Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Simulación en Unity 3D mediante modelos GOAP de personas de movilidad reducida en aeropuertos

Simulation in Unity 3D using GOAP models of people with reduced mobility at airports.

La Laguna, 10 de agosto de 2021

D. **Iván Castilla Rodríguez**, con N.I.F. 78.565.451-G profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Rafael Arnay del Arco**, con N.I.F. 78.569.591-G profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Simulación en Unity 3D mediante modelos GOAP de personas de movilidad reducida en aeropuertos"

ha sido realizada bajo su dirección por D. **Christian Torres González**,

con N.I.F. 43.841.669-N.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de agosto de 2021

Agradecimientos

En primer lugar, agradecer a ambos profesores, encargados de la tutorización de mi Trabajo de Fin de Grado, Iván Castilla y Rafael Arnay, que han sido los principales pilares en los que me he apoyado ante cualquier dificultad o problemática encontrada durante la realización de este además de estar disponible en todo momento para siempre prestar la máxima ayuda posible

Ajeno al trabajo, dado que es el último trabajo de este paso por la universidad agradecer a todos los profesores, que han aportado su grano de arena, a que este momento haya sido posible, que haya llegado este punto de terminar el último trabajo de la carrera

Licencia

Resumen

De todos los cientos de millones de pasajeros que pasan por los aeropuertos anualmente, solo un 1% son personas de movilidad reducida.

Pese a que representen solo el 1%, a nivel operativo, presentan una dificultad para los aeropuertos debido a que suponen una serie de costes y recursos adicionales, tales como que el aeropuerto disponga de asistentes para ayudarlos, sillas de ruedas donde transportar a estos pasajeros, dispositivos ambulifts con los que subirlos al avión, y otra serie de recursos que dificultan las operaciones en el aeropuerto.

Para intentar aportar una solución a esa situación, el objetivo de este TFG ha sido el desarrollo de una simulación en Unity 3D, que permita a los aeropuertos llevar a cabo el estudio de diferentes estrategias con las que poder encontrar mejores soluciones en cuestión de recursos económicos y materiales, adaptando los parámetros de esta simulación, a la infraestructura y recursos de dicho aeropuerto.

Abstract

Of all the hundreds of millions of passengers who pass through airports annually, only 1% of all these are cases attended by people with reduced mobility.

Although they represent only 1%, at operational level, presents a difficulty for airports because they entail a series of additional costs and resources, such as the airport having assistants to help them, wheelchairs to transport passengers, ambulifts with which to get passengers on the plane, and other resources that hinder operations at the airport.

To try to provide a solution to this problem, the objective of this TFG has been the development of a simulation in Unity 3D, to enable airports to carry out the study of different strategies with which to find better solutions in terms of economic and material resources, adapting the parameters of this simulation, to the infrastructure and resources of the airport.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Introducción

Los aeropuertos son una de las infraestructuras por donde más personas pasan de manera diaria, aproximadamente una media de un millón de personas por los aeropuertos españoles en la época de verano. Es por esto por lo que los aeropuertos tienen que estar preparados para gestionar esta afluencia de gente de manera constante.

Dentro de este millón de personas, podemos encontrar diferentes tipos de pasajeros, entre los cuales destacan los llamados pasajeros de movilidad reducida (PMR). El principal motivo por el que destaca este grupo es que requieren de una atención especial durante su paso por el aeropuerto, dada la necesidad de un asistente para poder moverse por él.

Para estudiar en detalle los problemas que existen detrás de la gestión de PMR en un aeropuerto, tenemos que dividir en tres el problema según el proceso en el que se vean involucrados:

- **Aeropuerto de salida:** los pasajeros llegan al punto de encuentro donde se reunirán con el asistente, quien les guiará en tareas de facturación, desplazamiento de equipaje de mano, soporte en controles de seguridad, escolta hacia la zona de embarque y finalmente entrada en la aeronave y acceso al asiento designado.
- **Aeropuerto de llegada**: esta fase comienza en el propio avión, una vez se encuentra en el punto de desembarque, empezando en primer lugar por el recorrido que va desde el asiento en el que se encuentra el pasajero hasta la puerta del avión. A partir de este punto, el asistente guía al pasajero en tareas tales como el desembarco del avión, viaje a la sala de equipajes, al igual que en la primera etapa y el apoyo en los controles de seguridad.
- **Aeropuertos de tránsito**: esta tercera fase del viaje suele darse cuando por motivos de planificación, el vuelo realiza escalas en aeropuertos intermedios. Dentro de algunas tareas que podemos encontrar, se encuentran algunas tareas de fases anteriores, de embarco y desembarco, traslado entre terminales, etc.

De las fases mencionadas, este trabajo se centrará en la primera: aeropuerto de salida. Desarrollada esta fase, a futuro se podrían desarrollar el resto, ya que los procesos involucrados en ellas se asemejan bastante, de tal forma que la solución se puede exportar al resto de los procesos de cada una de las fases.

1.2 Antecedentes y estado actual

Según datos estadísticos, en los aeropuertos españoles se atienden 1 millón de casos de PMR, lo cual no llega ni al 1% del total de pasajeros que pasan por ellos. Es por eso, que, en la actualidad, una de las situaciones más complicadas a las que se enfrentan los aeropuertos, es la de ofrecer este tipo de servicios de asistencia a estas personas.

Los principales motivos por los que esta situación es considerada como un problema al que se tienen que enfrentar, es debido a que requieren de una serie de recursos materiales, humanos y económicos adicionales, a los que puede disponer un aeropuerto. Entre estos recursos necesarios, destacan:

- Personal de asistencia con el que moverse por el aeropuerto.
- Sillas de ruedas donde poder trasladarse.
- Dispositivos ambulifts con los que poder acceder al avión.
- Puntos de encuentro señalizados donde el PMR pueda indicar la necesidad de asistencia.

Como es evidente, este servicio de ayuda no tiene ningún tipo de coste adicional para el pasajero, pero, por el contrario, es el aeropuerto la unidad encargada de ofrecer este servicio con la mejor calidad posible, de tal manera que el viaje para una persona de movilidad reducida se iguale a las comodidades y ventajas de un pasajero que no presente ningún tipo de limitación de movilidad. Es por este motivo, por el que esta situación es considerada un problema al que se enfrentan los aeropuertos.

Dada esta situación, la Unión Europea, se encuentra en una fase de eliminación de cualquier tipo de impedimento que suponga una restricción para aquellas personas que presentan algún problema de movilidad reducida. Principalmente se ha empezado a trabajar en los aeropuertos de los países pertenecientes a la UE, de tal manera que estos deben empezar a ofrecer una asistencia de calidad a este tipo de casos de PMR, con los recursos y el personal necesario para garantizar que se cumplen sus derechos.

Todo este sistema cuenta con algunos problemas debido al propio funcionamiento de un aeropuerto, situaciones de retrasos, cancelaciones, averías, de tal manera que todo este tipo de inconvenientes dificultan la tarea de facilitar el paso de PMR por los aeropuertos.

Entre los principales motivos de por qué se han utilizado estas tecnologías, encontramos el punto actual en el que se encuentran las IoT (Internet of Things), permitiendo la interconexión entre las diferentes partes interesadas de las operaciones aeroportuarias, además de destacar también por su gran importancia dentro del desarrollo de otras áreas tales como la geolocalización de PMR o el uso de sillas de ruedas autónomas entre otras.

Cabe destacar también, en cuanto a las sillas de ruedas, que actualmente todo este sistema se va implantando de manera progresiva, pasando por algunas fases tales como investigación y valoración del impacto del uso de sillas autónomas a través de estas simulaciones que se van realizando, con el objetivo también de detectar la cantidad óptima de sillas de ruedas a implantar.[9] Aplicado a la problemática de las PMR, a diferencia de las personas que no presentan ninguna

discapacidad las cuales, de manera general se encuentran interconectadas a través del móvil, las PMR podrían estar conectadas a través de alguna aplicación móvil o mediante un dispositivo IoT en formato de tarjeta inteligente.

Por último, quiero añadir que adicionalmente a este TFG, se está llevando a cabo una línea de investigación en la ULL, alrededor de una tesis doctoral dirigida por Iván Castilla, en la que se plantean diferentes soluciones al problema de gestión de PMR en aeropuertos. Una de las claves de esa línea es poder evaluar mediante simulación diferentes soluciones como cambios sobrevenidos o nuevas tecnologías de aplicación para el problema como bien puede ser el IOT (Internet Of Things) junto a las redes 5G y sillas de ruedas autónomas, que suponen una oportunidad única para incorporar nuevos servicios y soluciones.

Capítulo 2
Objetivos

2.1 Objetivo general del proyecto

El objetivo de este TFG es el desarrollo de una simulación en Unity 3D, que permita a los aeropuertos llevar a cabo el estudio de diferentes estrategias con las que poder encontrar mejores soluciones para la gestión de personas de movilidad reducida en cuestión de recursos económicos y materiales, adaptando los parámetros de esta simulación, a la infraestructura y recursos de dicho aeropuerto.

2.2 Objetivos específicos

2.2.1 Representación realista de un aeropuerto en Unity

Para poder llevar a cabo la simulación, se debe de contar con una maqueta real de un aeropuerto, pero Unity no soporta el formato de los archivos en los que está almacenado el aeropuerto. Es por eso por lo que se realizará un estudio de herramientas para poder representar archivos con la extensión deseada en software de diseño y modelado tal como Unity. En primer lugar, dando prioridad a herramientas libres, pero analizando también, cuando sea posible, herramientas comerciales.

2.2.2 Desarrollo de un escenario para la simulación

Se llevarán a cabo dos escenarios para la simulación:

- **Escenario de pruebas:** escenario donde iré desarrollando las bases de la simulación a base de prueba y error.
- **Escenario real:** será el resultado de lograr cargar los archivos IFC en Unity, y será el escenario real de un aeropuerto, donde se llevará a cabo el estudio de un aeropuerto real.

2.2.3 Diseño e implementación de un modelo GOAP

Se desarrollará el modelo GOAP (Goal-Oriented-Action-Planning) [10], el cual consiste en una arquitectura de planificación, diseñada para planificar y controlar en tiempo real, el comportamiento de un conjunto de agentes mediante inteligencia artificial.

2.2.4 Análisis de un caso de estudio sobre modelo GOAP

Se llevará a cabo el estudio de una estrategia que podría seguir un aeropuerto, estudiando y obteniendo datos tales como el número de asistentes óptimos, punto de ubicación de los asistentes. en base a los tiempos obtenidos durante la simulación, además de otras variables a definir para el estudio.

Capítulo 3
Tecnologías

En este apartado procederé a comentar las diferentes tecnologías (software, formato y tipo de archivos) utilizadas a lo largo del proyecto, su motivo de utilización y las ventajas que aportan.

3.1 BIM

Building Information Modeling (BIM), en español, Modelo de Información de Construcción, es un conjunto de procesos y metodologías para la edificación para la generación y gestión de datos de un edificio u obra de ingeniería, utilizando para ello un modelo digital compartido. BIM permite producir y almacenar toda la información necesaria para operar en las distintas fases del ciclo de vida de las construcciones.

Los modelos BIM se obtienen a partir de la creación de objetos con sus propios atributos que representan los elementos constructivos físicos a gestionar o controlar. Los objetos BIM de los modelos virtuales pueden estar conectados a uno o varios registros de una base de datos y permiten almacenar información tal como los espacios construidos, elementos, materiales, medios y recursos implicados en su gestión.

3.2 IFC

Industry Foundation Classes (IFC), es un estándar para el intercambio de datos en la industria de la construcción que permite compartir información independientemente de la aplicación de software que se utilice.

Dentro del mundo del modelado, los archivos con extensión IFC cobran gran importancia debido a la posibilidad que ofrecen para colaborar entre varias figuras involucradas en el proceso de construcción permitiendo el intercambio de información a través de un formato estándar. Esto lleva a una calidad mayor, disminución de errores, reducción de costes y ahorro de tiempo en coherencia de datos a lo largo del proceso de ejecución, utilizado en proyectos basados en BIM.

Enfocado en mi trabajo, esta tecnología, será el formato que tendrán los archivos que contienen la información referente a la estructura del aeropuerto sobre el que voy a desarrollar la simulación.

3.3 Unity 3D

Unity 3D es lo que se conoce como un motor de desarrollo o motor de juegos. El término motor de videojuego o game engine hace referencia a un software el cual tiene una serie de rutinas de programación que permiten el diseño, la creación y el funcionamiento de un entorno interactivo.

Dentro de las principales funcionalidades por las que he utilizado Unity, cabe destacar:

- Motor gráfico para renderizar gráficos 2D y 3D.

- Simulación de leyes de la física.
- Programación y scripting.
- Inteligencia artificial.

En mi caso, dado que el trabajo está enfocado en la realización de una simulación, éste se ha desarrollado en Unity debido a las funcionalidades mencionadas anteriormente, además de su capacidad de trabajar con inteligencia artificial, o en mi caso con modelos GOAP (explicados más adelante) el cual permitirá trabajar con múltiples agentes los cuales cumplirán sus objetivos en función de las condiciones en la que se encuentre la simulación.

Capítulo 4
Desarrollo

4.1 Representación realista de un aeropuerto en Unity 3D

La actividad con la que da comienzo este trabajo es con un estudio de herramientas que permitan y faciliten la representación de archivos con formato IFC (tecnología introducida en el apartado anterior, 3.2 IFC) en herramientas de diseño y modelado como Unity 3D, ya que es el software empleado para el proyecto.

El motivo de este estudio se debe a que como bien he adelantado anteriormente, se va a realizar una serie de simulaciones sobre un aeropuerto el cual ya se encuentra modelado y almacenado en archivos IFC, pero que necesitan ser cargados en el software de trabajo. El problema de estos archivos es que Unity no tiene soporte directo para este formato.

Es por ello por lo que se ha establecido como primera tarea del proyecto, la realización de este estudio, intentando recurrir en última instancia a tener que modelar el aeropuerto evitando la pérdida de toda la información que nos proporcionan los archivos IFC.

Durante este estudio se ha intentado realizar la búsqueda de herramientas priorizando aquellas que son gratuitas para poder cargar los archivos, y en última instancia aquellas que no sean gratis. De tal forma, que a continuación, se procede a comentar las diferentes herramientas encontradas y con las que se ha trabajado.

1. Unity Reflect

Como primera herramienta, se ha trabajado con una que ofrece el propio Unity, llamada Unity Reflect [7].

Al ser desarrollada por la propia empresa, funciona correctamente tal y como se muestra en los manuales y en los vídeos explicativos, pero tiene un inconveniente, y es que, para poder hacer uso de ella, se debe adquirir una licencia. Esta herramienta te da la opción de hacer una prueba gratuita dejando prepagado un plan anual básico con un coste de 690€ o un plan pro cuyo coste asciende a 1800€. Es por eso por lo que en lo que a esta herramienta se refiere, no he podido explorar mucho más su funcionamiento.

Por último, en cuanto a esta herramienta, he de comentar también que ofrece una versión para dispositivos móviles que, a diferencia de la de ordenador, no tienen ningún coste, pero solo te permite visualizar estos archivos sin poder trabajar en ellos, lo cual solo nos sirve para comprobar si los IFC cargados se visualizan correctamente.

2. Tridify

La segunda herramienta que se ha tratado para el estudio de carga de IFC en Unity ha sido Tridify [8].

Esta herramienta, al igual que la anterior, también cuenta con una serie de servicios de pago, pero a diferencia de Unity Reflect, ofrece unas ventajas a aquellas personas que forman parte de una organización perteneciente al mundo de la enseñanza tales como universidades.

En mi caso me he registrado indicando como centro formativo la Universidad de La Laguna, y lo cual me ha reportado la ventaja de tener el plan de acceso a la plataforma un año.

Más allá de tener esta ventaja de año de prueba, uno de los grandes inconvenientes es que el espacio del que se dispone para subir archivos a la cuenta es de 200 Mb, independientemente del plan que tengas, prueba gratuita de 30 días o estudiante (Figura 1 Cuenta universitaria en Tridify).

