es FlexSim. En este sentido, la experimentación ha consistido en presentar virtualmente a cada uno de los participantes un modelo muy simple de simulación fuente-cola-procesador-sumidero como el de la siguiente figura:

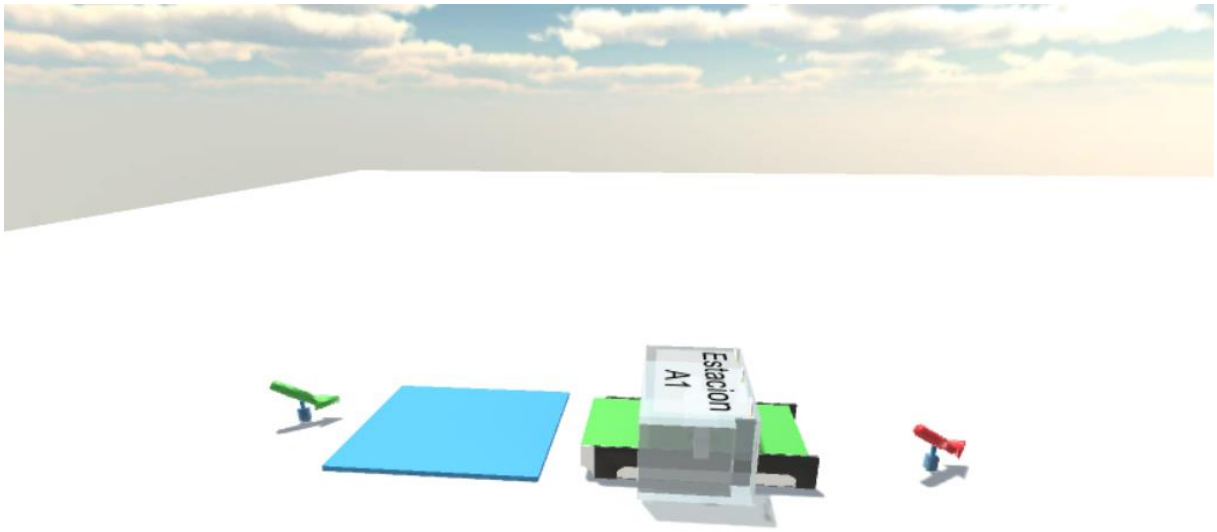

Figura 36. Caso propuesto de ensayos con el simulador. Fuente: Elaboración propia.

Una vez colocados los periféricos de VR y explicado brevemente los controles para poder manejar el simulador, se les ha propuesto la siguiente serie de tareas:

- Experimentar libremente con los diferentes menús de la interfaz, así como con el movimiento a través de la escena.
- Destrucción del elemento tipo cola de la escena y creación de un elemento similar en la misma posición inicial.
- Conexión aguas arriba y abajo del elemento cola con la fuente y el procesador respectivamente.
- Configuración de la capacidad y la posición de los ítems de la cola a través de su panel de configuración.
- Finalmente, introducir el tiempo de simulación e iniciar la misma.

Figura 37. Imágenes de dos de los participantes en la fase de pruebas del simulador, durante las pruebas con el mismo. Fuente: Elaboración propia.

Una vez realizada la pequeña fase de pruebas, se les ha pasado un pequeño formulario con una serie de aspectos a valorar sobre el simulador, siendo 1 la puntuación más baja o muy en desacuerdo, y 4 la más alta o muy de acuerdo. Los ítems a valorar han sido los siguientes:

- Ítems relativos a la experiencia VR:
  - ¿Crees que la sensación de inmersión en con este simulador es superior con respecto a la visualización VR de un *software* comercial de eventos discretos?
  - ¿Has tenido sensación de mareo o molestia durante la prueba?
  - ¿Consideras el sistema de movimiento con los joysticks adecuado con respecto a otro sistema alternativo como el tele-transporte?
- Ítems relativos a la interfaz gráfica de usuario desarrollada en el simulador:
  - Respecto a la sencillez y facilidad para entender la interfaz VR: ¿te ha parecido el simulador intuitivo?
  - Manejabilidad: ¿crees que el simulador permite crear y modificar rápida y fácilmente un modelo como el del ejemplo?
  - ¿Consideras la calidad gráfica de este simulador superior con respecto a un software comercial de eventos discretos?
  - ¿Piensas que los diferentes menús son visualmente agradables, están bien estructurados y son claros?
- Ítems relativos al uso de un simulador VR DES como posible herramienta a corto-medio plazo:
  - ¿Crees que un simulador basado en VR puede ser mejor que un simulador convencional respecto a la creación y modificación de un modelo de eventos discretos?
  - ¿Crees que un simulador basado en VR puede ser mejor que un simulador convencional respecto a la configuración del *layout* de un modelo de eventos discretos?

  - ¿Crees que un simulador basado en VR puede ser mejor que un simulador convencional a la hora de crear un modelo de simulación por personal técnico?
  - ¿Crees que un simulador basado en VR puede ser mejor que un simulador convencional para fines de comunicación de conclusiones o decisiones a personal no técnico? (por ejemplo, a un directivo de una empresa).
  - ¿Crees que un simulador basado en VR puede ser mejor que un simulador convencional en lo relativo a la experimentación con diversos escenarios?
  - ¿Crees que un simulador basado en VR puede ser una herramienta para fines comerciales?
  - ¿Crees que en un futuro existirán herramientas como este simulador?
- Otras observaciones.

Las valoraciones medias de los participantes han sido las siguientes:

Tabla 1. Tabla con las valoraciones medias de los participantes en la encuesta. Fuente: Elaboración propia.

Ítem	Media de los participantes
¿Crees que la sensación de inmersión con este simulador es superior con respecto a la visualización VR de un software comercial de eventos discretos?	4
¿Has tenido sensación de mareo o molestia durante la prueba?	2,2
¿Consideras el sistema de movimiento con los joysticks adecuado con respecto a otro sistema alternativo como el tele-transporte?	3,4
Respecto a la sencillez y facilidad para entender la interfaz VR: ¿te ha parecido el simulador intuitivo?	3,8
Manejabilidad: ¿crees que el simulador permite crear y modificar rápida y fácilmente un modelo como el del ejemplo?	3,4
¿Consideras la calidad gráfica de este simulador superior con respecto a un software comercial de eventos discretos?	4
¿Piensas que los diferentes menús son visualmente agradables, están bien estructurados y son claros?	3,2
¿Crees que un simulador basado en VR puede ser mejor que un simulador convencional respecto a la creación y modificación de un modelo de eventos discretos?	2
¿Crees que un simulador basado en VR puede ser mejor que un simulador convencional respecto a la configuración del layout de un modelo de eventos discretos?	2,6
¿Crees que un simulador basado en VR puede ser mejor que un simulador convencional a la hora de crear un modelo de simulación por personal técnico?	2,4
¿Crees que un simulador basado en VR puede ser mejor que un simulador convencional para fines de comunicación de conclusiones o decisiones a personal no técnico? (por ejemplo, a un directivo de una empresa).	4
¿Crees que un simulador basado en VR puede ser mejor que un simulador convencional en lo relativo a la experimentación con diversos escenarios?	2,8

¿Crees que un simulador basado en VR puede ser una herramienta para fines comerciales?	4
¿Crees que en un futuro existirán herramientas como este simulador?	3,8

Finalmente, entre las observaciones destacadas se encuentran las siguientes:

- «Respecto a la pregunta del tele-trasporte yo creo que sería interesante poder desplazarte de ambas formas, una para moverte más rápidamente y otra para mayor precisión.».
- «Una vez superado el conocimiento de los controles del mando, el simulador es muy intuitivo.».

