Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Desarrollo básico de Simulador de Control de Tráfico Aéreo (ATC)

Autor: J. Eduardo de Balle Pardo

Tutor: Jorge J. Fernández de la Cruz

Dpto. de Aeronáutica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Desarrollo básico de Simulador de Control de Tráfico Aéreo (ATC)

Autor:
J. Eduardo de Balle Pardo

Tutor:
Jorge J. Fernández de la Cruz
Profesor Titular

Dpto. de Aeronáutica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Grado: Desarrollo básico de Simulador de Control de Tráfico Aéreo (ATC)

Autor: J. Eduardo de Balle Pardo
Tutor: Jorge J. Fernández de la Cruz

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

A toda mi familia, en especial a mi Madre, la cual me ha apoyado y dado los medios necesarios para poder completar esta tarea. También en especial a mi Abuelo, José Luis Pardo González, Doctor Ingeniero Aeronáutico de la promoción del 55 en España, y mi mayor motivación para convertirme en Ingeniero Aeronáutico como él, que Descanses en Paz

A mis amigos y compañeros, quienes me han dado los ánimos y las fuerzas necesarios para continuar con el trabajo, y en especial aquellos que me han podido dar consejos en el ámbito de programación

A toda la escuela, la cual me ha dado los conocimientos y la oportunidad de realizar este trabajo. En especial a mi Tutor, Jorge Fernandez de la Cruz, sin el cual este trabajo no habría sido posible y quien ha tenido la paciencia suficiente conmigo. También al Profesor Alfonso Valenzuela Romero, cuya docencia en la asigatura de Gestión del Tráfico Aéreo fue una inspiración importante para la realización de este trabajo

A todos estos y muchos más que me han apoyado en este viaje, muchas gracias

Resumen

El Control de Tráfico Aéreo (ATC en adelante) es uno de los pilares fundamentales de la Navegación Aérea, y muy poca gente fuera del mundo Aeronáutico sabe cómo funciona. Es por ello que en este Trabajo se va a desarrollar una aplicación para acercar el mundo del ATC a gente que no esté tan familiarizada con él y quiera aprender. Además también sirve para poner en práctica los conocimientos adquiridos en el Grado con respecto a este tema.

Esta aplicación se ha desarrollado mediante el desarrollador de aplicaciones de MATLAB® , pues este lenguaje de programación da mucho juego para hacer cálculos grandes y complejos, que normalmente suelen ser necesarios para la simulación de vuelo de aeronaves. Además de que se ha usado para el propósito de calcular trayectorias de aeronaves a lo largo del Grado.

Al usar MATLAB® como lenguaje para esta aplicación, y al no haber datos de simulación de ATC adaptados a MATLAB® , se ha desarrollado en esta misma aplicación un Editor de Escenarios con el que crear un archivo de datos compatible con el Simulador desarrollado.

Como referencia se han considerado tres Simuladores de ATC que se usan actualmente, el primero que se ha investigado es el de RAMS Plus por ISA Software, el cual está más orientado a hacer simulaciones para calcular la capacidad de un puesto de Control, además de hacer otros estudios de análisis. El segundo es BlueSky ATC, de código abierto y desarrollado por la Universidad de Delft (Países Bajos), también tiene un enfoque más analítico aunque es menos profesional y más accesible, además de similar al que se ha desarrollado en este proyecto, en el sentido de que nacen a partir de un proyecto de una Universidad. El último que se ha investigado es el simulador Aurora de IVAO, el cual permite conectarse tanto como controlador como piloto e interpretar dichos roles en un entorno virtual que se asemeja mucho al de un puesto de Control real. Este último es básicamente lo que se quiere desarrollar con este proyecto (que se ha decidido llamar ATC Maker), con la diferencia de que ATC Maker se use exclusivamente en modo solitario, mientras que el de Aurora se utiliza en modo “multijugador”.

Figura 1 Logos de los Simuladores.

Pese a lo dicho en el párrafo anterior, sí que se pretende que pueda haber un modo “multijugador” para ATC Maker, en un sentido distinto al de Aurora, y que parte de la inspiración del nombre de ATC Maker, el videojuego “Mario Maker”. El videojuego Mario Maker consiste en un Editor de Escenarios para Pantallas del juego de Super Mario, estos escenarios se pueden almacenar en un servidor on-line y desde ahí otros jugadores se los pueden descargar y jugarlos. Algo similar podría suceder con ATC Maker, pues se ha basado un poco en ese concepto, el de crear Escenarios y Simularlos.