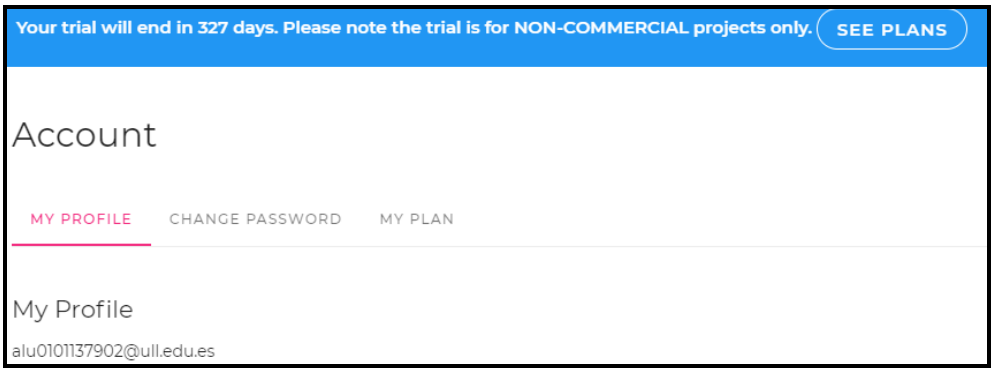


Figura 1 Cuenta universitaria en Tridify

En lo referente a la carga de archivos IFC, en primer lugar, la plataforma te obliga a subir los archivos a tu cuenta en la web, para luego enlazarlos con Unity, mediante un plugin proporcionado por la propia herramienta.

En este punto, considero que la herramienta presenta un gran inconveniente, el cual trata de, en cierta manera, tener que forzarte a subir los archivos a su web, ya que, al hacerlo, estás permitiendo que ellos también tengan acceso a esos archivos. En caso de que sea un proyecto más importante donde debe existir un nivel de confidencialidad, no sería lo más conveniente el usar esta herramienta.

En cuanto al funcionamiento, a la hora de subir los archivos, me he encontrado con otro problema y es que, de los tres archivos necesarios para la representación del aeropuerto, solo se pudieron subir dos con éxito pese a intentarlo varias veces a lo largo de varios días (Figura 2 Carga de archivos en Tridify).

3 IFC files (38 MB total)

Publish	File	Status	Uploaded at ↓
☐	AIRPORT_AIRM_EXAMPLE_v0	✗ Validation failed	23.03.2021 08:06
☐	AIRPORT_AIRM_EXAMPLE_v1	✓ Ready to publish	17.02.2021 15:29
☐	AIRPORT_AIRM_EXAMPLE_v2	✓ Ready to publish	17.02.2021 15:29

Figura 2 Carga de archivos en Tridify

Pese a este problema, he continuado con su uso, y he probado la herramienta dentro de Unity, para ver cómo se observaban los dos archivos que sí se pudieron subir a la web.

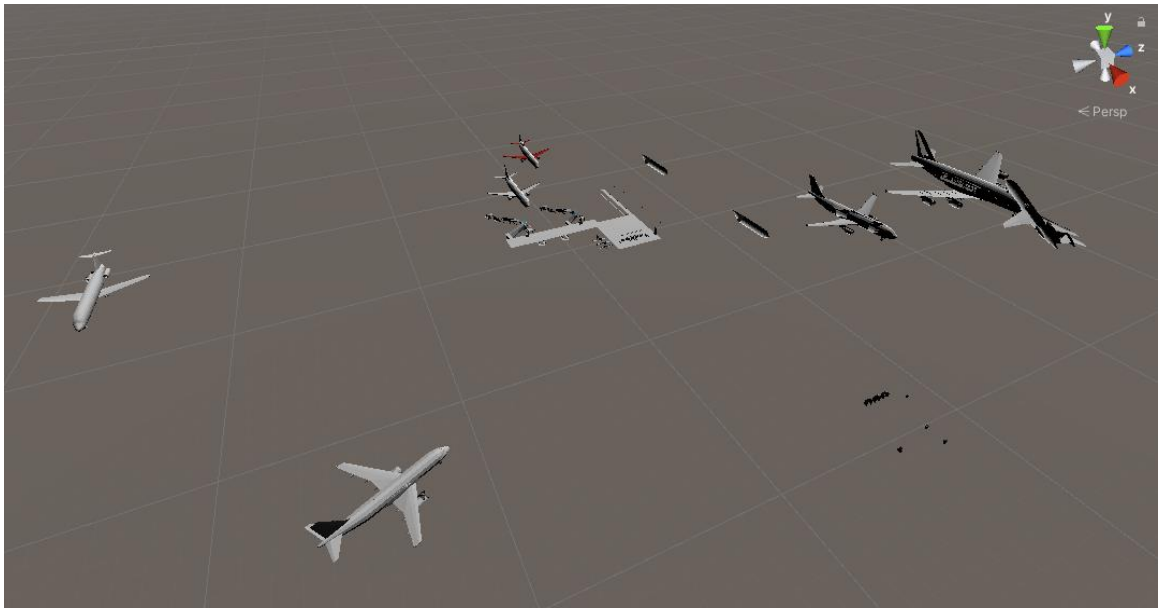


Figura 3 Resultados obtenidos con Tridify

Tal y como se aprecia en Figura 3 Resultados obtenidos con Tridify los resultados obtenidos no han sido nada satisfactorios ya que además de no asemejarse a un aeropuerto, falta un archivo por cargar, por lo que se concluye con esta herramienta, considerándola no exitosa para el estudio.

3. Otras aplicaciones e IfcOpenShell

Tras los intentos fallidos con las herramientas anteriores, durante la búsqueda de nuevas herramientas he encontrado algunas como módulos de GitHub que podías instalar para después convertir esos archivos IFC a archivos reconocidos por Unity, pero, aun así, no me ha funcionado ninguno.

Tras numerosos intentos, acabé encontrando otra herramienta, que vi más viable para poder trabajar con ella llamada IfcOpenShell, la cual tiene su propia web, repositorio con los archivos para descargar y una documentación de cómo funciona la aplicación una vez instalada. Pese a esto, al igual que con la primera herramienta, la he tenido que descartar sin poder obtener algún tipo de resultado referente a la carga de archivos, por problemas de instalación y módulos.

4. PiXYZ

Finalmente, tras numerosos intentos de búsqueda de alguna aplicación que permitiese el objetivo que estaba buscando, fue cuando contactamos con el otro profesor, que era el que había generado los archivos IFC. El motivo de contactar fue que sabíamos que el profesor había conseguido cargar los archivos en Unity, y ya había estado trabajando con ellos, por lo que fue ya tras numerosos intentos, cuando le preguntamos qué aplicación había usado él para poder visualizar estos archivos. La respuesta fue que la aplicación que había usado era PiXYZ, por lo que comencé a investigar sobre dicha aplicación.

En primera instancia, al igual que algunas aplicaciones comentadas anteriormente, esta aplicación, al ser desarrollada por una empresa, es de pago, pese a que ofrece una serie de días de prueba. En este caso, la aplicación ofrece tres productos diferentes, pero el que nos interesa a nosotros es “PiXYZ Plugin for Unity “. El principal problema que tiene la aplicación es que, al ser de pago, solo dispones de una prueba de 7 días para utilizarlo, a no ser que pagues el precio del plugin el cual asciende a 1000€ que cuesta tener la licencia durante un año. Como es evidente, en mi caso, me he registrado y he usado la aplicación durante ese periodo de prueba que ofrece, para ver qué resultados obtengo.

Como primer paso, me he descargado el plugin necesario para instalar en Unity, el cual, una vez te lo descargas y lo ejecutas, te pide que importes dicho paquete a Unity. Cargados ambos archivos en la escena, ya solo quedaría ubicarlos y empezar a trabajar con ellos, tal y como se aprecia en Figura 4 Resultados obtenidos mediante plugin de PiXYZ.

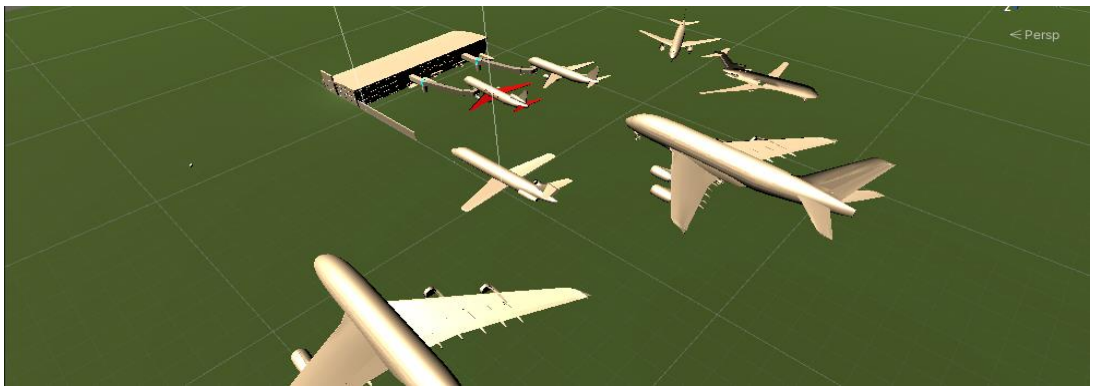


Figura 4 Resultados obtenidos mediante plugin de PiXYZ

4.2 Desarrollo de un escenario para la simulación

A modo de introducción para esta segunda tarea en el que se ha dividido el trabajo final, empezaré explicando y comentando la estructura que se ha utilizado para su documentación:

- Primero explicaré los diferentes escenarios que he creado para la simulación.
- En segundo lugar, explicaré en qué consiste cada zona que ha sido desarrollada y cuál es su objetivo dentro de la simulación.
- Finalmente explicaré el papel que lleva a cabo cada uno de los agentes dentro de esa zona.

Para la elaboración de los diferentes escenarios, me he guiado por un diagrama de flujo de las diferentes acciones que tiene que llevar a cabo un pasajero durante su recorrido por un aeropuerto. Hay que mencionar que la simulación se llevará a cabo solo durante el proceso de salida de un aeropuerto, pasando por las fases de Check-In, hasta el embarque en el avión.

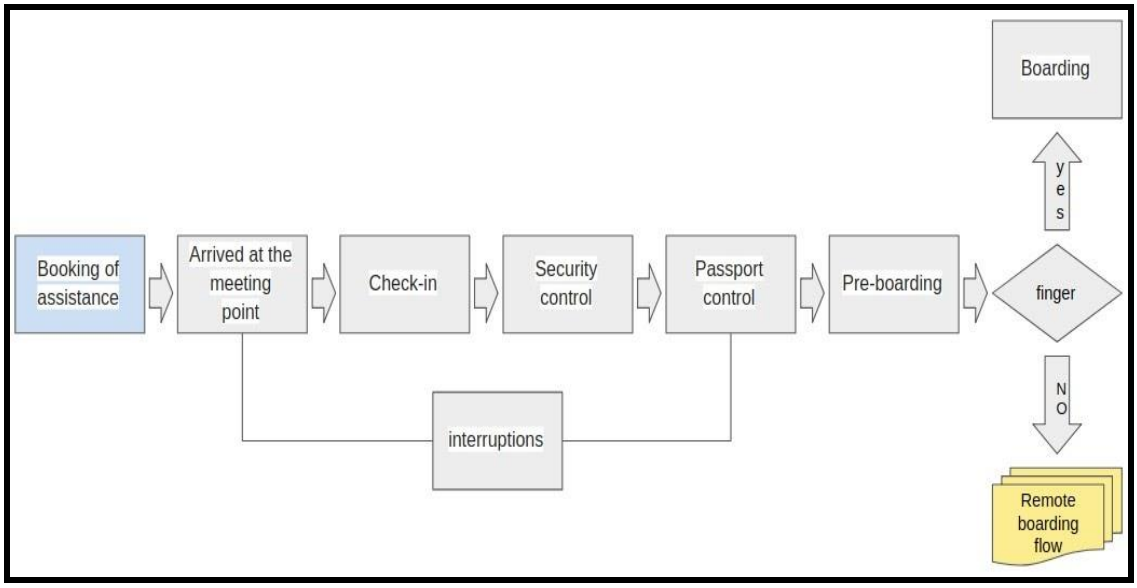


Figura 5 Flujo de acciones de la simulación.

(Extraída con permiso de: J.J. Herrera Martín e Iván Castilla Rodriguez. A conceptual model for assessing the impact of internet-of-things. Technologies for people with reduced mobility in airports)

Siguiendo el esquema de la Figura 5 Flujo de acciones de la simulación., el pasajero pasará por diferentes puntos del aeropuerto donde se llevarán a cabo diferentes acciones, las cuales procedo a describir detalladamente:

- **Booking of assistance (reserva del servicio de asistencia):** esta primera tarea consiste en la reserva del servicio de asistencia, la cual se llevará a cabo con días de antelación. En cuanto a la simulación, se simulará, valga la redundancia, que el pasajero ya se encuentra con este servicio reservado, por lo que no se tendrá en cuenta.

- **Arrived at the meeting point (llegada al punto de encuentro):** tras la llegada al aeropuerto, el pasajero acudirá al punto de encuentro con el asistente que le guiará por el resto de las tareas a desarrollar. Generalmente, en los aeropuertos reales, estas zonas se encuentran designadas como tótems de balizamiento.
- **Check-In (facturación):** esta es la primera tarea a la que tienen que acudir los pasajeros una vez llegados al aeropuerto, donde se registrará en el mostrador de la aerolínea, previa a la salida del vuelo. Junto al asistente, el PMR acudirá al mostrador indicado con el objetivo de registrarse y embarcar sus pertenencias en el avión.
- **Security control (control de seguridad):** tras pasar el Check-in, el pasajero estará listo para pasar el control de seguridad el cual le dará acceso a la zona interior del aeropuerto desde la cual se efectúan los embarques a los aviones.
- **Passport control (control de pasaporte):** en caso de que el pasajero haya realizado o provenga de un vuelo internacional, éste deberá dirigirse a la zona de control de pasaporte con el objetivo de comprobar que su pasaporte esté en orden.
- **Pre-boarding (pre-embarque):** tras pasar el control de seguridad o en su debido caso el control de pasaporte, el pasajero, tras acceder a la zona interior del aeropuerto, se dirigirá a su correspondiente puerta de embarque donde se comprobará su billete de avión posteriormente dándole acceso al avión.
- **Boarding (Embarque):** finalmente, tras completar todas las tareas anteriores, el pasajero ya se encuentra listo para realizar el embarque en el avión y dirigirse a su asiento asignado. En esta fase, los aeropuertos disponen de dos métodos de embarque diferentes:
    - **Finger (pasarela de acceso al avión):** en caso de ser un vuelo a una escala mayor a nivel nacional, el acceso al avión se realizará mediante una pasarela de acceso al avión, lo cual permitirá un fácil acceso a los PMR al embarque (Figura 6 Pasarela de acceso al avión.).

Figura 6 Pasarela de acceso al avión.

    - **Ambulifts (vehículo elevador de personas):** en caso de que el acceso al avión se realice mediante escaleras a nivel de pista, dado que los PMR no podrán acceder, el aeropuerto realizará el embarque de estos mediante unos dispositivos elevadores de personas. (Figura 7 Elevador de personas ambulifts)

Figura 7 Elevador de personas ambulifts

- **Interruptions (Interrupciones):** durante las primeras tareas del diagrama de flujo, se incluyen las denominadas interrupciones, que son aquellas acciones que el pasajero desea voluntaria o involuntariamente realizar, bien sea ir al baño, visitar alguna tienda, etc. En estas tareas deberá acompañarlo el asistente. A nivel operativo, estas tareas se encuentran separadas del flujo principal, ya que no son acciones estrictamente necesarias para la realización del vuelo.