A pesar de que la muestra es pequeña, el propósito principal de la encuesta es examinar las ventajas que realmente puede tener un simulador de eventos discretos en realidad virtual desde la perspectiva de personas ajenas al desarrollo del presente proyecto. Este pequeño conjunto de valoraciones sirve también como base para:

- Por un lado, evaluar la validez del simulador experimental para la creación y modificación de un modelo de eventos discretos, dentro de las posibilidades del mismo relativas al alcance del trabajo.
- Y por otro, analizar cuáles son las áreas en las que, a corto y medio plazo un simulador de estas características podría tener potencial en el mercado. En este sentido, nos permite definir en qué aspectos debería concentrarse el esfuerzo a la hora de desarrollar una herramienta de similares características

4.4 Caso de demostración

Como último punto del trabajo, se propone la aplicación del simulador desarrollado a un caso real de demostración donde pueda entreverse el potencial de la herramienta desarrollada. Como ya se aclaró en 1.3 Recursos del proyecto, el caso de demostración se ha desarrollado gracias a la información y los recursos 3D proporcionados por la Unidad Mixta de Investigación UDC – Navantia. No obstante, por motivos de confidencialidad, ha sido necesario reconstruir un caso ficticio basado en dicha información de manera que demuestre igualmente una posible aplicación del simulador.

4.4.1 Descripción del caso

El caso de demostración se basa en la construcción de una serie de columnas que conforman una estructura flotante en el mar. La elección del caso responde esencialmente a dos aspectos:

- Por un lado, la planificación de las tareas de fabricación de estas, debido a las diferentes restricciones que caracterizan el proceso y que son enumeradas más adelante.
- Por otro lado, en el proceso real, el retraso en una de las tareas previas a la fabricación obliga a llevar a cabo una evaluación de los posibles riesgos y, consecuentemente, a la elaboración de un plan de mitigación que cambia la planificación inicial, afectando tanto a proveedores como al cliente. En este sentido, se pretende evaluar la validez del simulador como herramienta de evaluación de las planificaciones posibles, así como para convencer al cliente de que la estrategia desarrollada es adecuada.

En la siguiente figura se muestra un diagrama simplificado del proceso, resaltado la operación que se va a simular aquí:

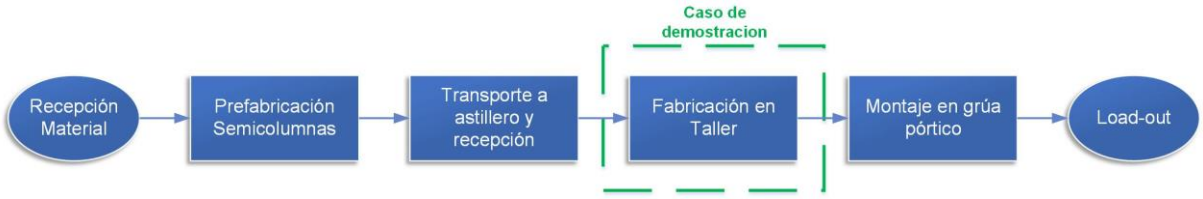

Figura 38. Diagrama de flujo simplificado del caso de demostración. Recuadrada en verde, la tarea que se emplea para el caso de demostración. Fuente: Elaboración propia.

Con respecto a las restricciones, toda la información es extraída de la planificación general del proyecto, la cual se desarrolla a lo largo de 44 semanas, desde el comienzo de la fabricación hasta el fin de la última tarea de ensamblaje. Esta se muestra en la siguiente tabla:

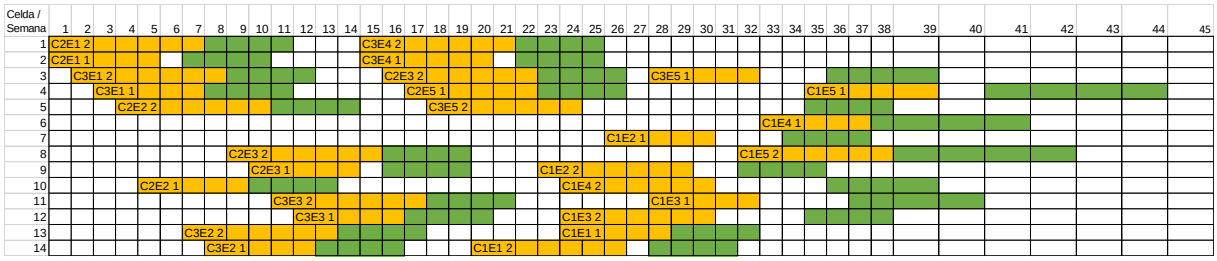

Figura 39. Diagrama de Gantt con la planificación *real* del caso de demostración.

Por motivos de confidencialidad, no es posible mostrar las fechas reales, por lo que se asume que el proceso comienza en la semana 1, y termina en la semana 44. En este sentido se enumeran continuación las restricciones extraídas:

- El número total de estructuras que se fabrican es de 5, enumeradas de la estructura número 1 a la 5.
- Cada una de las estructuras flotantes cuenta con 3 columnas, de las cuales una de ellas es una columna que denominaremos «central», frente a las otras dos que serán «secundarias».
- Cada columna divide a su vez en una semicolumna superior y una inferior, teniendo estas distintos tiempos de fabricación.
- Asimismo, el proceso de fabricación de cada semicolumna se divide en dos partes: la fabricación de las distintas partes de la chapa exterior, y el ensamblado de estas, de nuevo, con distintos tiempos de fabricación.
- El tiempo de fabricación de una semicolumna superior es de 7 semanas, más otras 4 semanas del ensamblado, resultando en un total de 11 semanas de fabricación de una semicolumna superior completa.
- El tiempo de fabricación de una semicolumna inferior es de 5 semanas, más 4 semanas del ensamblado, resultando en un total de 9 semanas de fabricación de una semicolumna inferior completa.
- Por razones relativas a etapas posteriores del proceso, si se cuenta en el taller con todas las piezas de una columna, siempre tendrá prioridad de fabricación la semicolumna superior frente a su homóloga inferior.
- Existen 14 celdas de fabricación, enumeradas de la celda número 1 a la 14.
- Por motivos relativos a la configuración del taller, las 7 primeras celdas no pueden procesar las semicolumnas inferiores de la estructura número 1.
- Debido a los tiempos de cambio, cada vez que se termina el ensamblado de una columna, es necesario esperar tres semanas completas para poder comenzar la fabricación de una nueva.
- En el caso de las celdas 6 y 7, por motivos ajenos al proyecto, no puede empezar a fabricarse hasta la semana 25.

- El material necesario para fabricar las columnas llega por medio de barcos al astillero.
- En ocasiones, el material necesario para la chapa metálica exterior de una semicolumna llega en distintos barcos que el necesario para realizar el ensamblado.
- Generalmente el material necesario para las columnas de menor índice llega antes, no cumpliéndose esta regla a lo largo de todas las semanas.
- En general, no es posible empezar la fabricación de dos semicolumnas la misma semana.
- Por razones relativas a tareas aguas abajo del proceso, siempre tendrán prioridad el comienzo de fabricación de semicolumnas cuya estructura flotante cuente con otras semicolumnas ya acabadas o en proceso. De esta manera, se intenta minimizar al máximo los tiempos de ciclo de las estructuras flotantes.