Una vez explicado un poco la base de la que se parte para el desarrollo de ATC Maker, caben destacar ciertas limitaciones que se han impuesto para que fuera factible desarrollar esta aplicación como un Trabajo de Fin de Grado, pues hacer una aplicación de este estilo suele llevar años de desarrollo por grandes equipos

Figura 2 Logo de ATC Maker.

de programadores, y este es un proyecto de aproximadamente un año de duración realizado por una sola persona. Por ello hay que definir estándares realistas, y establecer unos objetivos que sean posibles dentro de los límites que implica el que esto sea un TFG.

El objetivo principal entonces es desarrollar una aplicación funcional que pueda crear y Editar escenarios de Simulación de ATC y luego Simularlos. Para que esto sea más factible entonces se van a determinar unos límites a las posibilidades que se van a permitir en esta aplicación.

Primero de todo se va a considerar condiciones meteorológicas de cielos despejados y viento en calma, el modelaje meteorológico en MATLAB® se podría calificar como un TFG en sí mismo, de modo que se toman como condición de partida estas condiciones para reducir los parámetros de Simulación. Adicionalmente, solo se va a considerar un tipo de control, el Control de Área, el cual solo tiene en cuenta a Aeronaves en Ruta, las cuales suelen ir en régimen de crucero y por ende varían poco su altitud y velocidad, además de que los procesos de Aproximación, Despegue y Aterrizaje son bastante complejos en sí y requieren de mucha intervención por parte del Controlador, lo cual hace que Simularlos sea excesivamente complicado. Además de esto se va a considerar solo un tipo de aeronave, pues modelar las actuaciones de aeronaves también es algo digno de un TFG en sí mismo, los datos que se usan para las actuaciones que se vayan a tener en cuenta son los de un A320. Por último, para las consideraciones de distancias mínimas de separación de aeronaves, se considera que se dispone de multiradar y que siempre se está en una región con RVSM (Reduced Vertical Separation Minima), de manera que las distancias mínimas son fijas en todo momento.

Con límites y expectativas definidos, se procede a desarrollar el programa, el cual se explica en detalle en el cuerpo de este documento, pero a en términos generales se tiene lo siguiente. En el Editor se determinan los distintos parámetros del Sector que se va a controlar, los cuales son: El nombre del Escenario, el cual determina el nombre del archivo .mat en el que se almacenan todos los datos del escenario, el Tiempo de Simulación, determinado como las horas de inicio y final de la Simulación, La Frontera del Sector, Las Zonas Restringidas dentro del Sector, los Puntos de Paso y Aeropuertos del Sector y, lo más importante, las Aeronaves que van a Volar en el Sector durante la Simulación.

Dentro de las posibilidades de actuación de los Aeronaves, se han considerado 4: que sigan el Plan de Vuelo Establecido sin variaciones, que soliciten un Directo, que soliciten un Cambio de Nivel de Vuelo o que tomen un rumbo determinado que no consta en el Plan de Vuelo sin consultar. Estos supuestos no se habían mencionado anteriormente pero se tienen muy en cuenta en el desarrollo de la aplicación de ATC Maker.

Y con esto termina el Resumen, se ampliará el contenido de lo contado aquí en los respectivos Apartados del Cuerpo del Documento.

Abstract

Air Traffic Control (ATC from now on) is one of the fundamental pillars of Air Navigation, and very few people outside the Aeronautical World know how it works. That is why in this Work an application is going to be developed to bring the world of ATC closer to people who are not so familiar with it and want to learn. In addition, it also serves to put into practice the knowledge acquired in the Degree with respect to this topic.

This application has been developed using the MATLAB® application developer, as this programming language gives a lot of scope for large and complex calculations, which are normally necessary for aircraft flight simulation. In addition to that it has been used for the purpose of calculating aircraft trajectories along the Grade.

By using MATLAB® as the language for this application, and as there is no ATC simulation data adapted to MATLAB® , a Scenario Editor has been developed in this same application with which to create a Data file compatible with the developed Simulator.

As a reference, three ATC Simulators that are currently used have been investigated. The first one that has been investigated is the RAMS Plus Simulator by ISA Software, which is more oriented towards doing simulations to calculate the capacity of a Control post, in addition to doing other analysis studies. The second is BlueSky ATC, open source and developed by the University of Delft, it also has a more Analytical approach although it is less professional and more accessible, as well as similar to the one that has been developed in this project, in the sense that they are born to from a university project. The last one that has been investigated is IVAO's Aurora simulator, which allows you to connect both as a controller and as a pilot and interpret these roles in a virtual environment that closely resembles that of a real Control post. The latter is basically what we want to develop with this project (which has been called ATC Maker), with the difference that ATC Maker is used exclusively in solo mode, while Aurora's is used in "Multiplayer" mode.