Tras comentar el diagrama de flujo de acciones que se abordará, hay que comentar algunas consideraciones a la hora de llevar a cabo la simulación, ya que no todas las acciones comentadas anteriormente se llevarán a cabo:

- ❖ En primer lugar, se considerará que el pasajero ya acude al aeropuerto con el servicio de asistencia reservado.
- ❖ En segundo lugar, no se tendrá en cuenta la fase de control de pasaporte, por lo que, a nivel de simulación, solo se simulará el control de pasajeros.
- ❖ En cuanto a la fase de embarque, en mi simulación se llevará a cabo por pasarela de acceso al avión, en vez de ambulifts (posible implementación a futuro, 5.2 Líneas Futuras).
- ❖ En cuanto a las interrupciones, estas no serán consideradas en la simulación (posible implementación a futuro, 5.2 Líneas Futuras).

4.2.1 Desarrollo del escenario real

Esta primera parte de la tarea ha consistido en juntar los tres archivos cargados en la tarea anterior (4.1 Representación de archivos IFC en Unity 3D) para así obtener el escenario final donde se llevará a cabo la simulación.

- En primer lugar, se ha elaborado la terminal principal donde se desarrolla la simulación.
- Una vez elaborada lo que es la parte externa de la escena, se ha pasado a la parte interna de la terminal, la cual ha sido dividida a su vez en diferentes secciones:
    - **Mostrador**: en primer lugar, se ha elaborado la parte principal, considerada como "mostrador", que será la zona a la que acudirán ambos pasajeros tanto PMR como no PMR. El objetivo principal por el cual se ha elaborado esta primera zona es la de simular que, cuando ambos pasajeros llegan al aeropuerto, acuden al mostrador de información, en busca de ayuda y orientación sobre cómo poder llegar a su vuelo.

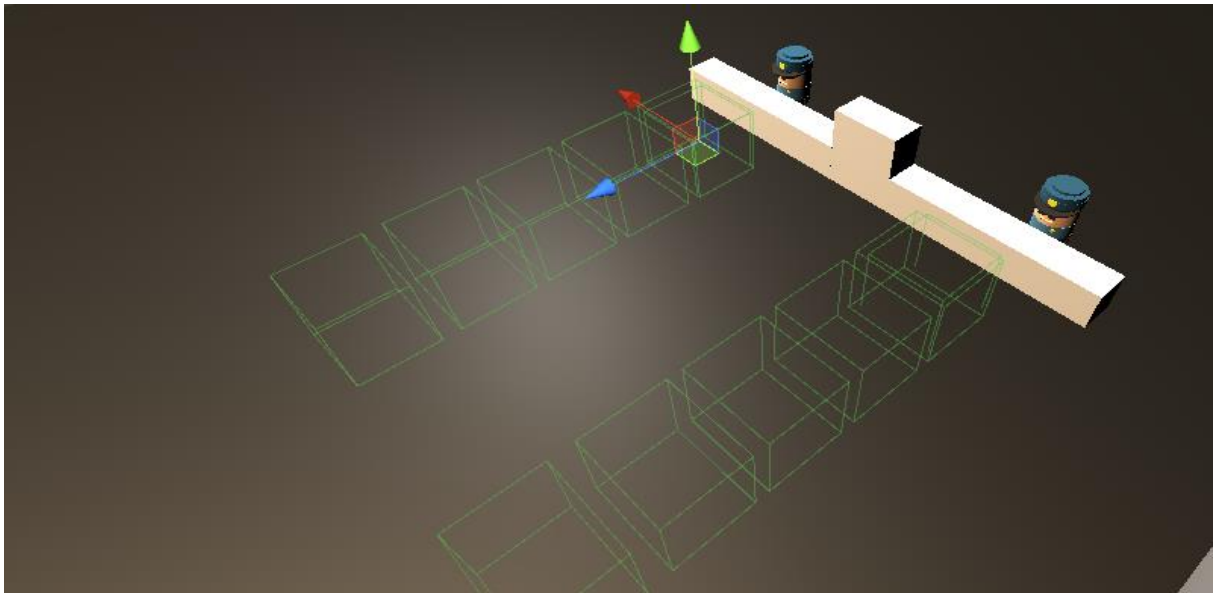


Figura 8 Zona de mostrador de información

En la Figura 8 Zona de mostrador de información, observamos la zona de Mostrador. Se han implementado dos mostradores para que la tarea de orientación para su vuelo sea algo más ágil que teniendo una sola cola. Dentro de cada una de las colas, podemos observar dos filas de cubos delimitados por bordes verdes, estos cubos son los diferentes puntos de la cola que van a ser ocupados por los pasajeros.

- **Zona de espera (Waiting Area):** es la zona a la que acudirán los pasajeros PMR, una vez han terminado en el mostrador de información. La idea de esta zona es que cada PMR que llega al aeropuerto, acuda a un punto de espera a que venga un asistente, quien le guiará y ayudará por las diferentes tareas que tiene que llevar a cabo por el aeropuerto. En los aeropuertos reales, esta zona que yo he desarrollado como una habitación, consiste en una especie de punto de encuentro a la que acudirá un asistente. Para los pasajeros que no sean PMR, esta zona no les afecta en nada, ya que nunca acudirán a ella.

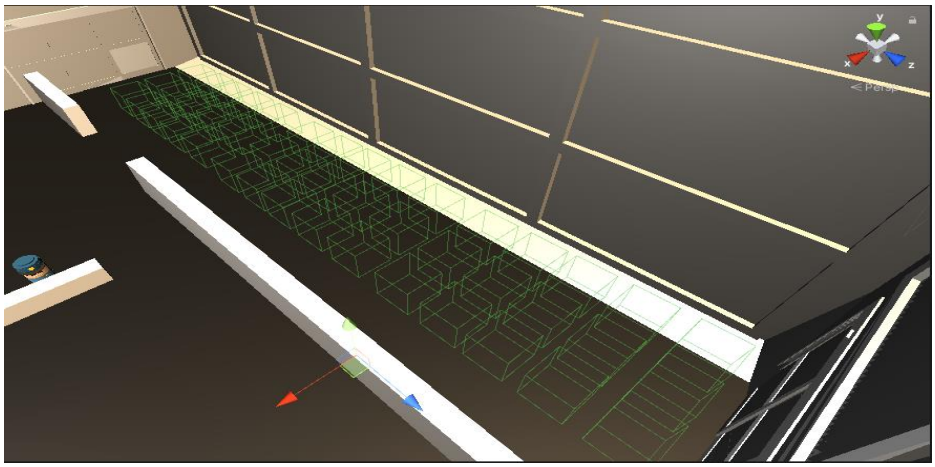


Figura 9 Zona de espera de pasajeros en silla de ruedas

En la Figura 9 Zona de espera de pasajeros en silla de ruedas, observamos la zona de espera de los pasajeros en silla de ruedas. A diferencia de la zona de mostradores, en este caso no es ninguna cola, sino que son diferentes puntos que van a ir ocupando los pasajeros a medida que estos van terminando en el mostrador para acudir a la espera de un asistente que los lleve al resto de tareas que tienen que desempeñar.

- **Facturación (Check In):** este tercer punto, es el mostrador al que acudirán los diferentes pasajeros que se encuentren en el aeropuerto para llevar a cabo la facturación de su equipaje. En el caso de los pasajeros no PMR, una vez terminan en el mostrador de información, acuden directamente a la cola del mostrador de Check In que le haya sido asignado para llevar a cabo la facturación de su equipaje. Para el caso de pasajeros PMR, estos solo acudirán, tras pasar por la zona de espera, y cuando ya tengan un asistente asignado, que será quien los lleve a dicho mostrador y le ayudará con el resto de las tareas. (Figura 10 Zona de mostradores de facturación)

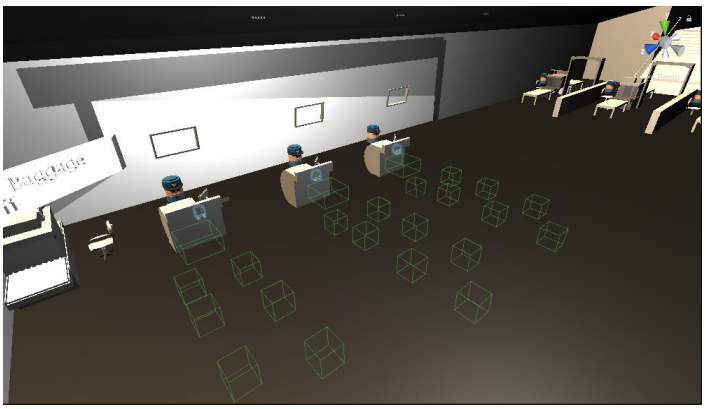


Figura 10 Zona de mostradores de facturación

En la Figura 10 Zona de mostradores de facturación, observamos la zona de mostradores de Check In. Esta zona está dividida en tres mostradores, ya que era con los que contaba los archivos IFC. Además de los tres mostradores, de la misma manera que el resto de las zonas, contamos con tres colas una para cada mostrador, ahora con más puestos dentro de cada una de las colas, en disposición de zigzag para que de esta manera haya sitio para que la cola sea más grande. Por último, cada una de las colas cuenta con un cubo más grande que es la primera posición de la cola, donde el primer pasajero que se encuentre en ella llevará a cabo la acción de simular que se encuentra haciendo la facturación de su equipaje.

- **Controles:** tal y como indica el propio nombre de la zona, es en este punto donde se lleva a cabo el control a cada uno de los pasajeros que pretenden acceder a la zona interna del aeropuerto, o más en concreto, a las diferentes puertas de embarque. De la misma manera que la zona anterior, ambos tipos de pasajeros, una vez terminan en la zona de Check In, acudirán directamente a la zona de control, simulando la misma tarea que en un aeropuerto real.

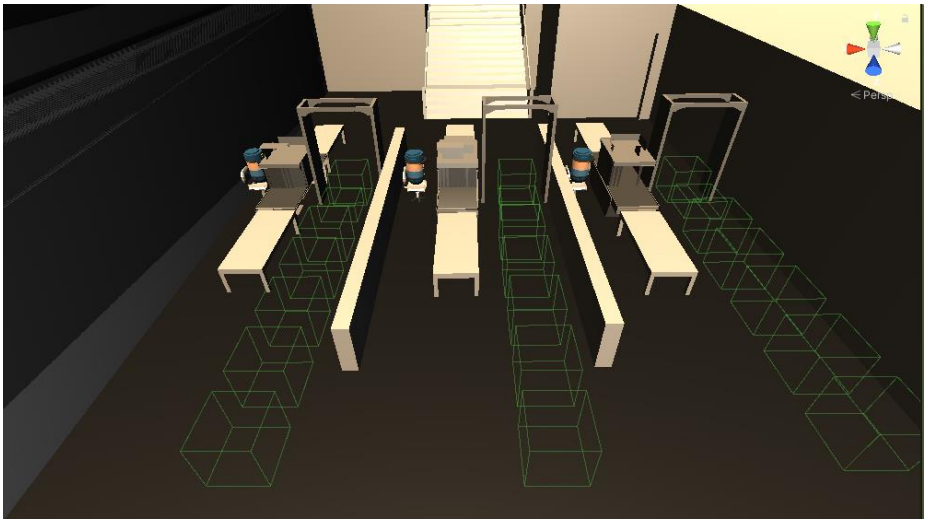


Figura 11 Zona de controles de seguridad

En la Figura 11 Zona de controles de seguridad, observamos la zona de controles. Esta zona está dividida en tres controles, ya que era con los que contaba el aeropuerto contenido en los archivos IFC. Además de los tres mostradores, de la misma manera que el resto de las zonas, contamos con tres colas una para cada control, ahora con más puestos dentro de cada una de las colas. Por último, cada una de las colas cuenta con un cubo más grande que es la primera posición de la cola, donde el primer pasajero que se encuentre en ella llevará a cabo la acción de simular que se encuentra haciendo el control para posteriormente acceder a la zona interna del aeropuerto desde donde tendrán acceso a las diferentes puertas de embarque.

- **Embarque (Boarding):** esta es la última zona con la que cuenta la simulación. De manera similar a un aeropuerto en un caso real, esta zona es el punto al que tendrán que acudir como objetivo final para tomar el vuelo a su destino. Igual que las dos zonas anteriores, esta zona es común a ambos tipos de pasajeros ya que indistintamente tienen que acceder a ella para entrar al avión (Figura 12 Zona de puertas de embarque).

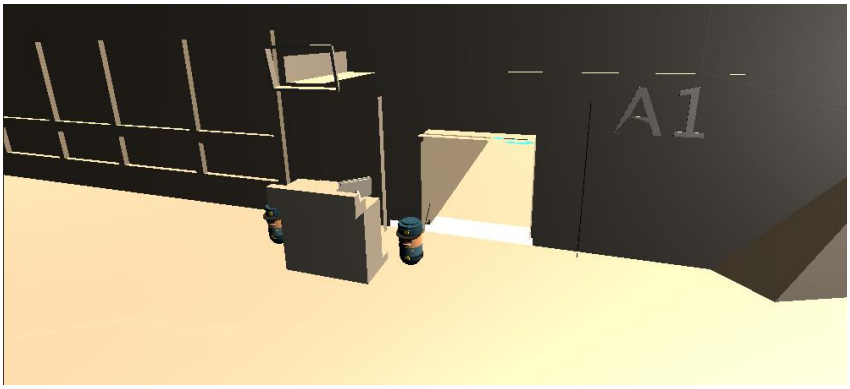


Figura 12 Zona de puertas de embarque

En la Figura 12 Zona de puertas de embarque, observamos la zona de embarque. Esta zona está dividida en dos puertas de embarque, ya que era con las que contaba el aeropuerto contenido en los archivos IFC. A diferencia de las zonas anteriores, esta es la zona más fácil, ya que no cuenta con ninguna implementación específica, simplemente consiste en un punto a donde acudirán los pasajeros, esperarán un tiempo determinado, simulando el tiempo de embarque que tienen que hacer fuera de las diferentes pasarelas de acceso al avión.

Finalizada la elaboración de las diferentes zonas por las que va a tener que pasar cada uno de los pasajeros, me encontré con un pequeño problema que fue que a la hora de llevar a cabo la simulación, mientras se ejecutaba, a la hora de desplazarse de un punto a otro, los pasajeros tomaban caminos que no estaban considerados como zonas por donde transitar o simplemente que no tenía sentido el camino por el que se estaban desplazando, bien sea porque atravesaban zonas destinadas a la recepción de pasajeros o bien porque a la hora de ir del mostrador de Check-in al punto de control, daban rodeos atravesando otras zonas por las que no debían estar. Por ello, tras finalizar las diferentes zonas de acción, la siguiente parte de esta tarea fue la elaboración de algunas paredes que se encuentran ocultas, de tal forma que los pasajeros puedan acceder a las zonas de manera más fácil, y que a la hora de ejecutar la simulación tenga un orden y sea lo más parecido a un aeropuerto en la vida real.

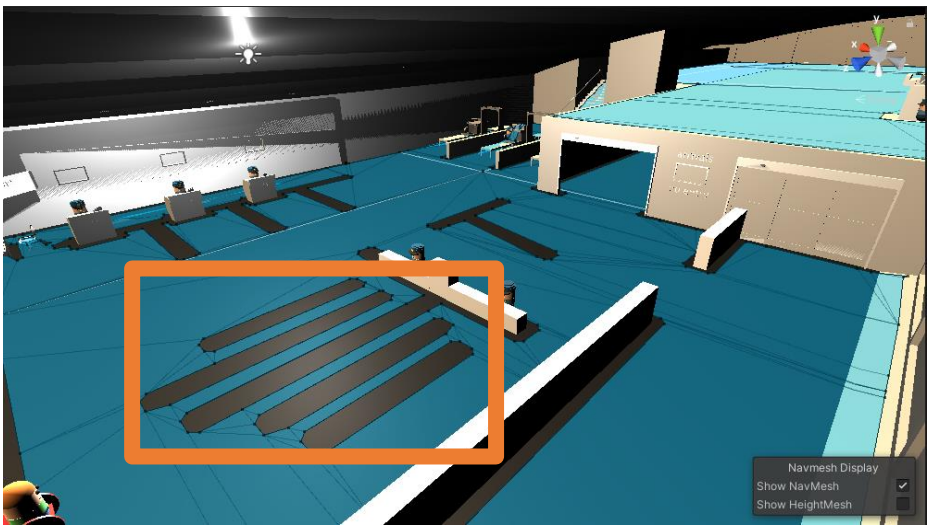


Figura 13 Paredes ocultas para la simulación

Tal y como se aprecia en Figura 13 , esas marcas en el suelo son las denominadas paredes ocultas, que facilitaran el tránsito de los pasajeros por el aeropuerto, permitiendo que mantengan un orden y una coherencia bien sea a la hora de formar las colas, o a la hora de transitar por el propio aeropuerto.