Sin embargo, en dicha planificación existen 4 singularidades que contradicen en algunas semanas estas restricciones generales:

- En la semana 28 se comienza el ensamblado de la semicolumna C3E5, a pesar de no respetar los tiempos de cambio. No obstante, esta singularidad no se tendrá en cuenta en el desarrollo del algoritmo.
- En algunas semanas coincidentes con la llegada de barcos, no se comienza la fabricación de ninguna semicolumna. Estas son la semana 6, la 30 y la 34.
- En las semanas 25 y 31 no se programa el comienzo de la fabricación de ninguna semicolumna, a pesar de que existe material y no coincide con la llegada de barcos.
- En las semanas 1, 15 y 28 se comienza la fabricación de dos semicolumnas a la vez. En la semana 24, la de 3 simultáneamente.

Con respecto a la llegada de barcos, la tabla siguiente resume las diferentes partes que llegan en cada barco:

Tabla 2. Fechas de llegada de las semicolumnas según el número de barco. Fuente: Elaboración propia.

1	0	1	2	Semicolumna Superior	Exterior	4	5	3	3	Semicolumna Superior	Ensamblado
		1	2	Semicolumna Inferior	Exterior			3	3	Semicolumna Inferior	Ensamblado
		1	3	Semicolumna Superior	Exterior			2	1	Semicolumna Superior	Exterior
		1	3	Semicolumna Inferior	Exterior			4	1	Semicolumna Superior	Exterior
		2	2	Semicolumna Superior	Exterior			3	1	Semicolumna Superior	Exterior
		2	2	Semicolumna Inferior	Exterior			1	1	Semicolumna Superior	Exterior
2	6	1	2	Semicolumna Superior	Ensamblado	5	20	4	3	Semicolumna Superior	Ensamblado
		1	2	Semicolumna Inferior	Ensamblado			4	3	Semicolumna Inferior	Ensamblado
		1	3	Semicolumna Superior	Ensamblado			5	2	Semicolumna Superior	Ensamblado
		1	3	Semicolumna Inferior	Ensamblado			5	2	Semicolumna Inferior	Ensamblado
		2	2	Semicolumna Superior	Ensamblado			2	1	Semicolumna Inferior	Exterior
		2	2	Semicolumna Inferior	Ensamblado			3	1	Semicolumna Inferior	Exterior
		3	2	Semicolumna Superior	Exterior			1	1	Semicolumna Inferior	Exterior
		3	2	Semicolumna Inferior	Exterior			5	1	Semicolumna Inferior	Exterior
		2	3	Semicolumna Superior	Exterior			4	1	Semicolumna Inferior	Exterior
		2	3	Semicolumna Inferior	Exterior			5	1	Semicolumna Superior	Exterior
		3	3	Semicolumna Superior	Exterior	6	27	1	1	Semicolumna Inferior	Ensamblado
		3	3	Semicolumna Inferior	Exterior			1	1	Semicolumna Superior	Ensamblado
3	11	4	3	Semicolumna Superior	Exterior	7	30	5	3	Semicolumna Inferior	Ensamblado
		4	3	Semicolumna Inferior	Exterior			5	3	Semicolumna Superior	Ensamblado
		5	2	Semicolumna Superior	Exterior			2	1	Semicolumna Inferior	Ensamblado
		5	2	Semicolumna Inferior	Exterior			2	1	Semicolumna Superior	Ensamblado
		2	3	Semicolumna Superior	Ensamblado	8	34	5	1	Semicolumna Inferior	Ensamblado
		2	3	Semicolumna Inferior	Ensamblado			5	1	Semicolumna Superior	Ensamblado
4	15	5	3	Semicolumna Superior	Exterior			4	1	Semicolumna Inferior	Ensamblado
		5	3	Semicolumna Inferior	Exterior			4	1	Semicolumna Superior	Ensamblado
		3	2	Semicolumna Superior	Ensamblado			3	1	Semicolumna Inferior	Ensamblado
		3	2	Semicolumna Inferior	Ensamblado			3	1	Semicolumna Superior	Ensamblado

Finalmente, con respecto a la planificación, pese a que no es constante a lo largo de las 45 semanas, se intentado seguir en todo momento la regla de planificar lo más tarde posible, algo que se tendrá en cuenta a la hora de modelizar el caso.

4.4.2 Objetivos y recursos 3D

Una vez descrito y modelizado el caso, se proponen dos objetivos que permitan demostrar las capacidades del simulador:

- Desarrollo de un modelo con Unity3D en el cual, a partir de la lectura de archivo de los diferentes parámetros de cada semicolumna, programe la llegada de estas tal y como se contempla en el plan de mitigación.
- Desarrollo e implementación de un algoritmo de planificación de tareas capaz de realizar una programación que cumpla con todas las restricciones existentes, de manera que pueda ser comparada con la del plan de mitigación.

De esta manera, la idea es que podamos contrastar dos aspectos:

- hasta qué punto la estrategia planificada es más o menos adecuada y,
- posteriormente, comprobar el impacto de alguna modificación introducida en el modelo en los tiempos de fabricación. Todo ello, dentro del simulador mientras la simulación está corriendo

En este sentido, con respecto a la visualización de la simulación, se ha contado de nuevo con la colaboración de la Unidad Mixta de Investigación, la cual ha proporcionado parte de los recursos 3D. El primero de ellos son los modelos de las semicolumnas, mostrados en la siguiente figura:

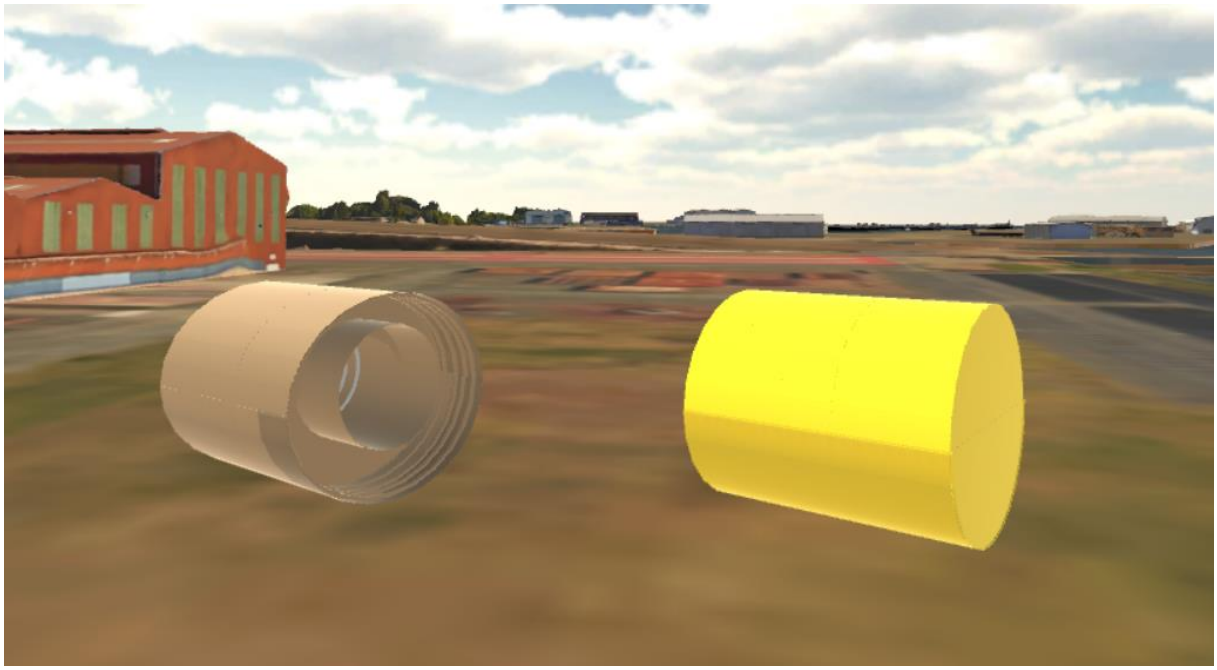

Figura 40. Modelos 3D de las columnas del caso de demostración proporcionadas por la UMI. Fuente: Elaboración propia.