Figura 3 Logos de los Simuladores.

Despite what was said in the previous paragraph, it is intended that there may be a "Multiplayer" mode for ATC Maker, in a different sense from that of Aurora, and that part of the inspiration for the name of ATC Maker, the video game "Mario Maker ". The Mario Maker video game consists of a Scenario Editor for Screens of the Super Mario game, these scenarios can be stored on an online server and from there other players can download and play them. Something similar could happen with ATC Maker, since it has been based a bit on that concept, that of creating Scenarios and Simulating them.

Once the basis for the development of ATC Maker has been explained a little, certain limitations that have been imposed to make it feasible to develop this application as a Final Degree Project should be highlighted, since making an application of this style usually take years of development by large teams of programmers,

Figura 4 Logo de ATC Maker.

and this is about a year-long project done by one person. For this reason, realistic standards must be defined, and objectives set that are possible within the limits implied by this being a TFG.

The main objective then is to develop a functional application that can create and Edit ATC Simulation scenarios and then Simulate them. In order for this to be more feasible, some limits will be determined to the possibilities that will be allowed in this application.

First of all, clear and calm weather will be considered,
matlab weather modeling could be described as a TFG in itself, so this is limited to reduce the Simulation parameters. Additionally, only one type of control will be considered (the three types of Control are explained later, in the body of the Work), Area Control, which only takes into account En-Route Aircraft, which usually go in cruise regime and therefore their altitude and speed vary little, in addition to the fact that the Approach, Takeoff and Landing processes are quite complex in themselves, which makes Simulating them problematic. In addition to this, only one type of aircraft will be considered, since modeling aircraft actions is also something worthy of a TFG in itself, the data used for the actions to be taken into account are those of an A320 . Lastly, for considerations of minimum aircraft separation distances, it is considered that multiradar is available and that it is always in a region with RVSM (Reduced Vertical Separation Minima), so that the minimum distances are fixed at all times.

With defined limits and expectations, we proceed to develop the program, which is explained in detail in the body of this document, but in broad strokes we have the following. In the Editor, the different parameters of the Sector to be controlled are determined, which are: The name of the Scenario, which determines the name of the .mat file in which all the data of the scenario is stored, the Simulation Time, determined as the start and end times of the Simulation, the Sector Border, the Restricted Zones within the Sector, the Waypoints and Airports of the Sector and, most importantly, the Aircraft that will Fly in the Sector during the Simulation .

Within the possibilities of action of the aircraft, 5 have been considered: that they follow the Established Flight Plan without variations, that they request a Direct Flight, that they request a Flight Level Change, that they take a certain course that does not appear in the Plan of Flight without consulting or that are in a conflict (whether of violation of minimum separation or penetration into unauthorized airspace) and must act to avoid it following the instructions of the Controller. These limits were not previously mentioned but are carefully considered in the development of the ATC Maker application.

And with this the Abstract ends, the content of what is told here in the respective Sections of the Document Body will be expanded.