4.2.2 Desarrollo del escenario de pruebas

Para facilitar las pruebas, he diseñado un primer escenario más simple que me permitiera comprobar el funcionamiento del modelo de simulación sin preocuparme de las características concretas del escenario. Es en este escenario donde trabajo, desarrollo y modelo las diferentes zonas y puntos de la simulación, y una vez terminados y comprobado que funcionan perfectamente, es cuando he pasado todos esos puntos al escenario práctico (el aeropuerto oficial). La estructura en la que ha sido dividido este escenario ha sido exactamente la misma que el escenario real, es decir, cuenta con las mismas zonas, los mismos puntos. La diferencia principal radica en que, en vez de tener forma de aeropuerto, simplemente consiste en un par de habitaciones donde de manera más simplificada, se simula que el pasajero lleva a cabo la actividad en ese punto y pasa al siguiente.

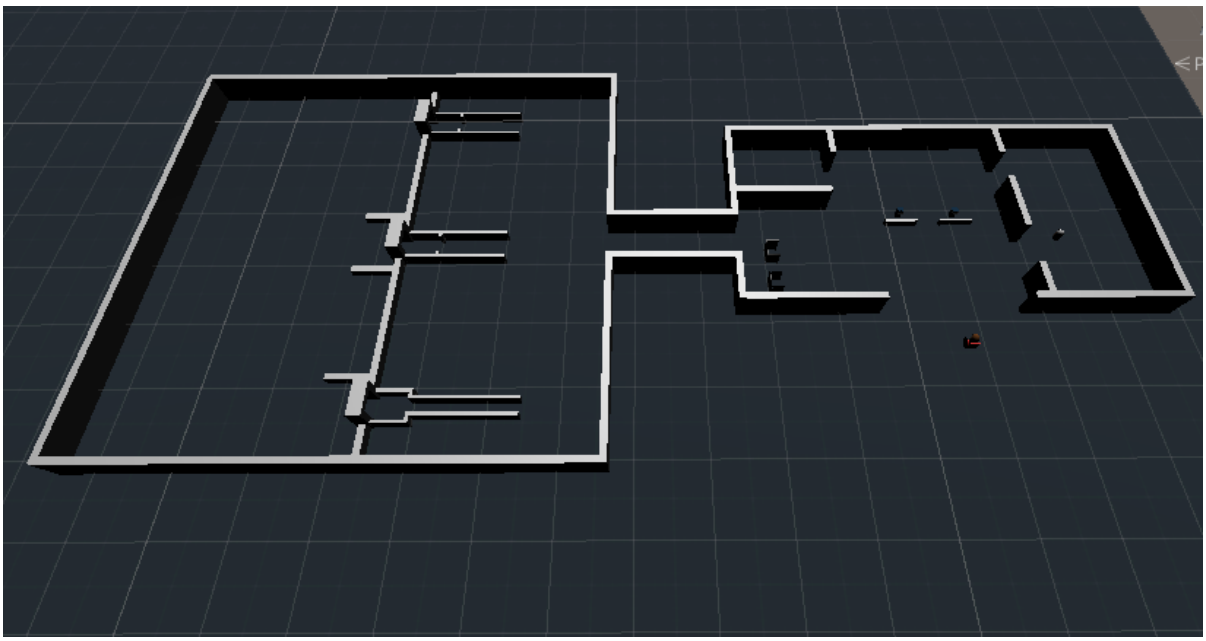


Figura 14 Escenario de pruebas

De manera similar al escenario real, el escenario de desarrollo cuenta con las mismas características y zonas, pero dispuestas en diferentes posiciones:

Figura 15 Zona de mostradores de información de escenario de pruebas

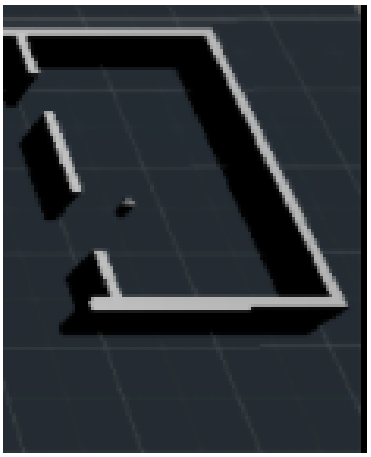


Figura 16 Zona de espera de pasajeros en silla de ruedas

Figura 17 Zona de mostradores de facturación

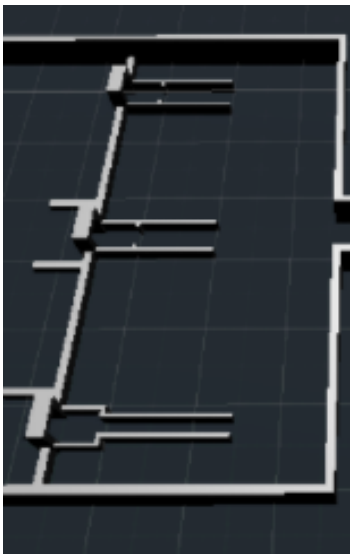


Figura 18 Zona de controles de seguridad

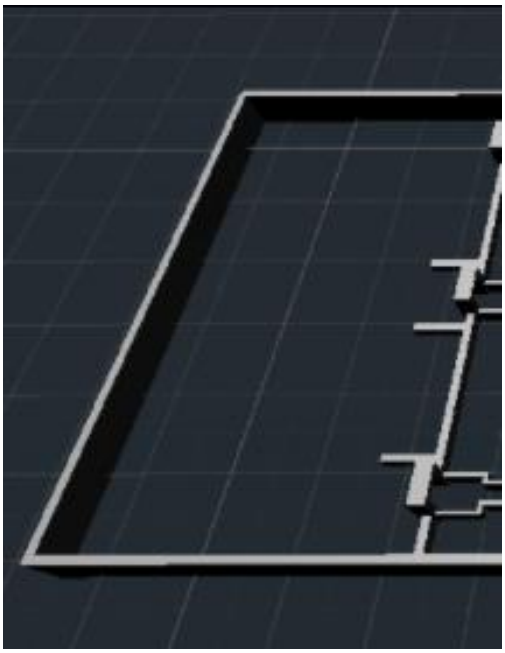


Figura 19 Zona de puertas de embarque

En cuanto a esta segunda tarea en la que ha sido dividido el trabajo, comentar, que en primer lugar se ha desarrollado el escenario de prueba, y una vez lo tenía totalmente operativo, ha sido cuando he pasado a desarrollar el escenario real.

4.3 Implementación de la simulación mediante GOAP

Terminadas las tareas anteriores, es en este punto cuando comienza el desarrollo de la simulación. Para que la comprensión de la implementación se lleve a cabo de la manera más fácil posible, lo que haré será, estructurar este punto en diferentes partes:

- En primer lugar, empezaré explicando en qué consiste un modelo GOAP
- A continuación, procederé a explicar cómo poder implementar este sistema en Unity.
- Finalmente explicaré los diferentes scripts con los que cuenta la simulación, necesarios para el desarrollo de las diferentes actividades del aeropuerto.

4.3.1 Modelo GOAP

GOAP (Goal-Oriented Action Planning) consiste en una técnica utilizada en el mundo de la programación, principalmente en el mundo de los videojuegos, la cual permite controlar el comportamiento de los Personajes No Jugables (abreviado NPC en inglés, Non Playable Characters).

Para entenderlo de manera sencilla, podríamos asimilar el comportamiento a una máquina de estado finito, la cual va dirigiendo al NPC por un camino de acciones hasta lograr su objetivo. El problema de este caso es que puedes tener un grafo considerablemente grande, o dividirlo en unidades más pequeñas, teniendo varios grafos con varias máquinas de estados finitos.

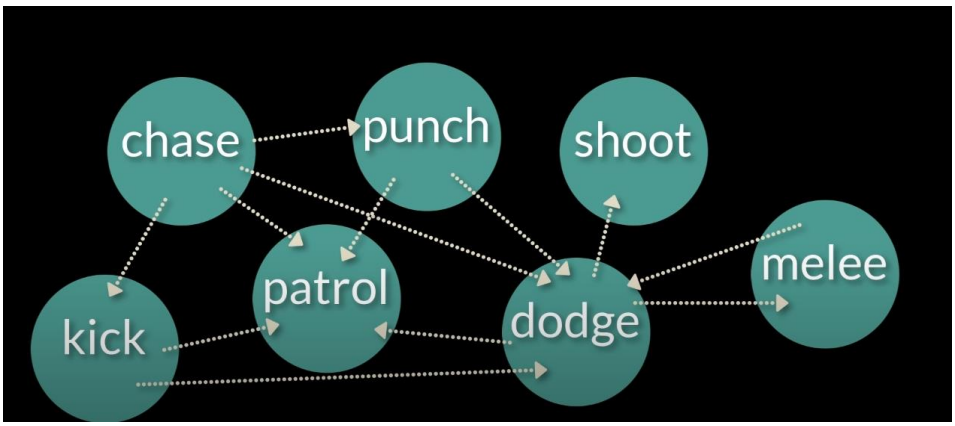


Figura 20 Acciones de una máquina de estado finito

En GOAP, la diferencia principal con una máquina de estado finito es que, desde un principio, el conjunto de acciones definidas para la simulación, no están enlazadas, es decir, en el grafo de la Figura 21 solo se tendrían los nodos, no las aristas. La forma de funcionar de GOAP consiste en ir comprobando el estado del agente respecto al mundo para así saber qué secuencia de acciones llevar a cabo.

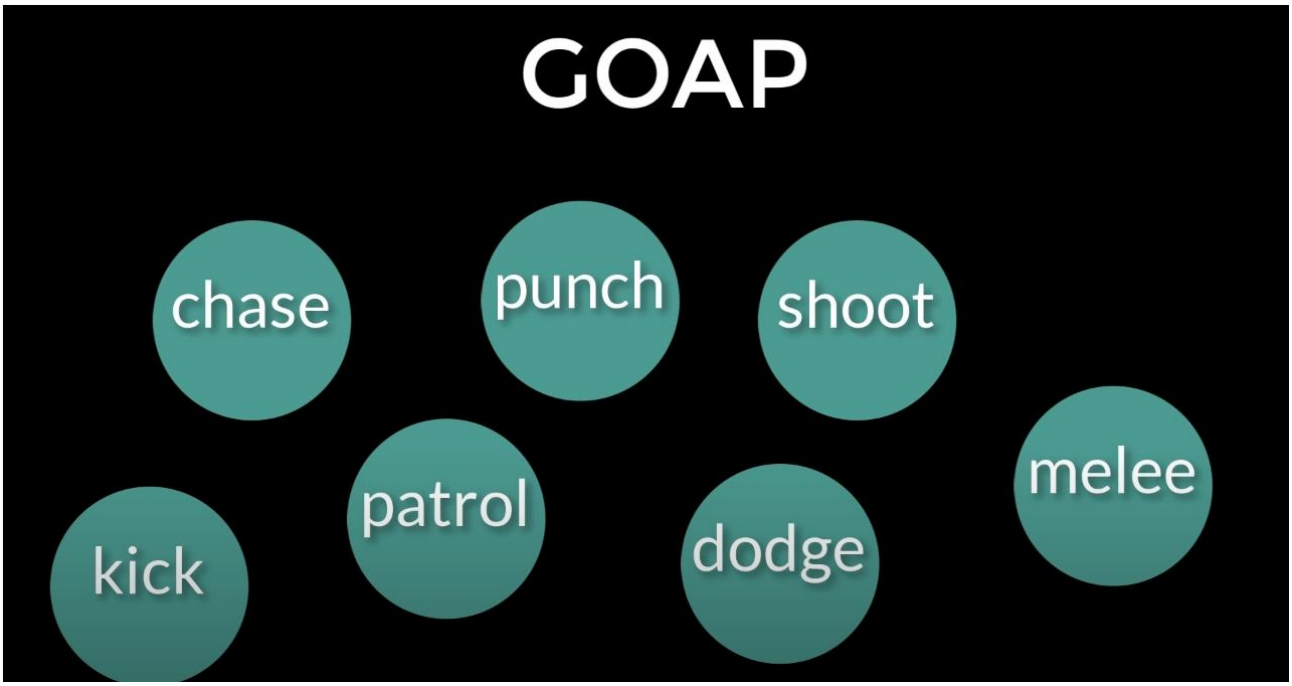


Figura 21 Acciones de un modelo GOAP

Por ejemplo, en la Figura 21 Acciones de un modelo GOAP, pongamos la situación de que un agente, recibe la orden de "matar a un enemigo". En el caso de una máquina de estado finito, lo que haría sería ir iterando por cada una de las acciones en la secuencia en la que las tiene programadas para llegar a la acción objetivo. GOAP, por el contrario, lo que hace es que, partiendo del estado actual del agente y un objetivo (por ejemplo, eliminar un enemigo), analiza las diferentes opciones que tiene en cada momento. Por ejemplo, una opción sería perseguir (Chase), esquivar (Dodge) y luego disparar (Shoot) o si por ejemplo en ese momento, el agente no tiene un arma, pues podría atacar cuerpo a cuerpo, sustituyendo la acción "Shoot" por "Melee".

En cuanto a la estructura de un modelo GOAP, este se basa en que un personaje, tiene una serie de acciones que completar de manera ordenada, donde cada acción viene dada por una precondición y un efecto que, a su vez, causará una serie de efectos dentro de la simulación.

Toda acción en un modelo GOAP viene dividida en tres partes:

- **Precondición**: se trata de una condición que tiene que cumplirse para que se pueda llevar a cabo dicha acción.
- **Acción**: consiste en la acción que se lleva a cabo, en caso de cumplirse la precondición.
- **Efecto**: es el estado en el que se queda el agente o el mundo de la simulación una vez se termina la acción.

Dentro de un modelo GOAP, cada acción lleva consigo un coste. Este coste es el valor que supone llevar a cabo dicha acción, es por eso por lo que, en el caso de que el planificador de acciones dentro del modelo tenga que elegir entre varias secuencias de acciones, este selecciona la secuencia de acciones de menor coste.