Por otra parte, modelo 3D del astillero también es proporcionado por la Unidad Mixta de Investigación – Figura 41.

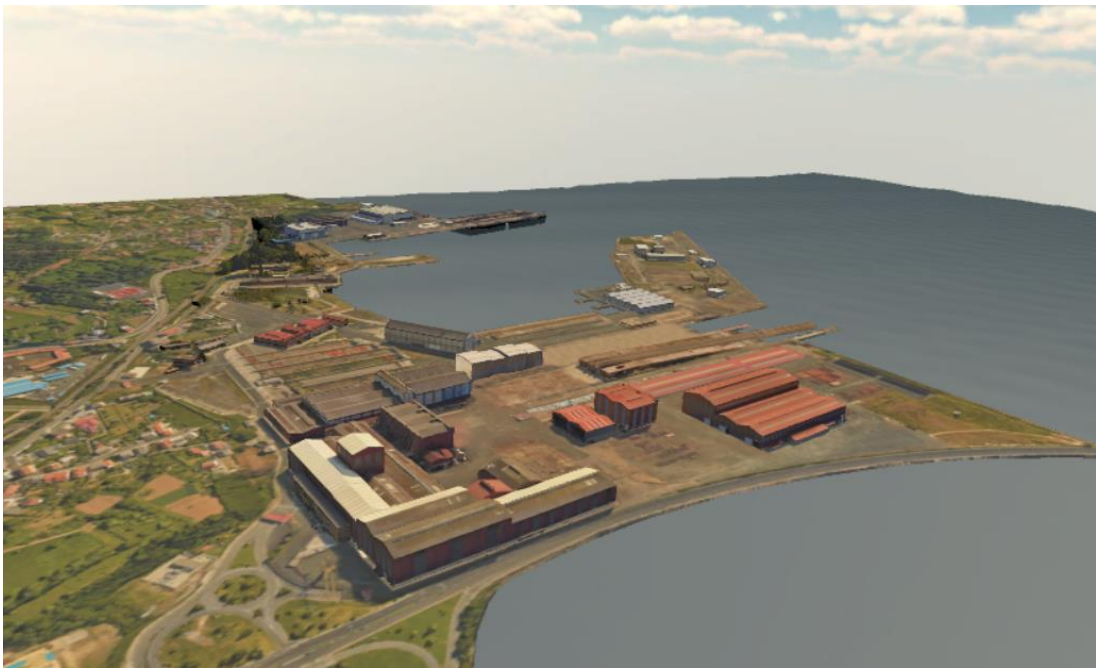

Figura 41. Modelo 3D del astillero del caso de demostración proporcionado por la UMI. Fuente: Elaboración propia.

Finalmente, con respecto a las celdas de fabricación, se emplea un modelo genérico representativo de las mismas. En la Figura 42 se muestran las 14 celdas de fabricación del caos sobre su teórico emplazamiento dentro de un taller del astillero.

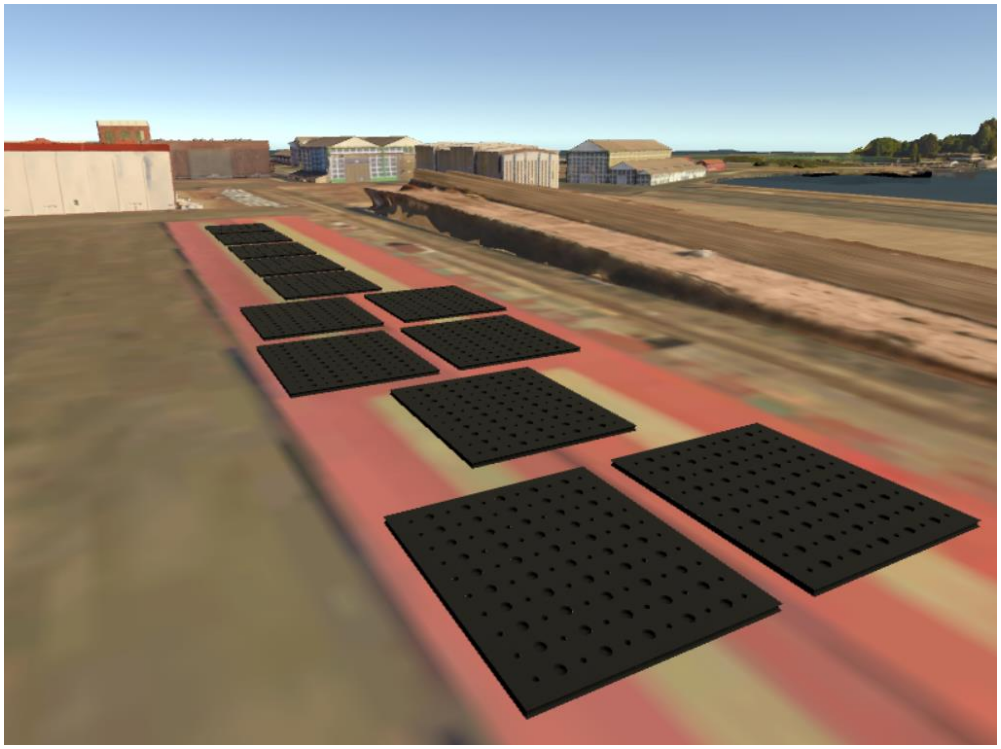

Figura 42. Modelo 3D empleado para representar las celdas de fabricación de las columnas, sobre la zona del taller donde se hayan en la realidad. Fuente: Elaboración propia.

4.4.3 Desarrollo del algoritmo de programación de tareas

Antes de entrar a valorar los resultados del caso, es importante destacar la creación del algoritmo de programación de tareas desarrollado en este caso, a través de una clase de C# denominada «SortingAlgorithm». Una vez extraídas las restricciones, se necesita un código que ordene cada una de las semicolumnas según diferentes parámetros a evaluar, como es el tipo de semicolumna, o si ya existen semicolumnas de la misma estructura flotante en proceso. En este sentido, para el desarrollo del código, se plantea un sistema de ordenación mediante distintas listas, ordenando cada una de ellas los ítems en base a un parámetro determinado. Finalmente, en función de la posición de cada semicolumna en dichas listas, se le asigna una puntuación total que es suma de la puntuación de ese elemento en cada una de ellas. La columna con mayor puntuación será la programada.

Las listas de puntuación contempladas en el desarrollo del algoritmo son las siguientes:

- *byColumnPriority*: almacena la semicolumna en caso de que su homóloga superior o inferior ya fuese programada.
- *byLowerPriority*: sirve para almacenar las semicolumnas inferiores cuyas respectivas semicolumnas superiores ya están en proceso.
- *bySemiColumnPriority*: da prioridad a las semicolumnas superiores frente a sus homólogas inferiores, tal y como se contempla en las restricciones del proceso.
- *byTime*: ordena las semicolumnas según los tiempos de fabricación, incluido el ensamblado posterior.
- *byIndex*: ordena las semicolumnas según el número de la estructura flotante a la que pertenezca. Esto es debido a que, teniendo en cuenta la llegada de material en barcos, interesa fabricar primeramente las columnas de estructuras flotantes de índices más bajos.
- *byStructurePriority*: ordena las semicolumnas según el número de semicolumnas de la misma estructura flotante que hayan sido ya programadas. Este orden responde al objetivo de minimizar los tiempos de ciclo de las estructuras.