... -translation by google-

Índice Abreviado

Índice

Notación

5LNC	5-Letter Name Code
A320	Modelo de avión de Airbus comercial (uno de los más comunes con el modelo de Boeing-737)
AIS	Servicio de Información Aeronáutica (Aeronautical Information Service)
Alfabeto Radiofónico	Alfabeto que se usa en comunicaciones por radio diseñado para que cada caracter sea claramente distinguible de otros caracteres. Este consiste en asociar palabras reconocibles a cada letra, en concreto, para ABCD: Alfa, Bravo, Charlie, Delta respectivamente.
α	Ángulo denominado Alfa
ALRS	Servicio de Alertas de Tránsito de Aéreo (Alert Service)
ANS	Servicios de Navegación Aérea (Air Navigation Systems)
archivo .mat	Formato de archivos binarios usado por MATLAB® que puede almacenar matrices, variables, funciones y otros tipos de datos
ASCII	American Standard Code for Information Interchange. Formato de codificación de caracteres para datos de texto más común en ordenadores
ASM	Gestión del Espacio Aéreo (Air Space Management)
ATC	Control del Tránsito Aéreo (Air Traffic Control)
ATF	Flujos de Tránsito Aéreo (Air Traffic Flow)
ATFM	Gestión de la Afluencia del Tránsito Aéreo (Air Traffic Flow Management)
ATM	Gestión del Tránsito Aéreo (Air Traffic Management)
ATS	Servicios de Tránsito Aéreo (Air Traffic Services)
ATZ	Aeródromo Controlado (Aerodrome Traffic Zone)
AWY	Aerovía (Airway)
BADA	Base of Aircraft DAta
CallbackFcn	Función de Respuesta, fragmento de código que se ejecuta después de que el Usuario interactúe de ciertas formas concretas con el objeto que da nombre a la Función. Por ejemplo: pulsar un botón, editar un Campo de Texto, etc...
cell	Tipo de variable de MATLAB® con el que se pueden almacenar varios tipos de variables distintos con dimensiones distintas
char	Tipo de variable de MATLAB® que sirve para almacenar caracteres en vectores del mismo modo que se almacenan datos numéricos en vectores o matrices
CNS	Servicio de Comunicaciones Navegación y Vigilancia (Communications Navigation and Surveillance)
CTA	Área de Control (Controlled Traffic Area)
CTR	Zona de Control (Controlled Traffic Region)
Δt	Paso de Simulación (Incremento de tiempo entre tomas de datos de Simulación)
double	Tipo de variable de MATLAB® que almacena datos numéricos como valores de punto flotante de doble precisión de 64 bits (8 bytes). Este es el tipo de variable por defecto de MATLAB® y proporciona precisión suficiente para la mayoría de tareas computacionales
ETO	Tiempo Estimado de Entrada al Sector (Estimated Time Over)

FAA	Federal Aviation Administration
Ficha de Progresión de Vuelo	Herramienta que usan los Controladores y que contienen los datos básicos y necesarios de un Vuelo para realizar efectivamente el Control. Los datos que incluye suelen ser: el Identificador de la Aeronave, el Nivel de Vuelo Autorizado para ese Vuelo, la Ruta que va a seguir el Vuelo, su ETO, entre otros.
find	Comando de MATLAB® que devuelve los índices de la matriz que se introduce como argumento de entrada en los que se tiene un valor lógico verdadero (o, en el caso de que se introduzca una condición lógica, aquellos índices en los que se cumpla dicha condición)
FIR	Región de Información de Vuelo (Flight Information Region)
FIS	Servicio de Información de Vuelo (Flight Information Service)
FL	Nivel de Vuelo (Flight Level), un Nivel de Vuelo se calcula dividiendo la Altitud en pies (ft) entre 100
ft	Pies (feet), unidad de longitud 1 ft = 0.3048 m
IFR	Reglas de Vuelo Instrumental (Instrumental Flight Rules)
ISA	Innovation for Sustainable Aviation
IVAN	IVAO Network
IVAO	International Virtual Aviation Organization
kt	Nudos (kt), unidad de velocidad 1 kt = 1 nm/h
libapp	Carpeta que actúa como librería de la aplicación, donde se almacenan los archivos necesarios para usar la Aplicación de ATC Maker
MATLAB®	Lenguaje de Programación que maneja principalmente variables en formato de matrices, de manera que se pueden realizar cálculos matemáticos más complejos que con otros lenguajes de programación
MET	Servicio de Información Meteorológica
METAR	Informe sobre las Condiciones Meteorológicas que se publica regularmente
min	Minutos
NASA	National Aeronautics and Space Administration
NextGen	Next Generation Air Transportation System, programa de mejora y modernización del Sistema del Espacio Aéreo de Estados Unidos, realizado por la FAA
NLR	Netherlands Aerospace Centre
nm	Milla náutica (nautic mile), unidad de longitud 1 nm = 1.852 km
OACI/ICAO	Organización de Aviación Civil Internacional (International Civil Aviation Organization)
plot	Comando de MATLAB® que representa datos numéricos en una figura con unos ejes cartesianos, con unas especificaciones de línea que se pueden indicar. Cuando se utiliza en una Aplicación de MATLAB® se debe especificar el objeto de ejes en el que se quiere representar los datos como primer argumento de entrada
Python	Lenguaje de Programación de Alto Nivel para uso general muy versátil para todo tipo de programas
QNH	Reglaje de referencia para la medición de altitud por Altímetros barométricos, que se usa para determinar la Altura relativa con respecto a un Aeropuerto
Raíl de Fichas	Espacio de un puesto de Control en el que los Controladores pueden colocar las Fichas de Progresión de Vuelo en orden de Altitud para poder hacer un seguimiento rápido de los posibles conflictos en el Sector
RAMS	Reorganised ATC Mathematical Simulator
RNAV	Método de Navegación Aérea de por IFR que permite a las aeronaves operar en cualquier rumbo deseado dentro de la cobertura de de las radioayudas o dentro de los límites de un sistema capaz de autocontenerse, o combinando ambas. El nombre proviene del nombre 'Navegación Aleatoria'(Random NAVigation)
Ruta FMS	Ruta almacenado en el FMS (Flight Management System)
RVSM	Reducción de distancias Verticales Mínimas de Separación (Reduced Vertical Separation Minimum)
SAR	Servicio de Búsqueda y Salvamento (Search and Rescue)