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

public class GoToHospital : GAction
{
    public override bool PrePerform()
    {
        if (Random.Range(1, 3) == 1)
        {
            this.GetComponent<GAgent>().mostradorAsignado = "MostradorRight";
        }
        else
        {
            this.GetComponent<GAgent>().mostradorAsignado = "MostradorLeft";
        }

        targetTag = this.GetComponent<GAgent>().mostradorAsignado + "QueueStart";

        return true;
    }

    public override bool PostPerform()
    {
        beliefs.ModifyState("pmrNoInAirport", -1);

        GameObject queue = GameObject.FindWithTag(this.GetComponent<GAgent>().mostradorAsignado + "Queue");
        queue.GetComponent<QueueManager>().EnqueueAgent(this.gameObject);

        return true;
    }
}
```

Figura 22 Estructura de una acción en GOAP programada en Unity

En la Figura 22 se describe el código que he implementado para definir una acción, la cual asignará un mostrador de manera aleatoria a cada uno de los pasajeros que van llegando al aeropuerto. La estructura para el resto de las acciones es exactamente la misma, lo único que cambia es el código que se realiza dentro del PrePerform y del PostPerform.

De forma gráfica, un modelo GOAP, sería algo como la **¡Error! No se encuentra el origen de la referencia.**. Dado un conjunto de acciones, donde cada acción tiene una precondición y un efecto, se establecerá una cadena de acciones en función del estado en el que se vaya encontrando el agente a la hora de realizar la acción, de tal forma que los puntos que aparecen en ambos lados de las acciones se vayan uniendo entre sí, formando una cadena de acciones que permitan al agente cumplir el objetivo asignado.

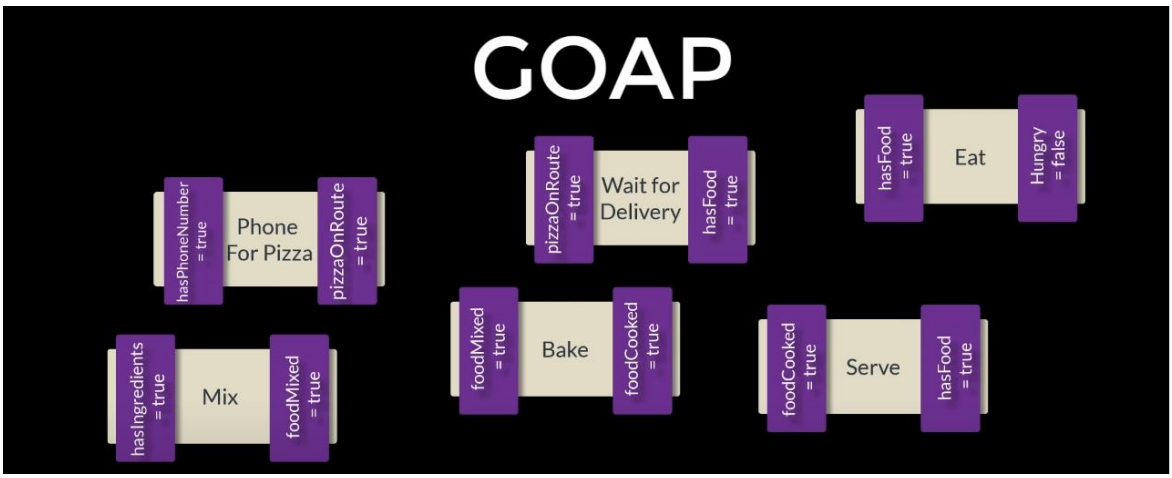


Figura 23 Caso real de Modelo GOAP

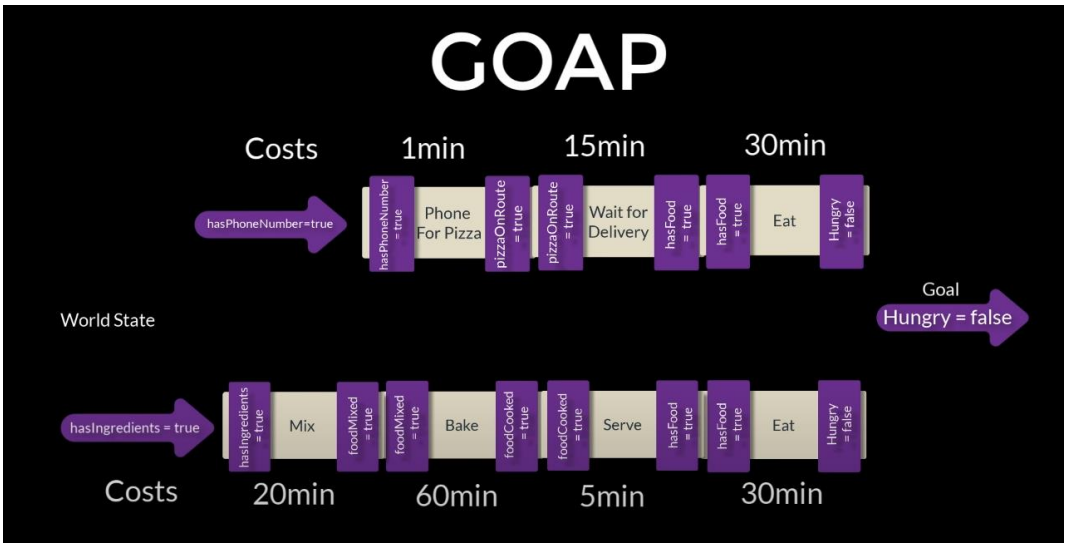


Figura 23 Caso real de Modelo GOAP 3

(Figura 23-24 extraída de "An introduction to Goal Oriented Action Planning. Holistic3. Autora: Penny de Byl")

Para finalizar con la parte referente al primer componente con el que se ha trabajado, solo quedaría explicar cómo funciona GOAP, ya que hasta este punto he comentado en qué consiste, y la estructura principal. Para ello, me basaré en la siguiente imagen:

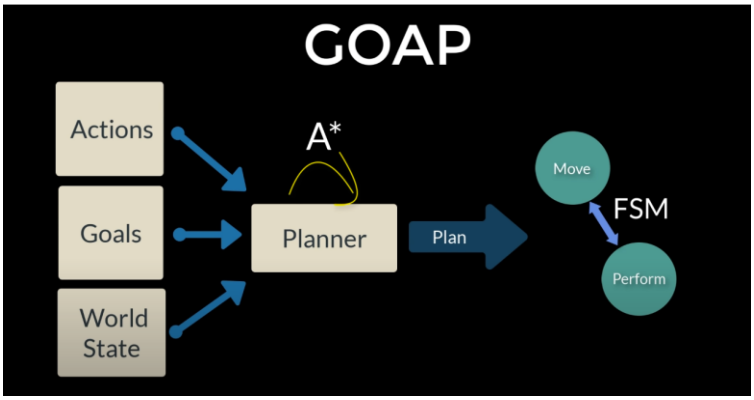


Figura 24 Funcionamiento de modelo GOAP

(Figura 25 extraída de "An introduction to Goal Oriented Action Planning. Holistic3. Autora: Penny de Byl")

Tal y como se parecía en la Figura 24 Funcionamiento de modelo GOAP, un modelo GOAP está dividido en diferentes secciones las cuales trabajaran en común para llevar a cabo la simulación:

- **Agentes:** representan los principales personajes dentro de la simulación, los cuales tienen una serie de variables asignadas, tales como el conjunto de acciones que tiene que llevar a cabo. Para mi trabajo, yo le he asignado a cada agente, una serie de variables específicas tales como el mostrador de Check In asignado, el mostrador de embarque asignado, ya que me facilitaba la simulación a la hora de programar.

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

public class SubGoal
{
    public Dictionary<string, int> sgoals;
    public bool remove;

    public SubGoal(string s, int i, bool r)
    {
        sgoals = new Dictionary<string, int>();
        sgoals.Add(s, i);
        remove = r;
    }
}

public class GAgent : MonoBehaviour
{
    public List<GAction> actions = new List<GAction>();
    public Dictionary<SubGoal, int> goals = new Dictionary<SubGoal, int>();
    public GInventory inventory = new GInventory();
    public WorldStates beliefs = new WorldStates();

    public GameObject pmrAsignado;
    public GameObject asistenteAsignado;
    public GameObject checkInAsignado;
    public GameObject controlAsignado;
    public GameObject embarqueAsignado;
    public GameObject pasajero;

    public string mostradorAsignado;
    public string mostradorCheckInAsignado;
```

Figura 25 Definición de agente en GOAP

Los agentes con los que cuenta la simulación son tres:

- **Pasajeros:** agente que llega al aeropuerto con el objetivo de embarcar en el avión correspondiente para realizar el vuelo planificado. Dentro de los pasajeros, a su vez encontramos dos tipos de pasajeros:
  - **PMR:** pasajeros de movilidad reducida que van en silla de ruedas.
  - **No PMR:** pasajeros que presentan una movilidad normal.
- **Asistentes:** agente encargado de guiar a los PMR por el aeropuerto.

- **Acciones:** es el conjunto de tareas a llevar a cabo por los agentes que se encuentran dentro de la simulación, tal y como se aprecia en la Figura 26 Ejemplo de acciones en GOAP donde se refleja el panel de configuración de una acción en Unity.

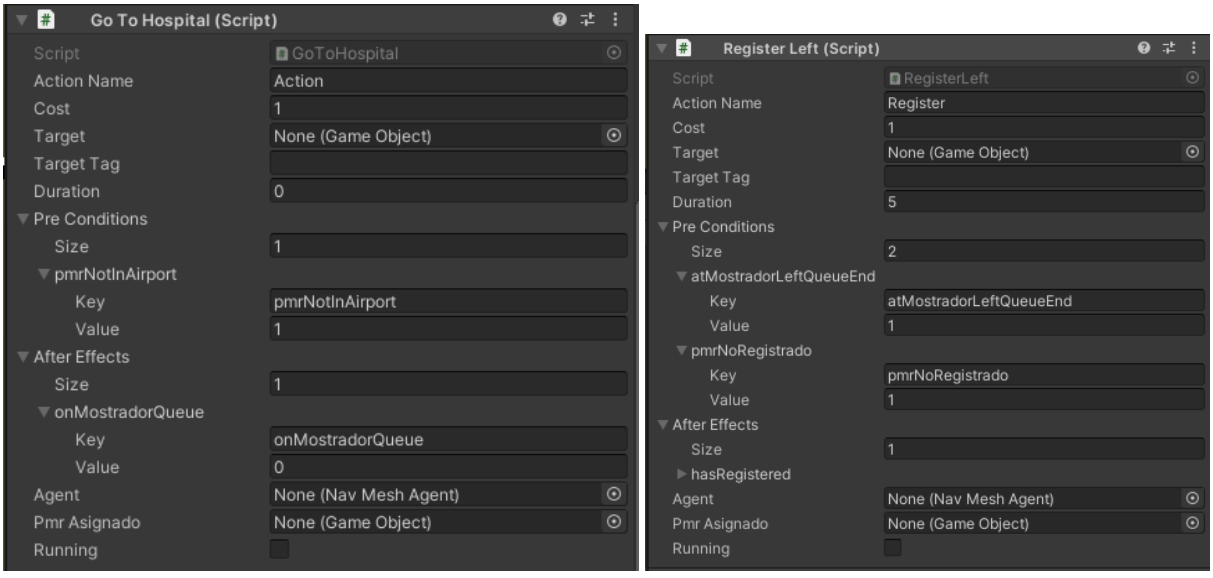


Figura 26 Ejemplo de acciones en GOAP

Dentro de este conjunto, podemos encontrar diferentes acciones que pueden llevar a cabo tanto pasajeros como asistentes.

En cuanto a las acciones que pueden realizar los **pasajeros**, he de mencionar que las acciones son las mismas para ambos tipos, exceptuando los PMR, que tienen una acción adicional que consiste en acudir al punto de espera por un asistente. Las acciones generales son:

- **Go to Airport:** esta es la primera acción de todo pasajero, la cual consiste simplemente en acudir al punto de información del aeropuerto, donde recibirán la información de su vuelo.

  Como toda acción de los agentes, estas tienen una precondición, que se debe cumplir previamente y un efecto, que modificará el mundo:

  - **Precondición:**
    - **pmrNotInAirport**: para que se pueda dar la acción, tiene que suceder que el pasajero no se encuentre en el aeropuerto. Inicialmente, no había ninguna precondición, por lo que una vez el PMR llegaba al embarque, este se dirigía al principio debido a que, al no haber precondición, se ejecutaba siempre, es por eso, por lo que, al poner la precondición, una vez terminado el ciclo, ya no se repetía más.
  - **Efecto:**
    - **onMostradorQueue:** este efecto, tiene doble función, ya que en primer lugar sirve para saber que se ha llegado al aeropuerto, pero además se utiliza para introducir al pasajero en la cola del mostrador.

- **Register Left/Right:** son dos acciones diferentes, ya que se le deben indicar ambas al agente, pero a nivel operativo hacen lo mismo, registran al pasajero en el mostrador de información, salvo que una lo hace en el mostrador izquierdo (Left) y otra en el derecho (Right).

  Como toda acción de los agentes, estas tienen una precondición, que se debe cumplir previamente y un efecto, que modificará el mundo:

  - **Precondición:**
    - **atMostradorLeftQueueEnd / atMostradorRightQueueEnd:** para que se dé la acción de registrar el pasajero en el mostrador, el pasajero debe encontrarse en el principio de la cola, es decir, debe ser el primero de esta, de tal forma que la siguiente posición que ocupe, sea la del mostrador. Se tiene una precondición para cada acción de registrarse, izquierda y derecha.
    - **pmrNoRegistrado:** de la misma forma que en la primera acción, para que no se produzcan ciclos, se establece la precondición de que para que se pueda registrar, este no tiene que haberse registrado anteriormente, de esta forma, solo accederán al mostrador pasajeros que no se hayan registrado.
  - **Efecto:**
    - **hasRegistered:** una vez registrado el pasajero, se establece este efecto, el cual a su vez servirá como precondición para la siguiente acción, de tal forma que, si no se ha registrado, no podrá pasar a la siguiente acción.

- **Go to Waiting Room:** solo los pasajeros PMR, tendrán asignada esta acción, la cual los llevará a la sala de espera, donde esperarán a que un asistente les sea asignado y vaya a recogerlos.

  Como toda acción de los agentes, estas tienen una precondición, que se debe cumplir previamente y un efecto, que modificará el mundo:

  - **Precondición:**
    - **hasRegistered:** el pasajero se dirigirá a la sala de espera, solo cuando se haya cumplido la acción de registrarse en el mostrador, independientemente del mostrador izquierdo o derecho.
  - **Efecto:**
    - **isWaiting:** como efecto de haber llegado a la sala de espera, la acción provocará en el pasajero un estado de espera por un asistente, el cual terminará en el momento de asignarle un asistente.

- **Get Check In:** una vez el pasajero haya sido recogido por un asistente (en el caso de un PMR) o se haya registrado en el mostrador (en el caso de un pasajero no PMR), estos se dirigirán al mostrador de Check-in, donde harán la facturación de su equipaje y se les entregará el pasaje para poder acceder al avión.

  Como toda acción de los agentes, estas tienen una precondición, que se debe cumplir previamente y un efecto, que modificará el mundo:

  - **Precondición:**
    - **Pasajeros PMR:**
      - **atWaitingRoom:** en el caso de un pasajero PMR, se establecerá esta precondición la cual indica que el pasajero se encuentra en la sala de espera de asistentes y que está listo para realizar el Check-in.
      - **pickUpPMR:** no solo basta con la otra precondición, ya que, si no estuviese esta, el PMR, una vez llega al punto de espera, se dirigiría al Check-in, ya que no se ha indicado que tiene que esperar por un asistente. Por ese motivo, se establece esta condición, la cual indica que el PMR, ya ha sido recogido por un asistente.
    - **Pasajeros NO PMR:**
      - **hasRegistered:** dado que el pasajero no PMR no tiene que acudir a la sala de espera, en este caso, el pasajero acudirá al mostrador de Check-in, una vez haya pasado por el mostrador de información.
  - **Efecto:**
    - **onCheckInQueue:** una vez llegado al punto del mostrador, previo al Check-in, se debe realizar la cola de espera, en función del turno de llegada. Para indicar que se ha llegado a la cola, se establece este efecto, el cual permitirá encolar al pasajero, en una de las colas asignadas para el Check-in.

- **Go Check In PMR 1-2-3:** una vez se ha terminado la cola, le corresponde al pasajero realizar la facturación, por lo que es en esta acción, donde el pasajero ha terminado la cola y se dirige al mostrador. Como se aprecia en el nombre de la acción, se han establecido varios mostradores, diferenciándolos con el indicativo del número.

Como toda acción de los agentes, estas tienen una precondición, que se debe cumplir previamente y un efecto, que modificará el mundo:

- **Precondición:**
  - **atCheckInQueue1End/atCheckInQueue2End/atCheckInQueue3End:** como ya se ha indicado, solo se accederá al mostrador una vez se haya terminado la espera en la cola del mostrador asignado, es el manejador de colas el que establecerá este estado interno en el pasajero, una vez se encuentre en la última posición de la cola.
- **Efecto:**
  - **isCheckedIn:** una vez se haya registrado el pasajero en el mostrador, se establecerá este efecto, indicando que el objetivo de registrarse ha sido cumplido.