De esta manera, las tres primeras listas sirven únicamente para almacenar objetos en algunos casos, frente a las otras tres que ordenan todas las semicolumnas disponibles cada semana en función de algún parámetro.

Por tanto, el funcionamiento del algoritmo es el siguiente:

- Al comienzo de la simulación, existe una lista de ítems pendientes que almacena todas las semicolumnas.
- Cada vez que la clase «ScheduleSource» ha de programar un ítem, pide a «SortingAlgorithm» que le envíe el ítem que ha de programar.
- Inmediatamente, se actualiza una segunda lista que almacena aquellas semicolumnas que ya están disponibles o, en otras palabras, que su barco ya ha llegado. En caso de no existir disponibles, se añaden las siguientes más cercanas en día de llegada.
- Se almacenan y ordenan las listas anteriores de puntuación con las semicolumnas disponibles. Con respecto a los posibles empates en las listas *byTime, byIndex* y *byStructurePriority* se genera un vector de dos dimensiones para cada lista, donde cada fila es un valor nuevo del parámetro. De esta manera, dos semicolumnas con la misma duración se almacenarán en la misma fila, por ejemplo.
- Con respecto a las tres primeras listas, la semicolumna a valorar recibe un punto por cada lista en la que se encuentre.
- En el caso de las listas ordenadas, la semicolumna recibirá tantos puntos de acuerdo a la siguiente operación: número de elementos en la lista - posición en la lista - 1.
- Cada vez que se puntúa una semicolumna, se añade a una lista de resultados que ordena cada ítem entrante según la puntuación asignada.

- Finalmente, se intenta programar el ítem con mayor puntuación, comprobando previamente si existe celdas disponibles, y si dentro de las disponibles, hay alguna válida según el tipo de semicolumna. En caso negativo, se prueba a programar el siguiente ítem.

4.4.4 Resultados y experimentación

En lo que se refiere a los resultados, en primer lugar se consigue exitosamente el primero de los objetivos, que es imitar la planificación de la **¡Error! No se encuentra el origen de la referencia.** a partir de la lectura de datos de archivo. Este objetivo, si bien no se muestran los resultados al ser idénticos a los de dicha tabla, sirvió como base para identificar aquellos parámetros de las necesarios para el desarrollo del algoritmo.

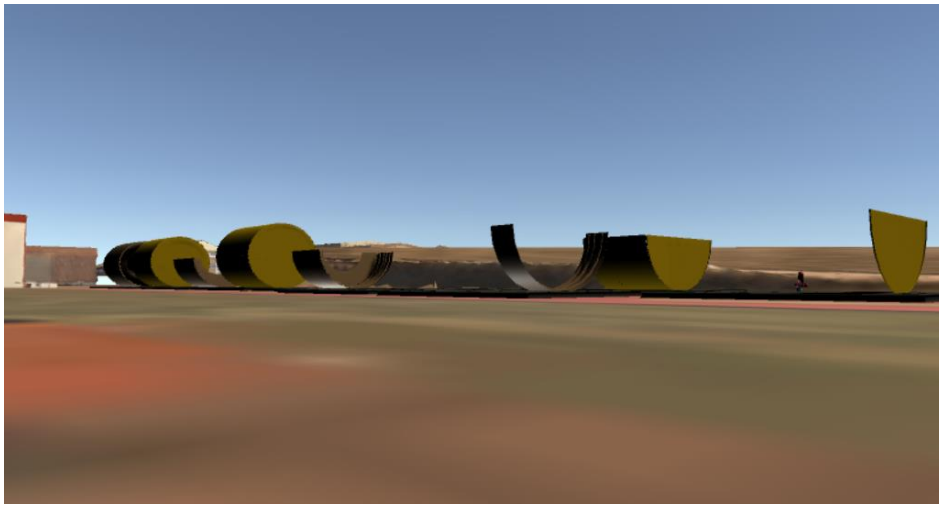

Figura 43. Detalle de la perspectiva del usuario en el mundo virtual mientras se fabrican las semicolumnas. Fuente: elaboración propia.

En segundo lugar, una vez desarrollado el algoritmo, se procede a la validación del mismo asignando las prioridades de manera que cumpla con las restricciones mencionadas en 4.4.1 Descripción del caso. El resultado es mostrado al usuario en el mundo virtual a través del siguiente panel:

Figura 44. Detalle de panel final con la duración de la fabricación de las semicolumnas. Fuente: Elaboración propia.

Una vez finaliza la simulación, Unity crea un archivo de formato txt con de 14 filas y 50 columnas, en correspondencia con las 14 celdas y una duración máxima de 50 columnas respectivamente. Esta *tabla* se va cubriendo con 1's y 0's donde los 1's representan una semana trabajada. Finalmente, los resultados obtenidos en la validación se exportan a Excel y se muestran a continuación en forma de diagrama de Gantt para un mejor entendimiento:

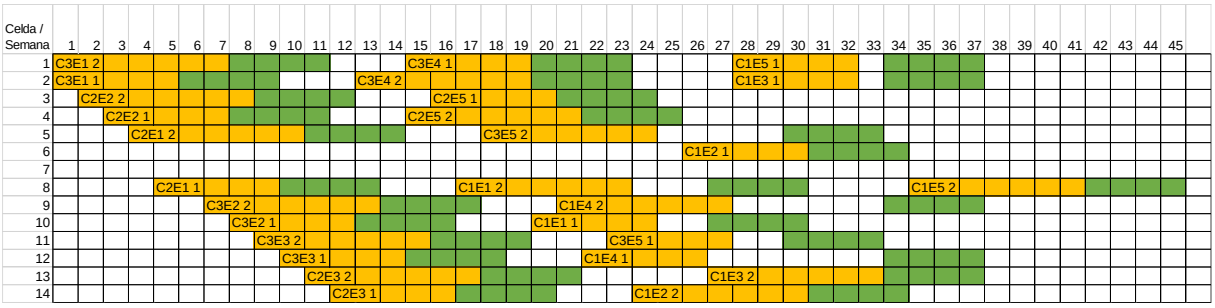

Figura 45. Diagrama de Gantt con los tiempos de planificación generados por el algoritmo. Fuente: Elaboración propia.

Es observable como el orden que aplica el algoritmo con las prioridades tal y como se definieron al principio da lugar a un tiempo total de fabricación de 45 semanas. No obstante, hay que tener en cuenta que la planificación real, existe una semana en la que no se respeta la restricción de los tiempos de cambio, singularidad que no es incluida en el algoritmo. Además, es comprobable como en la planificación generada por el algoritmo se cumplen todas y cada una de las restricciones, incluidos los tiempos de los barcos. Por tanto, se valida el algoritmo.

En tercer y último lugar, una vez validado el algoritmo, se propone experimentar con el mismo de dos formas distintas:

- Por un lado, se varían las ponderaciones a la hora de calcular las puntuaciones, modificando así las distintas prioridades a la hora de programar las tareas.
- Por otra parte, se bloquea una estación de trabajo a un determinado punto de la simulación, de manera que se compruebe el impacto de esta acción en el resultado final.