SESAR	Cielo Único Europeo (Single European Sky ATM Research), programa de mejora y modernización de la Gestión del Tránsito Aéreo en el Espacio Aéreo Europeo cuya premisa principal es unificar los proveedores de Servicios en uno solo para evitar duplicidades y sectorización muy pequeña
SIDs	Cartas de Navegación que determinan Rutas específicas para salidas de aeropuertos por vuelos IFR (Standard Instrument Departure)
STARs	Cartas de Navegación que determinan Rutas específicas para llegadas a aeropuertos por vuelos IFR (Standard Terminal Arrival Route), estas incluyen circuitos de espera
struct	Tipo de Variable de MATLAB® que permite almacenar datos en forma de directorio, añadiendole propiedades a una variable original, a las que se puede llamar con la sintaxis: Variable_Original.Propiedad. En todos sus campos se pueden tener dimensiones distintas de 1
TMA	Área de Control Terminal (Terminal Control Area)
TMX	Herramienta de Simulación desarrollada por el NLR para estudiar el concepto de Vuelo Libre
text	Comando de MATLAB® que representa un fragmento de texto en una posición relativa indicada con respecto a un punto determinado especificado con la sintaxis del comando plot
UTC	Tiempo Universal Coordinado (Coordinated Universal Time), es el principal estándar de tiempo mediante el que se regulan los relojes mundiales, el cual está dentro de un margen de error con respecto al tiempo solar medio en longitud 0º y no se ajusta a los horarios de ahorro de tiempo diurno. Efectivamente es el sucesor del Greenwich Mean Time (el tiempo solar en el Meridiano de Greenwich)
VHF	Banda de Frecuencias de Radio de Muy Alta Frecuencia (Very High Frecuency), donde se realizan la mayoría de comunicaciones Aeronáuticas
VOR	Radiofaro Omnidireccional de Muy Alta Frecuencia (VHF Omnidirectional Range)

1 Introduccón

Para entender el ATC hay que verlo en su contexto, pues el ATC es solo uno de una gran red de Servicios de Navegación Aérea (ANS en adelante) que posibilitan la circulación de las aeronaves que componen el tráfico aéreo de un punto del espacio a otro. Entre estos ANS se incluyen: los Servicios de Comunicaciones Navegación y Vigilancia (CNS), que se ocupan de gestionar y mantener las infraestructuras de ayuda a la navegación aérea; los Servicios Meteorológicos para la navegación aérea (MET), que se encarga de realizar y publicar las previsiones meteorológicas para el espacio aéreo; los Servicios de Búsqueda y Salvamento (SAR), que se encargan de movilizar los recursos de búsqueda y salvamento en caso de accidentes de aeronaves; los Servicios de Información Aeronáutica (AIS), que se encargan de proporcionar la información y los datos aeronáuticos para la seguridad, regularidad y eficiencia de la navegación aérea; y por último, los Servicios de Gestión del Tránsito Aéreo (ATM), dentro del cual se incluye el ATC y que se desarrollará en profundidad a continuación.

1.1 Gestión del Tránsito Aéreo

Figura 1.1 Logo de OACI.

[3][4]La Gestión del Tránsito Aéreo (ATM), es la administración, dinámica e integrada (entendiendo por integrada como segura, económica y eficiente) del espacio aéreo y del tránsito aéreo, donde el tránsito aéreo lo componen todas las aeronaves en vuelo y en el área de maniobras (conformada por las pistas y las calles de rodadura, pero no la plataforma). Esta incluye los siguientes servicios:

- Gestión de Afluencia del Tránsito Aéreo (ATFM): Se encarga de gestionar el equilibrio entre la demanda y la capacidad del espacio aéreo.
- Gestión del Espacio Aéreo (ASM): Se encarga de maximizar la utilización del espacio aéreo disponible mediante la compartición dinámica del mismo o la segregación del mismo entre las categorías de usuarios.
- Servicios de Tránsito Aéreo (ATS): entre los que se incluyen los servicios de Información de Vuelo (FIS), Alerta (ALRS) y ATC.