❖ **Get Control PMR:** una vez se haya realizado la facturación, el pasajero estará listo para pasar el control de seguridad que le llevará a la zona interna del aeropuerto. En esta acción, al igual que en el resto de las acciones anteriores que comienzan con el indicativo "**Get**" se encolará al pasajero en la cola correspondiente del control asignado.

Como toda acción de los agentes, estas tienen una precondición, que se debe cumplir previamente y un efecto, que modificará el mundo:

- **Precondición:**
  - **checkedIn:** con esta precondición, se comprueba que el pasajero haya pasado la facturación, de tal forma que solo los pasajeros registrados y con billete, podrán acceder al control.
  - **pmrNoControlPasado:** como segundo nivel de seguridad y para evitar ciclos una vez se termine la acción, se establece como segunda precondición, que solo los pasajeros que no hayan pasado el control puedan acceder a él.
- **Efecto:**
  - **onControlQueue:** como el resto de las acciones anteriores encargadas de encolar agentes, con este efecto, se indica que se ha introducido un pasajero en la cola.

❖ **Go Control PMR 1-2-3:** una vez el pasajero se encuentra en la primera posición de la cola, le corresponde el turno de acceder al control para acceder a la zona interna del aeropuerto. Como se aprecia en el nombre de la acción, se han establecido varios controles, diferenciándolos con el indicativo del número.

Como toda acción de los agentes, estas tienen una precondición, que se debe cumplir previamente y un efecto, que modificará el mundo:

- **Precondición:**
  - **atControlQueue1End/atControlQueue2End/atControlQueue3End:** con esta precondición, se comprueba que el pasajero está en la primera posición de la cola, de tal manera que la siguiente posición le corresponde el turno en el control.

- **Efecto:**
  - **isControlPassed:** se indica con este efecto, que el pasajero ha completado el objetivo de pasar el control de seguridad.

❖ **Go Boarding PMR:** finalmente, una vez el pasajero ha pasado el control de seguridad, para terminar el ciclo de acciones, solo le quedaría realizar el embarque en el avión.

Como toda acción de los agentes, estas tienen una precondición, que se debe cumplir previamente y un efecto, que modificará el mundo:

- **Precondición:**
  - **controlPassed:** solo podrán acceder al embarque en el avión, aquellos pasajeros que hayan superado el control de seguridad. Para ello, una vez terminas la acción de control de seguridad, se establece un estado interno al pasajero de que se ha completado el control, para poder acceder al embarque.
- **Efecto:**
  - **isBoarding:** se indica con este efecto, que el pasajero ha completado el objetivo de acceder al embarque del avión.

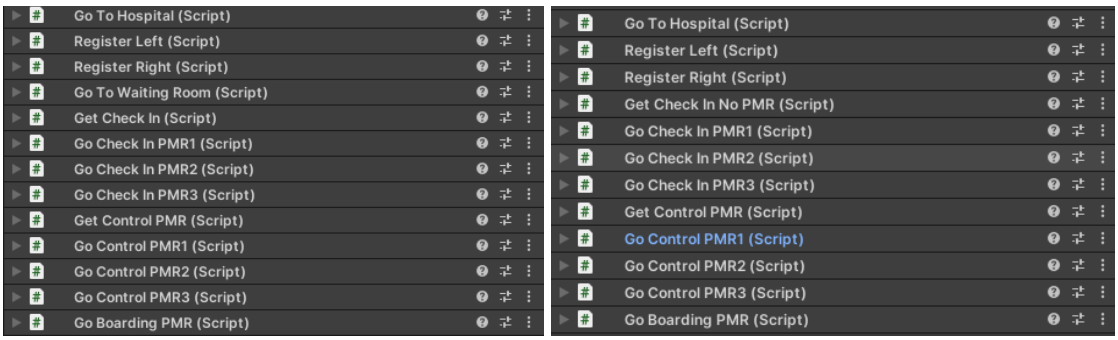


Figura 27 Ejemplo de acciones en GOAP

En cuanto a las acciones de los **asistentes**, ambos agentes (pasajeros y asistentes) comparten las mismas acciones, exceptuando una acción específica para los asistentes, la cual es la encargada de acudir al punto de espera a recoger a un PMR.

- **Objetivos:** es el conjunto de objetivos a lograr por los agentes que se encuentran dentro de la simulación. (Figura 28 Ejemplo de acciones en GOAP)

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

public class Patient : GAgent
{
    new void Start()
    {
        beliefs.ModifyState("pmrNotInAirport", 1);
        beliefs.ModifyState("pmrNoRegistrado", 1);
        beliefs.ModifyState("pmrNoControlPasado", 1);

        GameObject manager = GameObject.FindWithTag("AnalisisManager");
        manager.GetComponent<AnalisisManager>().AñadirPasajero(this.gameObject);
        manager.GetComponent<AnalisisManager>().IniciarCronometro(this.gameObject);

        base.Start();
        SubGoal s1 = new SubGoal("onMostradorQueue", 1, true);
        goals.Add(s1, 1);

        SubGoal s3 = new SubGoal("isWaiting", 1, true);
        goals.Add(s3, 3);

        SubGoal s4 = new SubGoal("isCheckedIn", 1, true);
        goals.Add(s4, 4);

        SubGoal s5 = new SubGoal("onCheckInQueue", 1, true);
        goals.Add(s5, 5);

        SubGoal s6 = new SubGoal("onControlQueue", 1, true);
        goals.Add(s6, 6);

        SubGoal s7 = new SubGoal("isControlPassed", 1, true);
        goals.Add(s7, 7);

        SubGoal s8 = new SubGoal("isBoarding", 1, true);
        goals.Add(s8, 8);
    }
}
```

Figura 28 Ejemplo de acciones en GOAP

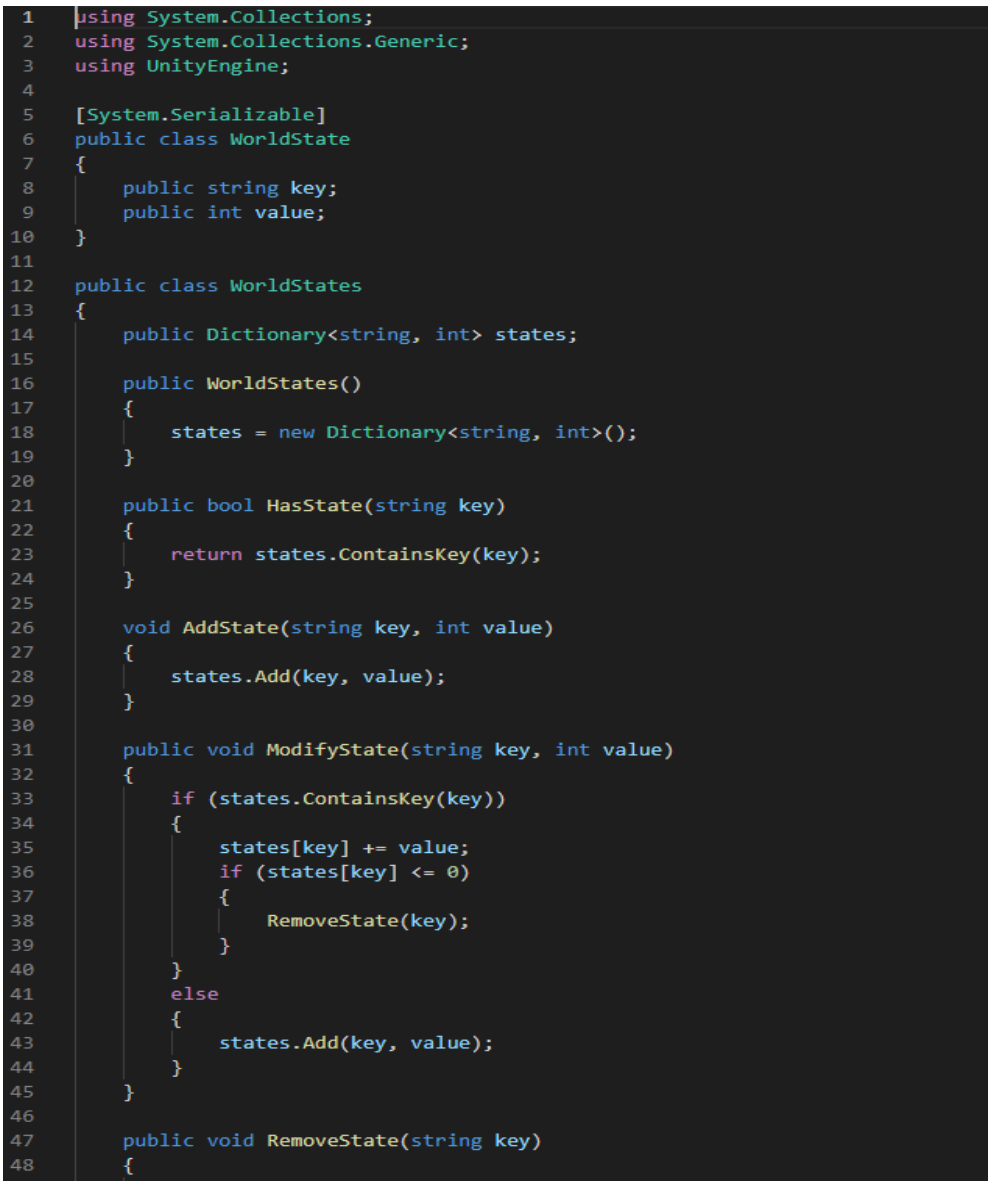
En la Figura 28 Ejemplo de acciones en GOAP, se contempla el código que he implementado para definir un agente de la simulación, en este caso el pasajero de movilidad reducida. Como se puede observar, se encuentran definidos varios objetivos, ya que a la hora de llevar a cabo la simulación programáticamente, es más sencillo el dividir el objetivo en varios subobjetivos, ya que además permite intercalar posibles interrupciones que quiera realizar el pasajero.

En cuanto al objetivo general de todo pasajero, se establece el objetivo general de llegar al aeropuerto con destino a embarcar en el avión. Pero para poder lograrlo, se establecen una serie de subobjetivos, los cuales harán de esta tarea, una tarea más sencilla de lograr:

- **onMostradorQueue:** objetivo el cual se completa cuando el pasajero llega a la cola del mostrador de información.
- **isWaiting:** el pasajero ha llegado al punto de espera, y se encuentra a la espera de un asistente.
- **isCheckedIn:** se cumple cuando el pasajero termina de facturar y de registrarse en el mostrador de Check-in.
- **onControlQueue:** el pasajero se encuentra en la cola de acceso al control de seguridad.
- **isControlPassed:** el pasajero termina la cola, y ha pasado el control de seguridad.
- **isBoarding:** el pasajero ha llegado a la puerta de embarque, y ha accedido al avión.

Una ventaja de dividir el objetivo general del pasajero en varios subobjetivos es que, en caso de implementar más acciones o interrupciones, la tarea se facilita, ya que, durante esas fases intermedias, el pasajero puede satisfacer las necesidades de ese momento.

- **Estado del mundo:** este es el estado que va a ir marcando la realización de las acciones, ya que, en función del estado actual, pues se llevará a cabo una acción u otra. (**¡Error! No se encuentra el origen de la referencia.**)



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

[System.Serializable]
public class WorldState
{
    public string key;
    public int value;
}

public class WorldStates
{
    public Dictionary<string, int> states;

    public WorldStates()
    {
        states = new Dictionary<string, int>();
    }

    public bool HasState(string key)
    {
        return states.ContainsKey(key);
    }

    void AddState(string key, int value)
    {
        states.Add(key, value);
    }

    public void ModifyState(string key, int value)
    {
        if (states.ContainsKey(key))
        {
            states[key] += value;
            if (states[key] <= 0)
            {
                RemoveState(key);
            }
        }
        else
        {
            states.Add(key, value);
        }
    }

    public void RemoveState(string key)
    {
```

Figura 29 Ejemplo de estados del mundo en GOAP

En la Figura 29 Ejemplo de estados del mundo en GOAP, es el código que he implementado para definir los estados del mundo que se van a poder dar en la simulación. Un estado estará definido por un par clave valor, donde la clave será el nombre que se le asigna al estado, y el valor, asociado a dicho estado.

Los estados a nivel del mundo con los que se trabajan en la simulación son:

- **FreeWaitingRoom:** este estado se utiliza para indicar la cantidad de espacios libres que hay disponibles en la zona de espera de un asistente.
- **Waiting:** este estado se utiliza para indicar que un pasajero PMR, se encuentra esperando por un asistente.

Adicionalmente a estos estados, también existen otra serie de variables similares, denominadas **"beliefs"**, que modifican el estado del agente al que se asigna, sin modificar el estado del mundo. De esta manera, se puede conseguir que, en determinados momentos, a la hora de elegir qué acción realizar, podemos descartar una u otra en función de los estados internos del agente. El conjunto de variables del agente es:

- **atAirport (pasajero):** se utiliza para indicar que el pasajero se encuentra en el aeropuerto.
- **atWaitingRoom (pasajero):** se utiliza para indicar que el pasajero PMR, se encuentra en la sala de espera por un asistente, una vez es recogido por un asistente, este estado es eliminado de su lista de estados internos.
- **FreeAsistente (asistente):** se utiliza para registrar la cantidad de asistentes disponibles, de tal manera, que cuando un asistente es asignado a un pasajero, se resta una unidad a la cantidad de asistentes libres, de tal manera que el resto de los pasajeros se encuentran a la espera.
- **pmrNoRegistrado (pasajero):** se utiliza como mecanismo de seguridad para que solo aquellos pasajeros que todavía no se han registrado se registren, evitando así bucles indebidos.
- **pmrPickUp (pasajero):** es exactamente el mismo estado que el anterior, con la diferencia que este se utiliza en los pasajeros, mientras que el otro en los asistentes.
- **controlPassed (pasajero):** se utiliza para indicar que el pasajero ha superado el control de seguridad, por lo que puede continuar con la siguiente acción.
- **pmrNotInAirport(pasajero):** se utiliza para indicar que el pasajero no se encuentra en el aeropuerto, principalmente se utiliza para evitar bucles una vez termina el ciclo de acciones.
- **checkedIn:** registra que el pasajero ha completado la acción de realizar la facturación.
- **atMostrador1/2/3QueueStart (asistente y pasajero):** indica que el pasajero se encuentra en el principio de la cola de registro en el mostrador, por lo que procede a encolarse al agente.
- **atMostrador1/2/3QueueEnd (asistente y pasajero):** indica que el pasajero se encuentra en el final de la cola de registro en el mostrador, por lo que procede a realizar la acción indicada, en este caso, procedería a realizar el registro en el mostrador de información.
- **atCheckInQueue1/2/3Start (asistente y pasajero):** indica que el pasajero se encuentra en el principio de la cola de Check-in, por lo que procede a encolarse al agente.
- **atCheckInQueue1/2/3End (asistente y pasajero):** indica que el pasajero se encuentra en el final de la cola de Check-in, por lo que procede a realizar la acción indicada, en este caso, procedería a realizar la facturación en el mostrador de Check-in.
- **atControlQueue1/2/3End (asistente y pasajero):** indica que el pasajero se encuentra en el final de una cola, por lo que procede a realizar la acción indicada, en este caso, procedería a pasar el control de seguridad.
- **pmrControlPassed (asistente):** se utiliza para indicar que el pasajero ha superado el

control de seguridad, por lo que puede continuar con la siguiente acción.

- **pmrPickedUp(asistente):** se utiliza para indicar en los asistentes, que ya se ha recogido a un PMR.