Con respecto a la primera prueba, se cambian las prioridades de manera que no influya si la semicolumna que se va a procesar corresponde a otra que ya ha entrado en proceso. Y junto con ello, se bonifica mucho más a las semicolumnas superiores. Esto se consigue introduciendo en panel de configuración de «UnityScheduleSource» los parámetros de la siguiente manera:

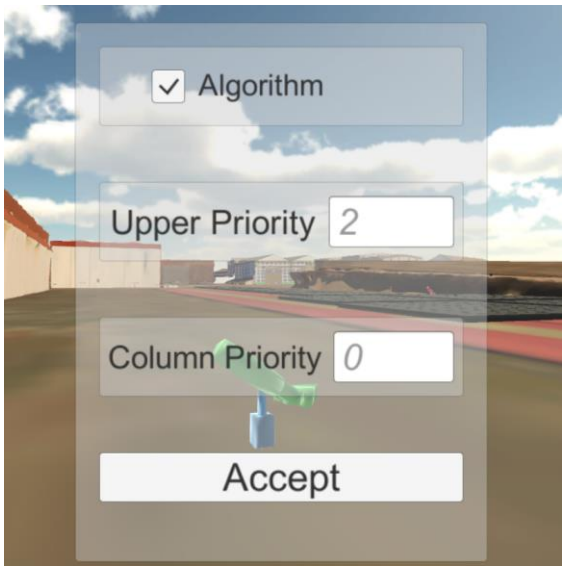

Figura 46. Panel de configuración de un elemento tipo «UnityScheduleSource» con los nuevos parámetros del algoritmo. Fuente: Elaboración propia.

Con estas modificaciones, se obtienen los siguientes resultados:

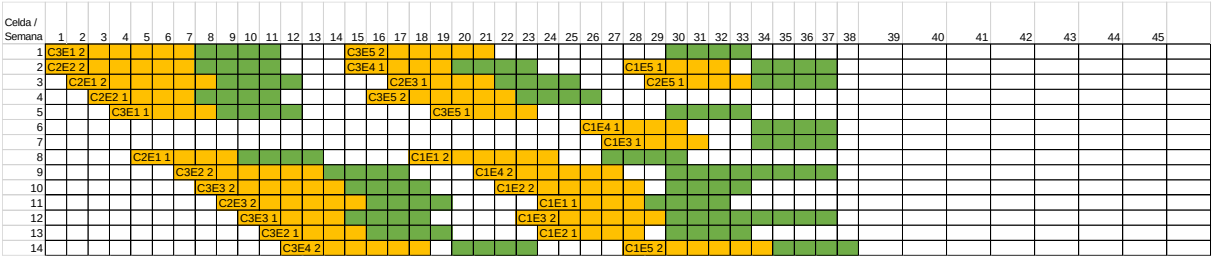

Figura 47. Diagrama de Gantt con los tiempos de planificación cambiando las prioridades del algoritmo. Fuente: Elaboración propia.

En este caso, al eliminar algunas de las restricciones y cambiar las prioridades, se observa como el tiempo total de fabricación se reduce a 38 semanas, pero el algoritmo sigue cumpliendo con el resto de restricciones consideradas.

Finalmente, con respecto al segundo de los experimentos, se trata de introducir un cambio en el modelo con el simulador mientras este está corriendo, para comprobar el efecto en los resultados finales. Para ello, se parte del caso de demostración, y se bloquean dos estaciones al azar: en este caso se ha decidido bloquear la estación 8 y 9 a la semana 7 de proyecto. Este bloqueo se realiza dentro del propio mundo virtual a través del panel de configuración de la estación de trabajo, que incluye un botón que permite bloquearla − Figura 48. En caso de haber alguna semicolumna en proceso, se acaba la fabricación de esta y la estación queda bloqueada hasta que el usuario la desbloquee de nuevo.

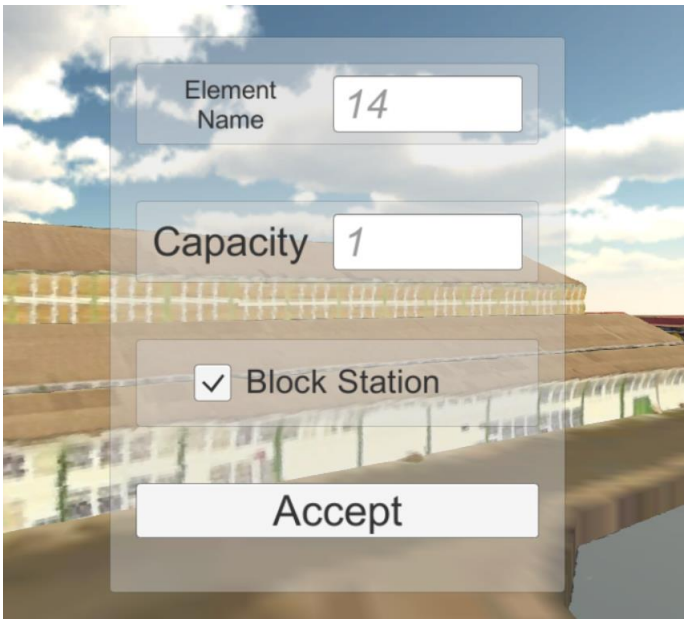

Figura 48. Detalle del panel de configuración de la estación de trabajo, con el botón «Block Station» que permite bloquear la estación seleccionada. Fuente: Elaboración propia.

Con todo, los resultados obtenidos son los siguientes:

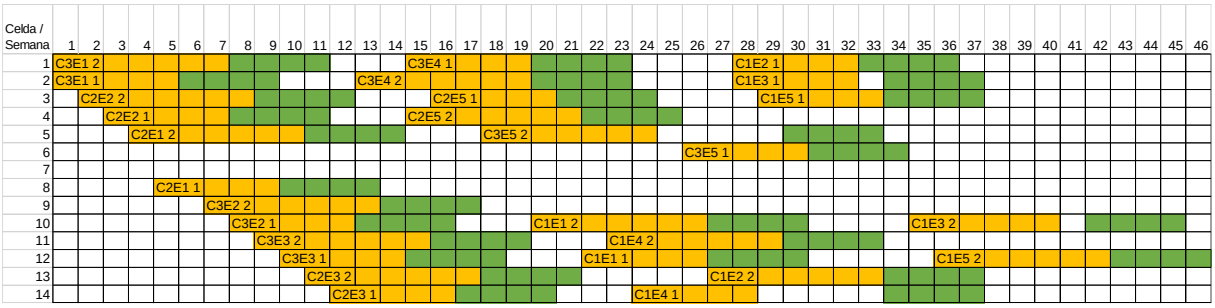

Figura 49. Diagrama de Gantt con los tiempos de planificación bloqueando las estaciones 8 y 9 a la semana 7. Fuente: Elaboración propia.

Es observable como a partir de la 7 no se vuelve a programar trabajos en ninguna de las dos estaciones bloqueadas, respectando así la restricción introducida por el usuario. En este caso, el tiempo sube solamente una semana, a pesar de contar con dos estaciones más, lo cual puede sugerir que el taller está sobredimensionado en cierta medida.

Por tanto, si bien se llevaron a cabo algunos experimentos más, como desbloquear las estaciones 6 y 7 desde el principio – lo cual rebajaba el tiempo de fabricación a 38 semanas −, no es objeto de este trabajo optimizar la programación del caso. No obstante, el propósito real ha sido mostrar el funcionamiento del algoritmo y demostrar como desde el propio simulador podemos evaluar escenarios sencillos, mediante pequeñas modificaciones en el modelo y al alcance de cualquier persona.

Capítulo 5

Una vez desarrollados el simulador y el caso de aplicación, en el presente capítulo se exponen las conclusiones del trabajo realizado, analizando el potencial que puede tener dicho simulador dentro del campo de la simulación de eventos discretos, y con respecto a los *softwares* comerciales de simulación actuales.

Finalmente, se plantean asimismo las posibles líneas de trabajo futuro respecto al desarrollo del simulador.