[2][4]Dentro de los servicios que se incluyen en el ATM nos interesan los ATS. Estos se encargan de cumplir una serie de objetivos, que consisten en:

1. Prevenir colisiones entre aeronaves
2. Prevenir colisiones entre aeronaves en el área de maniobras, y entre aeronaves y obstáculos de que haya en dicha área.
3. Acelerar y mantener ordenadamente el movimiento del tránsito aéreo.
4. Asesorar y proporcionar información útil para la marcha segura y eficaz de los vuelos.
5. Notificar a los organismos pertinentes respecto a las aeronaves que necesitan ayuda de búsqueda y salvamento, y auxiliar a dichos organismo cuando sea necesario.

Estos objetivos se cumplen mediante tres servicios principales englobados dentro de los ATS:

- Servicio de Información de Vuelo (FIS): para satisfacer el objetivo 4.
- Servicio de Alerta (ALRS): para satisfacer el objetivo 5.
- Servicio de Control de Tránsito Aéreo (ATC): el que nos ocupa y que satisface los objetivos 1, 2, 3. Este se subdivide en tres servicios distintos dependiendo del tramo de vuelo al que dan servicio:
  - Servicio de control de área: Provee servicio de ATC a los vuelos controlados, a excepción de los casos descritos en los siguientes servicios. Satisface los objetivos 1 y 3.
  - Servicio de control de aproximación: Provee servicio de ATC a aquellas partes de vuelos controlados relacionados con la llegada o salida. Satisface los objetivos 1 y 3.
  - Servicio de control de aeródromo: Provee servicio de ATC al tránsito de aeródromo (todo el tránsito en el área de maniobras y en las inmediaciones del aeródromo). Satisface los objetivos 1, 2 y 3.

La facilitación de los ATS lleva a tener que distinguir el espacio aéreo en designaciones según los servicios que se prestan, estas Designaciones de Espacio Aéreo incluyen:

- **Región de Información de Vuelo** (Flight Information Region, FIR): Aquellas partes del espacio aéreo en las que se facilita servicio de Información (FIS) y Alerta (ALRS). Estas normalmente abarcan la totalidad del espacio aéreo sobre el territorio de un Estado, las fronteras entre distintas FIR deben ser contiguas y, a ser posible, su geometría debe atender a aspectos operacionales antes que a hacer coincidir fronteras. Pueden existir subdivisiones en varias FIR del territorio estatal, que dependerán de las rutas, la topografía, la relación coste-eficacia y las capacidades de las dependencias ATS que prestan servicio. En ocasiones puede existir una división vertical, siendo la FIR la parte inferior y siendo la parte superior UIR (Upper Flight Information Region).
- **Áreas de Control** (Control Area, CTA): Aquellas partes del espacio aéreo en las que se facilita el servicio de ATC desde un límite inferior especificado hacia arriba. Estas deben establecerse de manera que abarquen las trayectorias de los vuelos IFR, o partes de las mismas, a las que se desee facilitar los pertinentes servicios de ATC. Los límites verticales de una CTA se establecen de manera que los límites inferiores no queden en ningún momento por debajo de los 200 m (700 ft) sobre el nivel del mar, aunque se recomienda dejar un límite inferior superior para dar más libertad a la circulación de vuelos VFR que no suelen requerir de los servicios de ATC. Por otro lado los límites superiores se establecen a partir de donde se deja de facilitar servicios de ATC o cuando la CTA se sitúe por debajo de una región de control superior, en cuyo caso el límite superior de la CTA coincidirá con el límite inferior de la región de control superior. Los límites verticales por encima de los 900 m (3000 ft) sobre el nivel del mar deberán coincidir con un nivel de crucero VFR. Existen algunos tipos de CTA particulares:
  - **Área de Control Terminal** (Terminal Control Area,TMA): son CTA establecidas en las confluencias de rutas en las inmediaciones de aeródromos principales (uno o más).
  - **Aerovías** (Airways, AWY): Son CTA dispuestas en forma de corredor con una extensión lateral dada por la precisión de las aeronaves para mantenerse en ellas y los medios de navegación disponibles, y con una extensión vertical que abarca todos los niveles de vuelo que requieran ATC.