- **Planificador:** este es el corazón de GOAP, ya que es el encargado de juntar los tres componentes anteriores, e ir comprobando, mediante el algoritmo A*, cual es el camino más corto, es decir, el de menor coste, que nos lleva a completar el objetivo marcado al agente mediante la realización de las acciones descritas. Una vez está determinada la acción a llevar a cabo, solo quedaría mover al agente por el mapa hasta el punto donde se lleva a cabo la acción, y llevar a cabo la acción.

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

public class Node
{
    public Node parent;
    public float cost;
    public Dictionary<string, int> state;
    public GAction action;

    public Node(Node parent, float cost, Dictionary<string, int> allstates, GAction action)
    {
        this.parent = parent;
        this.cost = cost;
        this.state = new Dictionary<string, int>(allstates);
        this.action = action;
    }

    public Node(Node parent, float cost, Dictionary<string, int> allstates, Dictionary<string, int> beliefstates, GAction action)
    {
        this.parent = parent;
        this.cost = cost;
        this.state = new Dictionary<string, int>(allstates);
        foreach (KeyValuePair<string, int> b in beliefstates)
        {
            if (!this.state.ContainsKey(b.Key))
            {
                this.state.Add(b.Key, b.Value);
            }
        }
        this.action = action;
    }
}
```

Figura 30 Ejemplo de planificador en GOAP

```
public class GPlanner
{
    public Queue<GAction> plan(List<GAction> actions, Dictionary<string, int> goal, WorldStates beliefstates)
    {
        List<GAction> usableActions = new List<GAction>();

        foreach (GAction a in actions)
        {
            if (a.IsAchievable())
            {
                usableActions.Add(a);
            }
        }

        List<Node> leaves = new List<Node>();
        Node start = new Node(null, 0, GWorld.Instance.GetWorld().GetStates(), beliefstates.GetStates(), null);

        bool succes = BuildGraph(start, leaves, usableActions, goal);

        if (!succes)
        {
            // Debug.Log("No plan");
            return null;
        }

        Node cheapest = null;

        foreach (Node leaf in leaves)
        {
            if (cheapest == null)
                cheapest = leaf;
            else
            {
                if (leaf.cost < cheapest.cost)
                {
                    cheapest = leaf;
                }
            }
        }

        List<GAction> result = new List<GAction>();
        Node n = cheapest;

        while (n != null)
```

Figura 31 Ejemplo de planificador en GOAP

Dentro del planificador, la función más importante, es la llamada “BuildGraph” la cual se encarga de ir construyendo el grafo de acciones con el orden de las acciones que se llevan a cabo.

```
107     private bool BuildGraph(Node parent, List<Node> leaves, List<GAction> usuableActions, Dictionary<string, int> goal)
108     {
109         bool foundPath = false;
110         foreach (GAction action in usuableActions)
111         {
112             if (action.IsAchievableGiven(parent.state))
113             {
114                 Dictionary<string, int> currentState = new Dictionary<string, int>(parent.state);
115                 foreach (KeyValuePair<string, int> eff in action.effects)
116                 {
117                     if (!currentState.ContainsKey(eff.Key))
118                     {
119                         currentState.Add(eff.Key, eff.Value);
120                     }
121                 }
122 
123                 Node node = new Node(parent, parent.cost + action.cost, currentState, action);
124 
125                 if (GoalAchieved(goal, currentState))
126                 {
127                     leaves.Add(node);
128                     foundPath = true;
129                 }
130                 else
131                 {
132                     List<GAction> subset = ActionSubset(usuableActions, action);
133                     bool found = BuildGraph(node, leaves, subset, goal);
134 
135                     if (found)
136                     {
137                         foundPath = true;
138                     }
139                 }
140             }
141         }
142 
143         return foundPath;
144     }
```

Figura 32 Ejemplo de planificador en GOAP

4.3.2 NavMeshAgent

Tras la definición e implementación del modelo Goap, solo quedaría la implementación del sistema que permita a los agentes de la simulación, poder moverse por el escenario libremente, esquivando aquellos obstáculos que se encuentren por el camino, pero siempre con la premisa de buscar el camino más corto para ir del punto A al punto B.

Para ello, Unity, ofrece la herramienta llamada NavMeshAgent o lo que es lo mismo, Malla de Navegación de Agentes, la cual permite generar un grafo sobre la superficie del terreno, el cual indica las zonas por las que va a poder transitar cada uno de los agentes.

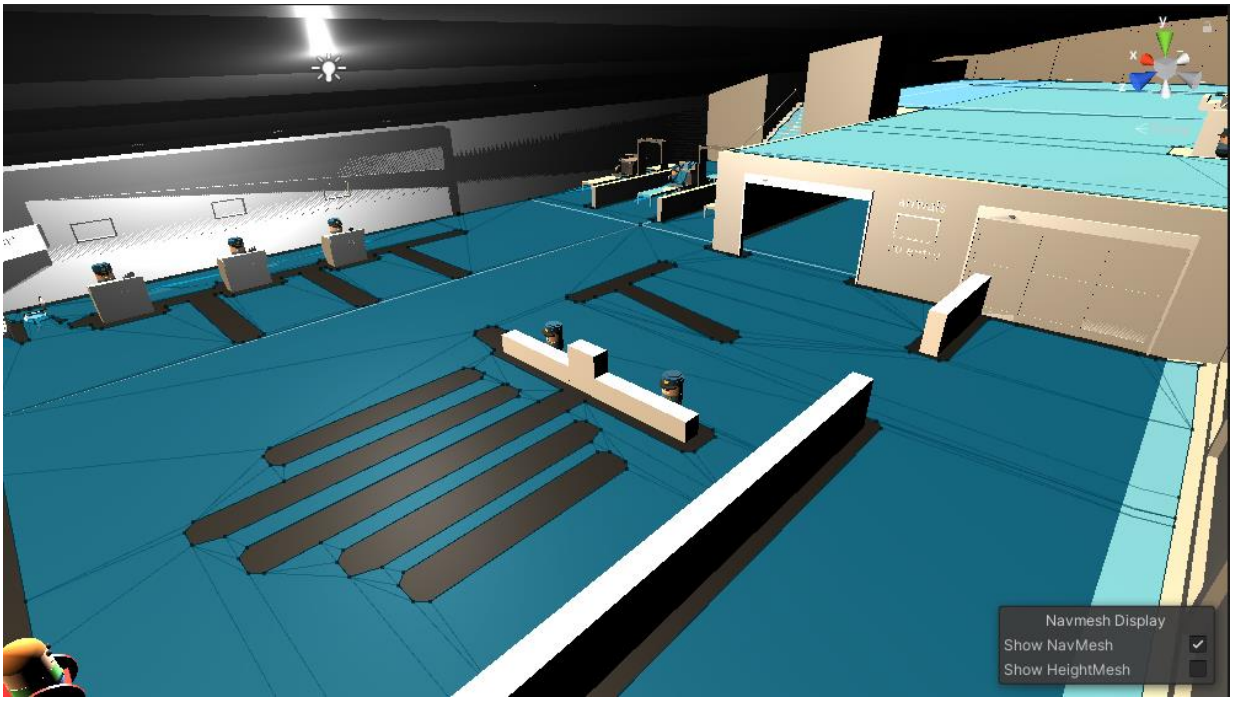


Figura 33 NavMeshAgent en Unity 3D

Tal y como se aprecia en la Figura 33 NavMeshAgent en Unity 3D, esa capa azul que se ve en el suelo es el grafo de navegabilidad por el que van a poder transitar los agentes. Además, podemos observar que existen una serie de zonas grises, las cuales representan una serie de zonas que he establecido como no navegables para que, en la simulación los personajes sigan unos caminos más naturales, similares a los que se tomarían en un aeropuerto real.

Para que todo esto sea posible, solo quedaría añadir al agente la opción de que se va a poder mover por una malla de navegación. Al añadir esta opción, el agente adopta una serie de variables con las que se va a definir el tamaño que ocupa el agente en la malla, además de la velocidad a la que se va a mover, la altura, los grados de inclinación que va a poder ascender, etc.

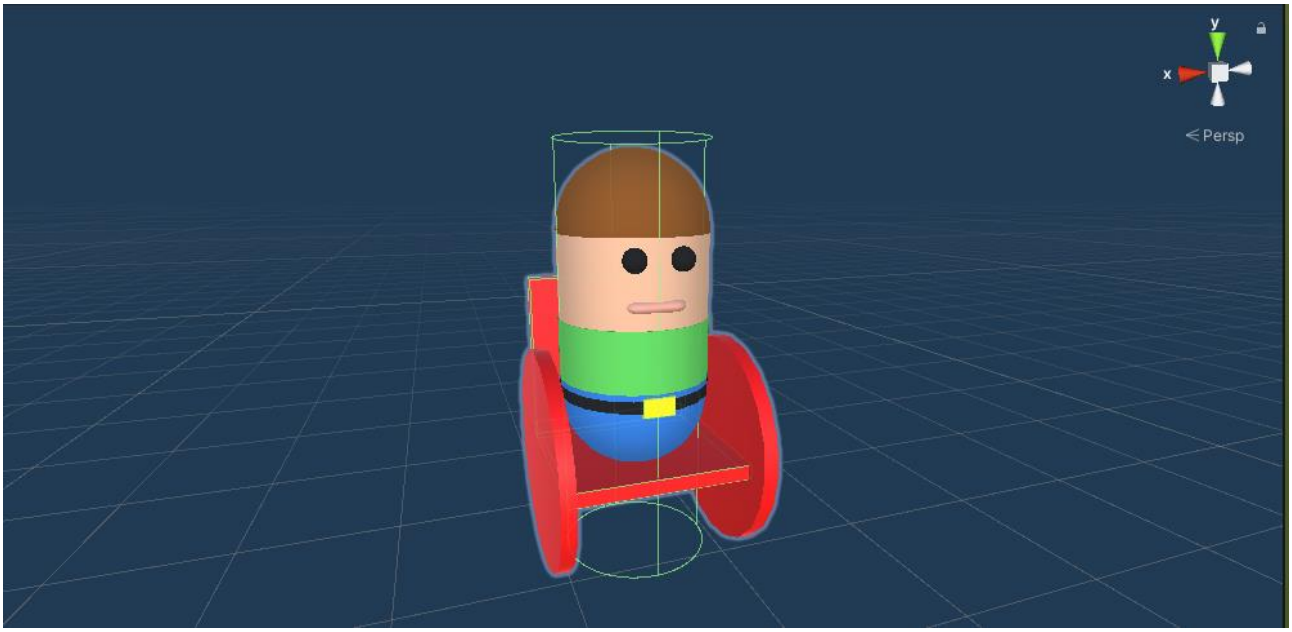


Figura 34 NavMeshAgent del agente 1

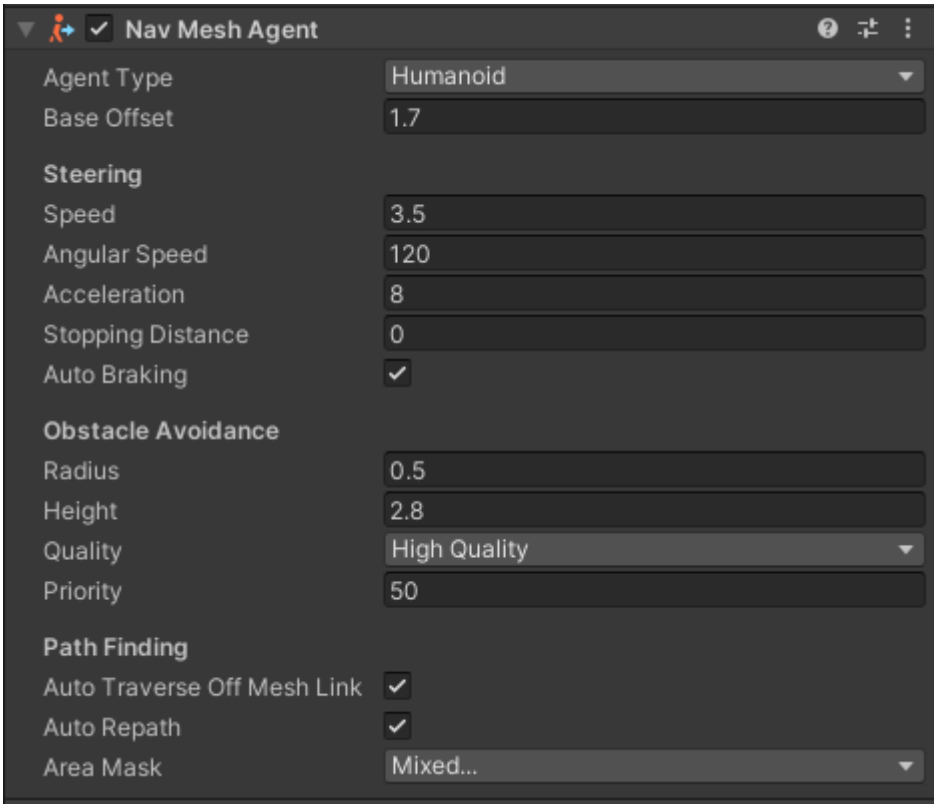


Figura 35 NavMeshAgent del agente 2

4.3.3 QueueManager

Para terminar con la explicación de la implementación llevada a cabo, solo me quedaría hablar del manejador de colas, que es el script encargado de controlar las colas que se encuentran repartidas en cada uno de los puntos de la simulación para que, cuando llegasen los pasajeros, estos se vayan encolando y sea más fácil de gestionar a la hora de determinar cuál es el primer agente en la cola para realizar la acción.

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

public class QueueManagerChecklIn : MonoBehaviour
{
    public List<GameObject> positions;
    public GameObject heapPosition;
    public string queueId;
    private List<GameObject> agents = new List<GameObject>();

    public void EnqueueAgent(GameObject agent)
    {
        agents.Add(agent);

        //indicar belief si está al principio de la cola
        if (agents.Count == 1)
        {
            agents[0].GetComponent<GAgent>().beliefs.ModifyState("at" + queueId + "End", 1);

            if (agents[0].GetComponent<GAgent>().asistenteAsignado != null)
            {
                agents[0].GetComponent<GAgent>().asistenteAsignado.GetComponent<GAgent>().beliefs.ModifyState("at" + queueId + "End", 1);
            }
        }
        else {
            if (agents.Count <= positions.Count) {
                Vector3 pos2 = positions[agents.Count - 1].transform.position;
                agent.GetComponent<GAgent>().actions[0].agent.SetDestination(pos2);
                if (agent.GetComponent<GAgent>().asistenteAsignado != null)
                {
                    agent.GetComponent<GAgent>().asistenteAsignado.GetComponent<GAgent>().actions[0].agent.SetDestination(pos2);
                }
            }
        }

    }

    public void DequeueAgent()
    {
        if (agents.Count == 0)
        {
            return;
        }

        GameObject agent = agents[0];

        agents.RemoveAt(0);

```

Figura 36 QueueManager

Tal y como se aprecia en la Figura 36 QueueManager, este script cuenta únicamente con dos funciones, una para encolar los agentes, la cual hace una comprobación. Si la cola está vacía, el agente que se quiere encolar es el primero en la cola, por lo que automáticamente pasa a realizar la acción objetivo, mientras que si, por el contrario, la cola no está vacía, entonces encolamos el agente, para que posteriormente vaya avanzando hasta llegar a la primera posición. Por otro lado, la segunda función, es la encargada de sacar al agente de la cola, además de ir avanzando el resto de los agentes que se encuentran dentro de la cola a la siguiente posición a ocupar dentro de la misma.