5.1 Conclusiones

En primer lugar, en este trabajo se ha completado la creación de un simulador de eventos discretos basado exclusivamente en realidad virtual. Este simulador permite crear y configurar pequeños modelos de simulación con una librería limitada de elementos desde el propio mundo virtual, sin necesidad de recurrir a otros periféricos de salida como el teclado o el ratón. Una vez construidos, pueden ser simulados desde el propio mundo virtual, permitiéndole al usuario tener una experiencia totalmente inmersiva, así como control absoluto sobre el comportamiento de los diferentes elementos antes, durante y después de la simulación, con la posibilidad de modificarlos en cualquier momento.

Para ello, primeramente, se han ampliado las librerías de partida para integrar nuevos elementos de simulación que expanden las posibilidades del simulador, como el operador o la carretilla elevadora. Esta fase también ha permitido experimentar con otras capacidades de Unity, explorando así nuevos modos de desarrollo.

A continuación, se diseñó y creó una interfaz gráfica de usuario que es la que permite al usuario crear, configurar y simular el modelo, por lo que su desarrollo supone una parte central del proyecto. En esta fase se revisaron las principales características de la interacción usuario-mundo virtual de los videojuegos de la actualidad, con el fin de integrar algunas de ellas para ofrecer una experiencia de gran calidad.

Una vez creado el simulador, y tras una fase de ensayos con varios participantes con conocimientos en simulación de eventos discretos, se desarrolló un caso de demostración basado en un proyecto real para validar la herramienta y analizar sus potencialidades. En esta fase, fue necesario añadir un hito no definido previamente como fue el desarrollo del algoritmo de programación de las tareas concerniente al caso elegido. Mediante este algoritmo lo que se pretendió fue acercar más el proyecto a la realidad, de cara a mostrar alguna de sus posibles aplicaciones a corto plazo.

Finalmente, se demuestra de nuevo la superioridad gráfica del *software* de videojuegos con respecto a cualquier simulador de eventos discretos, lo cual ratifica que estos programas siguen teniendo un largo camino por recorrer en lo que se refiere a la optimización de su motor gráfico. Por otra parte, en base al trabajo desarrollado y a la experiencia de los participantes en la fase de experimentación, ha sido posible extraer las siguientes conclusiones:

- En lo que se refiere a la experiencia en realidad virtual, la superioridad gráfica del motor de videojuegos le confiere una sensación mucho mayor de inmersión con respecto a cualquier simulador. Esta es una de las conclusiones más claras extraídas de la encuesta, donde algunos participantes calificaron la experiencia de asombrosa con respecto a la sensación de estar en el mundo virtual. Sin embargo, también experimentaron cierta sensación de mareo en momentos puntuales, lo cual puede llegar a ser muy molesto para el usuario. En este sentido, cabe decir que el movimiento continuo a través de los joysticks puede tener cierta responsabilidad, al hacerle creer al cerebro que está caminando cuando realmente uno está quieto. Por ello, cabe considerar otras formas de movimiento como el tele-transporte, que puede resolver este tipo de problemas.

- En cuanto a la interfaz desarrollada, casi todos los participantes en la encuesta están de acuerdo en que resulta amigable e intuitiva, si bien con necesidad de aprender inicialmente el funcionamiento de los controles y el modo de desplazamiento por la escena. En este sentido, cabe subrayar las buenas opiniones respecto a la estructura de los menús, y especialmente en lo relativo a la manejabilidad. La mayoría de los participantes consideran que la interfaz permite modificar fácilmente un modelo, y que además es bastante intuitiva, lo cual es un resultado muy positivo.
- Uno de los principales objetivos de este proyecto es contrastar la validez de una herramienta de estas características para crear y configurar modelos de simulación. En este sentido, y a la vista de los resultados de la encuesta, es posible concluir que más allá de detalles puntuales, no tiene en sentido en la actualidad modelar un caso a través de realidad virtual. Si bien puede tener cierto potencial en el diseño del *layout*, no se aprecian mejoras significativas respecto a la creación de los elementos, la configuración de sus parámetros, o la experimentación con distintos escenarios, pues es mucho más directo y sencillo a través de interfaz habitual.
- No obstante, en lo referido al proceso de toma de decisiones y comunicación de resultados a personal no técnico, la opinión de los encuestados es unánime: un simulador de estas características potenciaría enormemente esta fase de transmisión de resultados y conclusiones, especialmente cuando el receptor no tiene formación en este tipo de simulación. La interacción con el modelo, tal y como se describe en el caso de demostración, puede ayudar en casos de querer convencer a un cliente de que una estrategia adoptada es adecuada, o mostrarle el impacto de un error o retraso en una tarea, por ejemplo.
- Como se viene demostrando desde hace tiempo, no cabe duda alguna del potencial de esta tecnología como herramienta comercial, debido a la impresión positiva que sigue generando la realidad virtual en su público hoy en día. No obstante, se trata de una tecnología que aún no es fácilmente accesible, debido a la necesidad de un poder computacional medio-alto, y al coste de los diferentes dispositivos que existen en el mercado. Esto implica que en general siga siendo algo novedoso en la actualidad, lo que provoca esas reacciones de impresión. Por tanto, se sugiere aquí que posiblemente en unos años se establezca como una tecnología mucho más corriente, lo que restaría interés y potencial como herramienta para fines puramente comerciales.

Como conclusión final, cabe decir que en este trabajo supone un salto cualitativo respecto al desarrollado en (Pernas Álvarez, 2017), y permite revocar una de sus conclusiones. Un simulador VR ya no es puramente una herramienta comercial, sino que abre puertas a nuevos aspectos de la simulación, y a acelerar procesos existentes en cualquier empresa como es la toma de conclusiones, o el entendimiento de un modelo complejo. Si bien considero que aún queda mucho por desarrollar, y que quizás no muchos crean en el potencial de esta herramienta a día de hoy, lo que se ha perseguido aquí es crear algo nuevo, sin nada existente que lo imite, para explorar las posibilidades que ello podría aportar. Así, teniendo en cuenta el avance incesante de la tecnología y, especialmente, en los campos de realidad aumentada y virtual, cabe pensar que algún día los modelos se desarrollen puramente en el mundo virtual, con una interfaz muy avanzada que mejore incluso los tiempos del modelado tradicional.

5.2 Trabajos futuros

En último lugar, a la vista del trabajo desarrollado, así como durante el proceso de creación del simulador, se han pensado como posibles líneas de trabajo futuro las citadas a continuación:

- Respecto a la interfaz gráfica, explorar nuevas formas de interacción como el tele-transporte por la escena, o la posibilidad de agarrar objetos o interactuar físicamente con ellos.
- Ampliar las funcionalidades del simulador, desde la posibilidad de modificar las dimensiones de los objetos creados, a la simulación de varios escenarios en paralelo.
- Una línea muy interesante sería la posibilidad de incorporar varios sujetos en la misma escena, teniendo varios participantes en el mismo mundo virtual, desde donde poder explicar algún aspecto del caso, por ejemplo.
- La combinación de un modelo en realidad virtual con el mundo real a través de la realidad aumentada– realidad mixta – abriría nuevas posibilidades respecto, por ejemplo, al diseño del *layout*, pudiendo contrastar en la realidad la localización exacta de una máquina, o la ruta que ha de seguir un operador.
- Finalmente, aunque se trate de un objetivo muy ambicioso, este proyecto se enmarca dentro del concepto de gemelo virtual de la fábrica, en el campo de la fabricación inteligente. Por tanto, poder conectar el modelo con la realidad, de manera que actualice en tiempo real los movimientos y los tiempos de fabricación, permitiría poder controlar nuestra propia planta desde cualquier punto del mundo, mientras se evalúa el impacto de posibles riesgos durante el proceso de fabricación.