- **Zona de Control** (Control Zone, CTR): Aquellas partes del espacio aéreo en las que se facilita el servicio de ATC desde la superficie terrestre hasta un límite superior especificado. Estos deben establecerse de manera que sus límites laterales abarquen al menos las partes del espacio aéreo que, no perteneciendo a una CTA, contengan las trayectorias de los vuelos IFR que llegan y salen de los aeródromos. Una CTR podría incluir dos o más aeródromos que estén suficientemente juntos. Los limites deberán cumplir:
  - **Límites laterales:** como mínimo a 5 nm del centro del aeródromo o aeródromos, en las direcciones en las que se puedan efectuar las aproximaciones, pero no extenderse demasiado, para no quitar espacio innecesario a los vuelos VFR que operan en las inmediaciones del aeródromo.
  - **Límites verticales:** Si se encuentra dentro de los límites laterales de una CTA, se extenderá hacia arriba hasta el límite inferior de la CTA (pudiendo ser incluso superior a este). En caso de que sea superior al de la CTA, o bien si excede los límites laterales de la CTA, entonces el límite superior de la CTR debería establecerse en un nivel fácilmente identificable por los pilotos. Si está a más de 900 m (3000 ft) sobre el nivel del mar, debe coincidir con un nivel de crucero VFR.
- **Aeródromo Controlado** (Aerodrome Traffic Zone, ATZ): Aquellas partes del espacio aéreo de un aeródromo en las que determina que ha de facilitarse ATC al tránsito de aeródromo. Se establece cuando la responsabilidad de tener un tráfico seguro y rápido alrededor de un aeródromo no puede dejarse a la discreción de los pilotos, debido a la operación de vuelos IFR o a tener un volumen de tráfico VFR muy alto. Este servicio debe considerar el tránsito de aeródromo y todo aquel tránsito que opera a una distancia razonable del mismo. Normalmente no debe exceder de las 25 nm.

1.2 Distancias mínimas de Separación

Para cumplir los objetivos del ATC de evitar colisiones entre aeronaves se establecen distancias mínimas de separación entre aeronaves de manera que se reduzca el riesgo de colisión manteniéndose separadas. Estas distancias se establecen según si son Verticales u Horizontales.

1.2.1 Distancias Verticales

[1][4]Esta separación se obtiene mediante la asignación de diferentes niveles de vuelo seleccionados de las tablas de niveles de crucero del Anexo 2 de OACI. En estas tablas, se puede ver que la separación vertical mínima es de:

- Por debajo de FL 290: 1000 ft
- Entre FL 290 (inclusive) y FL 410: 2000 ft salvo que exista un acuerdo regional que permita reducirlo a 1000 ft.
- Por encima de FL 410 (inclusive): 2000 ft salvo que exista un acuerdo regional que permita reducirlo a 1000 ft.

En la mayoría de las FIR, actualmente se tiene que la separación vertical mínima entre FL 290 y FL 410 es de 1000 ft, lo que se conoce como Reduced Vertical Separation Minimum (RVSM). Para poder hacer uso de esta separación reducida la aeronave debe estar capacitada para volar en RVSM (se necesita equipamiento especial, como dos altímetros independientes o un sistema de control de altitud automático, entre otros), en caso de que no lo estuviera, tan solo se le autorizaría a volar por debajo de FL 290 o por encima de FL 410.

Se considera que un nivel está ocupado por una aeronave si las variaciones de altitud de esta con respecto al nivel de vuelo son menores a 200 ft en caso de RVSM o 300 ft en otro caso.

1.2.2 Distancias Horizontales

Existen dos métodos de proporcionar distancias mínimas de separación horizontal, según procedimientos y según radar.

Distancias Horizontales según Procedimientos

Se proporciona cuando el control no dispone de vigilancia radar, de manera que la posición de las aeronaves se obtiene mediante informes que transmiten los pilotos por radio, siendo este método menos preciso que

el radar. Debido a esto dicha separación horizontal se obtiene diferenciando entre Separación Lateral y Separación Longitudinal.

Separación lateral: esta se aplica exigiendo a las aeronaves que vuelen por rutas diferentes o sobre puntos geográficos distintos que se puedan determinar por observación visual, mediante ayudas para la navegación o equipo de navegación de área (RNAV). La distancia mínima para la separación lateral se establece de manera que tenga en cuenta las inexactitudes de navegación y un margen de seguridad. Los medios para obtener la separación lateral se establecen en el Documento 4444 de OACI. Éstas suelen ser, para dos aeronaves operando en un mismo VOR, que ambas aeronaves se establezcan en radiales de 15º y una de ellas se encuentra a 15 nm o más de la radioayuda.