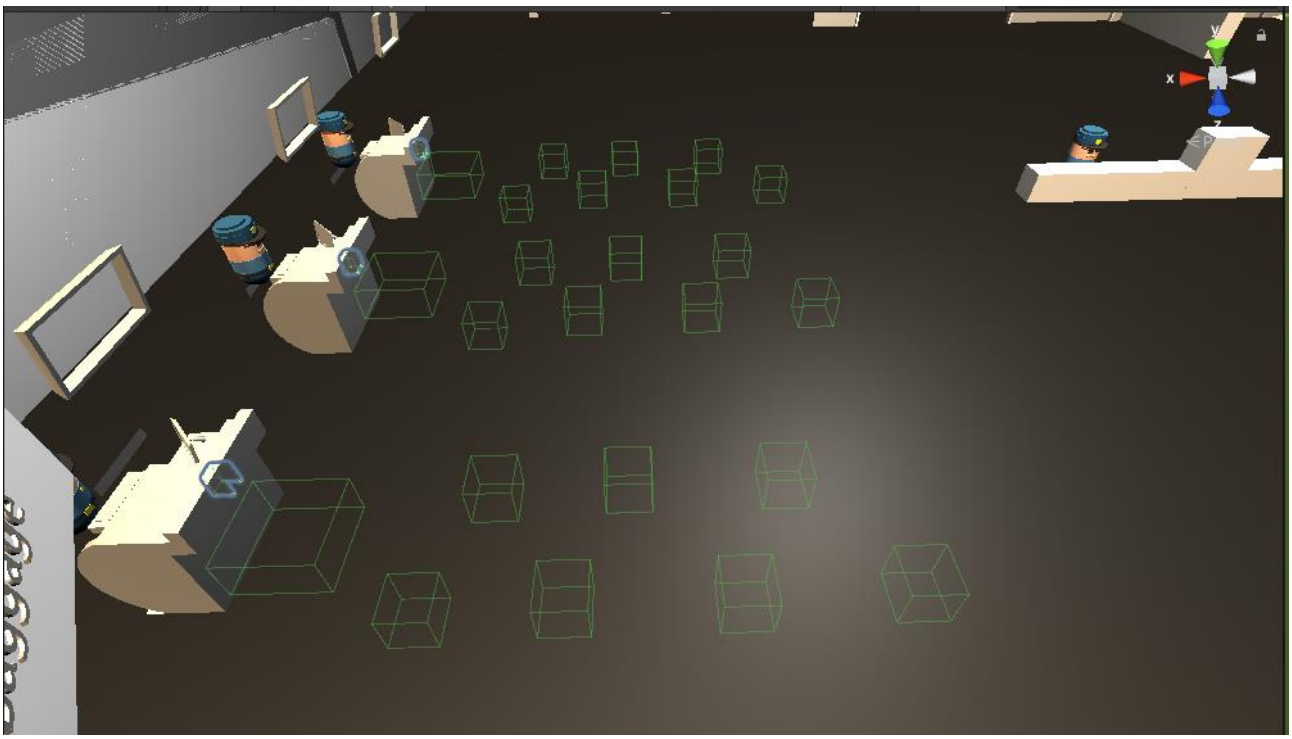


Figura 37 Colas de mostradores de facturación

4.4 Análisis de un caso de estudio con un aeropuerto

Como bien he adelantado en puntos anteriores, el objetivo principal del TFG ha sido el desarrollo de una simulación sobre la que poder probar diferentes estrategias con la que poder buscar una mejor solución a la situación que suponen los pasajeros de movilidad reducida en los aeropuertos.

Es por eso por lo que, tras la finalización de los puntos previos, para terminar con las tareas principales en las que había sido dividido el proyecto, solo queda llevar a cabo una pequeña simulación en la que se pondrá a prueba una posible estrategia que se podría aplicar en un aeropuerto.

En cuanto a la simulación, esta se ha desarrollado en un aeropuerto específico, con una estructura específica y con unas características concretas:

- La simulación contará con ambos tipos de pasajeros, PMR y no PMR, para mayor semejanza al funcionamiento de un aeropuerto real.
- Los pasajeros estarán llegando de manera continua al aeropuerto durante el tiempo de simulación.
- Para la estrategia seguida, se dispondrá de una cantidad de 10 asistentes.
- En esta estrategia el asistente estará junto al PMR durante toda la simulación.
- Se simulará que el pasajero ya se encuentra en silla de ruedas, por lo que no hará falta ir a buscarla, o que el asistente tenga que buscarla.
- Se simulará que cada 4-6 segundos llega un pasajero al aeropuerto.
- Los tiempos se expresan en horas - minutos - segundos (hh:mm:ss)

Para concluir, mostrará una tabla estadística de tiempos obtenidos en la simulación, tras un tiempo ejecutando, con las siguientes variables

- Cantidad de pasajeros en la simulación
- Tiempos de cola
- Tiempo de espera de PMR (tiempo que se encuentra esperando un PMR por un asistente)
- Tiempos de tránsito (recorrido de un punto a otro)
- Tiempo de acción (tiempo que tarda el pasajero en realizar la acción)
- Cantidad de PMR atendidos
- Porcentaje de uso de agentes (tiempo que pasa el agente con el pasajero)
- Tiempo de asistente ocioso (sin ningún pasajero asignado)
- Tiempo en sala de espera de un pasajero
- Tiempo total

Hay que tener en cuenta que los tiempos obtenidos son los basados en el aeropuerto en el que yo he realizado la simulación y durante un tiempo específico, donde los trayectos de un punto a otro están definidos por la estructura del aeropuerto. Los tamaños de las colas están definidos bajo mi criterio, es decir, soy yo el que ha establecido la cantidad de pasajeros por cola, además del tiempo que tarda en llegar un pasajero al aeropuerto.

Variable	Tiempo
Tiempo de simulación	**1:00:56**
Cantidad de pasajeros en la simulación	**410**
Cantidad de PMR atendidos	**369**
Tiempo de espera de PMR	**5:08:49**
Tiempos de cola	**1:43:46**
Tiempos de tránsito	**04:48:46**
Tiempo de acción	**03:01:25**
Tiempo de asistente ocioso	**02:55:30**
Porcentaje de uso de agentes	**83%**
Tiempo total	**17:36:48**

Tabla 1 Análisis de tiempo

En la tabla, se ha visualizado el sumatorio de todos los tiempos de cada uno de los pasajeros, de tal forma que, por ejemplo, el tiempo de espera de un pasajero por un asistente, es la suma del tiempo que ha esperado cada pasajero.

Capítulo 5
Conclusiones y líneas futuras

5.1 Conclusiones

En lo referente a mi trabajo, la idea inicial, era intentar llevar a cabo una simulación del ciclo de vida de un pasajero dentro de un aeropuerto, idea la cual ha sido cumplida bajo todos los puntos e implementaciones que se han comentado que sería adecuado haber añadido.

En un principio, el trabajo era llevar a cabo la simulación para los pasajeros de movilidad reducida, pero dado el avance con el que se llevó el trabajo, pudo añadir a la simulación también pasajeros que no presentan ningún tipo de movilidad reducida, lo cual le dio más vida a la simulación, además de darle más realismo, ya que los datos que se obtenían eran más variados y realistas, ya que, en un aeropuerto, la cantidad de pasajeros que pasan diariamente es bastante grande.

Respecto al resto de tareas, en mi caso el trabajo estaba dividido en cuatro, todas han sido cumplidas de manera satisfactoria, pasando por la realización del estudio de cargar archivos IFC, ya que esta tarea era un punto clave en el trabajo, puesto que todo partía de aquí, ya que, en el caso de no poder encontrar ninguna herramienta, tendría que haber modelado yo el aeropuerto, lo que habría retrasado el resto de las fases del trabajo. Terminada esta primera fase fue cuando se llevó a cabo la elaboración de los diferentes escenarios de trabajo. Hasta finalmente llegar a la última tarea donde se llevó a cabo el grueso del trabajo, siendo este la elaboración de la simulación en todas sus funcionalidades

Es por eso por lo que, como conclusión, el objetivo final a lograr sería que la mayor cantidad de aeropuertos existentes en la vida real, hicieran uso de la herramienta para mejorar esta situación por la que tienen que pasar este tipo de pasajeros. Dado que es esperable que grandes infraestructuras, como los aeropuertos, cuenten con los archivos de modelado de sus instalaciones físicas, resultaría de gran utilidad que pudieran llegar a trabajar con esta herramienta, ya que el único trabajo que tendrían que llevar a cabo, sería adaptar mi escenario y mi simulación, a lo suyo, modificando en esencia el modelo del aeropuerto y los tiempos promedio que tardan los pasajeros en llevar a cabo las diferentes tareas.

Desde un punto de vista más personal, la realización de este proyecto ha sido de gran satisfacción por varios motivos entre los que destacaría, trabajar con Unity, ya que es una herramienta con la que siento cómodo trabajando, principalmente por mi afición al mundo del desarrollo de videojuegos, y en segundo lugar también por trabajar con modelos GOAP, ya que es una herramienta que si se explota bien puede brindar muchísimas oportunidades como en mi caso, una especie de IA para videojuegos.

5.2 Líneas futuras

En cuanto al estado actual de la simulación, esta cuenta con las principales fases por las que pasa un pasajero que pretende realizar algún tipo de viaje en un aeropuerto. Pese a que cumple las funcionalidades necesarias, se deben de tener una serie de puntos en cuenta:

- En primer lugar, se encuentran las fases principales implementadas, pero sí que, en un aeropuerto, real, y con personas reales, no solo se van a producir estas fases, sino que se van a dar más situaciones tales como que el pasajero quiera ir al baño, o que dado un momento quiera tomar algo en la cafetería o visitar alguna tienda. Todas estas situaciones comentadas, son las denominadas interrupciones, es por eso, que, de cara a un futuro, sería un buen punto incluir estas interrupciones en la simulación, ya que darían un punto de realismo aún mayor del que ya proporciona la herramienta base.
- Como posible implementación a futuro, también cabría la posibilidad de implementar o evaluar nuevas tecnologías como sillas de ruedas autónomas, como solución alternativa a los asistentes de personas de movilidad reducida en aeropuertos. En esta solución, este tipo de sillas, tendrían un GPS, por el cual se guiarían a través del aeropuerto y de las diferentes zonas por las que tendría que ir, además de permitir tener ubicado al pasajero en todo momento, lo que facilitaría gran cantidad de los problemas existentes en la metodología de trabajo clásica (asistentes que llevan a los pasajeros).
- Ahora mismo, los tiempos de cada una de las actividades que tiene que realizar cada uno de los pasajeros, se introducen mediante el inspector, de tal manera que son invariables. En implementaciones futuras, sería deseable que, se pudiesen leer directamente de un archivo para que, si no hubiese que indicarlos uno a uno, sino poder cargarlos directamente.

Capítulo 6
Conclusions and Future Works

6.1 Conclusions

This TFG is part of a larger project, where another TFG from another partner is involved. Regarding my work, the initial idea was to try to carry out a simulation of the life cycle of a passenger inside an airport, an-idea that has been fulfilled under all the points and implementations that have been commented that it would be appropriate to have added.

At the beginning, the job was to carry out the simulation for passengers with reduced mobility, but given the progress with which the work was carried out, he was able to add to the simulation also passengers who do not have any reduced mobility, which gave more life to the simulation, in addition to giving it more realism, since the data that were obtained were more varied and realistic, since, in an airport, the amount of passengers that pass daily is quite large.

Regarding the rest of the tasks, in my case the work was divided into four, all have been fulfilled satisfactorily, going through the realization of the study of loading IFC files, since this task was a key point in the work, since everything started from here, because, if I could not find any tools, I would have had to model the airport, which would have delayed the rest of the work phases. This first phase was completed when the elaboration of the different work scenarios was carried out. Until finally arriving at the last task where the bulk of the work was carried out, being this the elaboration of the simulation in all its functionalities

That is why, as a conclusion, the ultimate goal to achieve would be that the largest number of airports in real life, make use of the tool to improve this situation that these types of passengers have to go through. Since each country has its own airport modeling files, it would be very useful if they could come to work with this tool, since the only work they would have to do would be to adapt my scenario and my simulation, to do their thing, essentially modifying the model of the airport and the average times it takes passengers to carry out the different tasks.

From a more personal point of view, the realization of this project has been of great satisfaction for several reasons among which I would highlight, working with Unity, since it is a tool with which I feel comfortable working, mainly because of my fondness for the world of video game development, and secondly also for working with GOAP models, since it is a tool that if exploited well can offer many opportunities as in my case, a kind of AI for video games.

6.2 Future works

As for the current state of the simulation, it has the main phases through which a passenger who intends to make some kind of trip in an airport passes. Although it fulfils the necessary functionalities, a number of points must be taken into account:

- First, there are the main phases implemented, but if, in an airport, real, and with real people, not only will these phases occur, but there will be more situations such as the passenger wants to go to the bathroom, or that given a moment you want to have a drink in the cafeteria or visit a store. All these situations discussed, are the so-called interruptions, that is why, for a future, it would be a good point to include these interruptions in the simulation, since they would give an even greater point of realism than the base tool already provides.
- As a possible future implementation, there would also be the possibility of implementing automatic wheelchairs, as an alternative solution for assistants of people with reduced mobility in airports. In this solution, these types of chairs would have a GPS, which would guide them through the airport and the different areas through which they would have to go, in addition to having the passenger located at all times, which would facilitate a lot of existing problems in the classical working methodology (attendants carrying passengers).
- Right now, the times of each of the activities that each of the passengers has to perform, are introduced through the inspector, in such a way that they are invariable, by this I mean that for a future implementation, it would be good to adapt those data, and that instead of having to indicate them in the inspector, they could be read directly from a file so that if from the other work or external information, it was not necessary to indicate them one by one, but to be able to load them directly.

Capítulo 7
Presupuesto

En este apartado procederé a comentar el presupuesto que acarrearía llevar a cabo este proyecto, fuera de un ámbito universitario, más orientado al mundo laboral, donde ya no es un trabajo de fin de grado, sino que es un país, quien quiere llevar a cabo este estudio para su aeropuerto.

En primer lugar, en mi caso, he tenido que realizar un curso de GOAP en Unity 3D, ya que mis conocimientos de GOAP en Unity, eran prácticamente nulos, por lo que, si se quiere considerar el precio del curso para incluir en el presupuesto, este sería de 15€.

En segundo lugar, hay que recordar que, a la hora de las herramientas utilizadas para trabajar, yo he usado las versiones de prueba que ofrecía cada una de ellas (las herramientas para cargar los archivos del aeropuerto en Unity, recordar que Unity es gratis). Por lo que, si nos encontramos en un proyecto real, más grande y con más presupuesto, existe la posibilidad de adquirir las licencias de la herramienta a utilizar para poder trabajar con total libertad, recordar que los precios ascienden entre 600€ hasta los 2000€, tal y como se ha visto en el punto 2, donde se llevó a cabo el estudio pertinente.

Por último, solo quedaría el precio asociado a los recursos humanos involucrados en el trabajo, donde entran factores como la cantidad de horas dedicadas al proyecto, los recursos disponibles, presupuesto del proyecto, etc.

Tipos	Horas de trabajo	Coste
Planificación y estructuración del desarrollo	20	500€
Estudio y obtención de datos promedio del aeropuerto real	40	1000€
Desarrollo del modelo para la simulación	80	2400€
Testeo y análisis del modelo	20	500
Licencias de terceros	0	900€
Total	160	5300€

Tabla 2 Presupuestos

Capítulo 8
Bibliografía

[1] Goal-Oriented Action Planning. http://alumni.media.mit.edu/~jorkin/goap.html

[2] What is GOAP? https://gamedevelopment.tutsplus.com/tutorials/goal-oriented-action-planning-for-a-smarter-ai--cms-20793

[3] Udemy. Cursos online. https://www.udemy.com/

[4] Unity Scripting Reference. https://docs.unity3d.com/ScriptReference/

[5] Advanced AI For Games with Goal-Oriented Action Planning. Peny. https://www.udemy.com/course/ai_with_goap/

[6] An introduction to Goal Oriented Action Planning. Holistic3. https://www.youtube.com/watch?v=jUSrVF8mve4&ab_channel=Holistic3d

[7] Unity Reflect. https://unity.com/es/products/unity-reflect

[8] Tridify. https://www.tridify.com/

[9] Two-level simulation approach to evaluate the performance of autonomous wheelchairs. https://journals.sagepub.com/doi/abs/10.1177/0037549720941609

[10] Goal-Oriented Action Planning. http://alumni.media.mit.edu/~jorkin/goap.html