Capítulo 6

6.1 Bibliografía

Abidi, M., Chevaillier, P., Lyonnet, B., Kechiche, M., Baert, P., & Toscano, R. (2015). How to create a new generation of industrial processes simulation by coupling simulation tools with VR platforms. *28th International Conference on Computer Applications in Industry and Engineering, CAINE 2015*. Retrieved from https://www.researchgate.net/profile/Patrick_Baert/publication/282572338_How_to_Create_a_New_Generation_of_Industrial_Processes_Simulation_by_Coupling_Simulation_Tools_with_VR_Platforms/links/5a269e4a4585155dd423e838/How-to-Create-a-New-Generation-of-Indu

Akpan, J.I., & Brooks, J. R. (2005). Practitioners' perception of the impacts of virtual reality on discrete-event simulation. *Proceedings of the 2005 Winter Simulation Conference : Hilton at the Walt Disney World Resort, Orlando, Florida, U.S.A., Dec 4-7, 2005*, 1976–1984. Retrieved from https://dl.acm.org/citation.cfm?id=1163050

Akpan, Justice I., & Brooks, R. J. (2005). Experimental investigation of the impacts of virtual reality on discrete-event simulation. *Proceedings - Winter Simulation Conference*, *2005*, 1968–1975. https://doi.org/10.1109/WSC.2005.1574475

AnyLogic. (2017). Simulation Software Comparison: Discrete Event Simulation Competitors. Retrieved May 6, 2019, from https://www.anylogic.com/resources/white-papers/simulation-software-comparison/

Deloitte. (2017). The smart factory. *The Smart Factory Responsive, Adaptive, Connected Manufacturing*. Retrieved from http://media.daimler.com/marsMediaSite/en/instance/ko.xhtml?oid=9905147

Descreye Solutions. (2018). A Comparison of Discrete Event Simulation Software. Retrieved May 6, 2019, from https://www.descreye.com/blog/a-comparison-of-discrete-event-simulation-software/

Dorozhkin, D. V., Vance, J. M., Rehn, G. D., & Lemessi, M. (2012). Coupling of interactive manufacturing operations simulation and immersive virtual reality. *Virtual Reality*, *16*(1), 15–23. https://doi.org/10.1007/s10055-010-0165-7

IT-Enterprise. (2019). Smart Factory - Factory of "Industry 4.0." Retrieved May 2, 2019, from https://it-enterprise.com/knowledge-base/technology-innovation/smart-factory

Jain, S., Lechevalier, D., Woo, J., & Shin, S.-J. (2015). Towards a virtual factory prototype. *Proceedings of the 2015 Winter Simulation Conference*, 2207–2218. Retrieved from https://dl.acm.org/citation.cfm?id=2888870

Jain, S., & Shao, G. (2015). Virtual factory revisited for manufacturing data analytics. *Proceedings - Winter Simulation Conference*, *2015-Janua*, 887–898. https://doi.org/10.1109/WSC.2014.7019949

Kagermann, H., Wahlster, W., & Helbig, J. (2013). Recommendations for implementing the strategic initiative Industrie 4.0. In *ACATEC-National Academy of Science and Engineering*. Retrieved from https://web.archive.org/web/20131014183202/http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf

Kelsick, J., Vance, J. M., Buhr, L., & Moller, C. (2003). Discrete Event Simulation Implemented in a Virtual Environment. *Journal of Mechanical Design*, *125*(3), 428. https://doi.org/10.1115/1.1587745

Kusiak, A. (2018). Smart manufacturing. *International Journal of Production Research*, *56*(1–2), 508–517. https://doi.org/10.1080/00207543.2017.1351644

Mujber, T. S., Szecsi, T., & Hashmi, M. S. J. (2004). Virtual reality applications in manufacturing process simulation. *Journal of Materials Processing Technology*, *155–156*, 1834–1838. https://doi.org/10.1016/J.JMATPROTEC.2004.04.401

Nee, A. Y. C., & Ong, S. K. (2013). Virtual and Augmented Reality Applications in Manufacturing. *IFAC Proceedings Volumes*, *46*(9), 15–26. https://doi.org/10.3182/20130619-3-RU-3018.00637

Pérez Martínez, F. J. (2011). Presente y Futuro de la Tecnología de la Realidad Virtual. *Creatividad y Sociedad: Revista de La Asociación Para La Creatividad*, (16), 3–39. Retrieved from http://www.creatividadysociedad.com/articulos/16/4-Realidad Virtual.pdf

Pernas Álvarez, J. (2017). *Desarrollo de un modelo de simulación para frabricación basado en Unity3D*. Retrieved from https://ruc.udc.es/dspace/handle/2183/19731

Qi, Q., & Tao, F. (2018). Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison. *IEEE Access*, *6*, 3585–3593. https://doi.org/10.1109/ACCESS.2018.2793265

Rekapalli, P. V., & Martinez, J. C. (2010). Discrete-Event Simulation-Based Virtual Reality Environments for Construction Operations: Technology Introduction. *Journal of Construction Engineering and Management*, *137*(3), 214–224. https://doi.org/10.1061/(asce)co.1943-7862.0000270

Robinson, S. (2005). Discrete-event simulation: From the pioneers to the present, what next? *Journal of the Operational Research Society*, *56*(6), 619–629. https://doi.org/10.1057/palgrave.jors.2601864

Rodic, B. (2017). Industry 4.0 and the New Simulation Modelling Paradigm. *ORGANIZACIJA*, *50*(3), 193–207. https://doi.org/10.1515/orga-2017-0017

Rüßmann, M., Lorenz, M., Gerbert, P., & Waldner, M. (2015). Industry 4.0: The future of productivity and growth in manufacturing industries. *Boston Consulting*. Retrieved from http://www.inovasyon.org/pdf/bcg.perspectives_Industry.4.0_2015.pdf

Shao, G., Shin, S. J., & Jain, S. (2015). Data analytics using simulation for smart manufacturing. *Proceedings - Winter Simulation Conference*, *2015-Janua*, 2192–2203. https://doi.org/10.1109/WSC.2014.7020063

Simio LLC. (2016). Press Release. Retrieved May 7, 2019, from https://www.simio.com/resources/news-releases/2016/Simio8-136/index.php

Söderberg, R., Wärmefjord, K., Carlson, J. S., & Lindkvist, L. (2017). Toward a Digital Twin for real-time geometry assurance in individualized production. *CIRP Annals - Manufacturing Technology*, *66*(1), 137–140. https://doi.org/10.1016/j.cirp.2017.04.038

Swain, J. J. (2015). Simulation Software Survey - Simulated Worlds. *OR/MS Today*, *42*(5). Retrieved from https://www.informs.org/ORMS-Today/OR-MS-Today-Software-Surveys/Simulation-Software-Survey

Turner, C. J., Hutabarat, W., Oyekan, J., & Tiwari, A. (2016). Discrete Event Simulation and Virtual Reality Use in Industry: New Opportunities and Future Trends. *IEEE Transactions on Human-Machine Systems*, *46*(6), 882–894. https://doi.org/10.1109/THMS.2016.2596099

Wee, D. (2015). Manufacturing ' s next act | McKinsey & Company. *Timereaction.Com*, 1–3. Retrieved from https://www.timereaction.com/papers/manufacturing_next_act.pdf

Xu, J., Huang, E., Hsieh, L., Lee, L. H., Jia, Q. S., & Chen, C. H. (2016). Simulation optimization in the era of Industrial 4.0 and the Industrial Internet. *Journal of Simulation*, *10*(4), 310–

320. https://doi.org/10.1057/s41273-016-0037-6