Separación longitudinal: esta se aplica manteniendo un intervalo entre las aeronaves que operan a lo largo de la misma derrota, derrotas convergentes u opuestas, determinado en tiempo o en distancia. Este intervalo se mantiene exigiendo a las aeronaves que salgan a horas determinadas, que permanezcan en circuito de espera hasta una hora determinada o mediante control de la velocidad. Según el Documento 4444 de OACI las separaciones mínimas suelen ser de 15 min, que pueden reducirse a 10 min si lo permiten las ayudas a la navegación, o incluso menos si ambas aeronaves siguen e mismo rumbo y la precedente tiene mayor velocidad.

Distancias Horizontales según Radar

Se proporciona cuando el control sí dispone de radar, lo que permite que, además de los procedimientos anteriores, el controlador disponga de herramientas como el encaminamiento directo o la guía vectorial. Con el encaminamiento directo el controlador autoriza al piloto a modificar su ruta prevista, volando directamente hacia una ayuda, un fijo o un punto significativo, lo cual puede facilitar la gestión del tráfico y/o ahorrar tiempo. Con la guía vectorial, el controlador indica al piloto el rumbo al que debe volar y, si dicho rumbo se desvía de la ruta asignada, se indicará el motivo por el que se da dicho vector, además de indicar su límite (por ejemplo, hasta x posición, para aproximación…)

De manera general, y según el Documento 4444 de OACI, la separación horizontal mínima basada en radar es de 5 nm, aunque este valor se puede ver reducido a 3 nm cuando así lo permita la capacidad del radar en determinado lugar, o incluso hasta 2,5 nm en casos específicos de aproximación final de la aeronave. Este valor también puede verse aumentado a 15 nm en caso de no disponer de multiradar.

En los casos que vamos a contemplar, que para simplificar se va a incluir solo control de área con disponibilidad de multiradar en FIR con RVSM, las separaciones mínimas horizontal y vertical equivaldrán a asociar a cada aeronave un volumen de protección en forma de cilindro vertical centrado en sus masas, de radio 5 nm y semialtura 1000 ft (2000 ft en caso de estar en FL 410 o superior), en el cual no deben penetrar otras aeronaves.

2 Estado del Arte, Simuladores de ATC actuales

A la hora de realizar un proyecto, siempre es importante realizar un estudio del estado del arte, ver qué productos existen ya que se asemejen o cumplan el propósito y los objetivos del proyecto que se quiere realizar y buscar en ellos inspiración, así como soluciones a problemas que se puedan encontrar en el desarrollo productivo del proyecto. En concreto para este proyecto se han estudiado tres de los simuladores ATC más usados, entre los que se encuentran: Rams Plus, BlueSky ATC (Simulador de la Universidad de Delft) y AURORA (usado en IVAO).

2.1 Simulador RAMS Plus

[6]El primer simulador que se ha investigado es el conocido como RAMS Plus (Reorganised ATC Mathematical Simulator), desarrollado por la empresa ISA Software, y que está diseñado para la realización de simulaciones para análisis de ATM en escenarios de todo el mundo. Por esta razón, este simulador no presenta herramientas de ATC como tal, sin embargo, sí que dispone de herramientas de modelado de escenarios.

Figura 2.1 Logo de ISA Software.

[]En concreto, cuenta con dos "dominios funcionales" (así es como se refieren a ellos): el RAMS Airside y el RAMS Groundside. Ambos dominios funcionan al mismo tiempo para mostrar una vista de puerta a puerta del sistema de tráfico aéreo modelado.

Dominio Airside

El dominio Airside es un simulador de pista a pista, cuya funcionalidad principal incluye:

- Cálculo de trayectoria de vuelo en 4D (incluye el tiempo)
- Sectorización 3D
- Detección de conflictos de espacio en 4D
- Múltiples estrategias de Separación (p.e. separación por controlador, por equipos, etc…)
- Resolución de Conflictos por Inteligencia Artificial
- Resolución de Maniobras en 4D (para retrasos en ruta)
- TMA (secuenciación de llegadas y salidas, SIDS/STARS, esperas, vectorización en aproximación, etc)
- Asignación de Tareas Dinámico

- Conflictos de planificación y uso de pistas
- Tiempo (efectos del viento en las actuaciones de aeronaves en vuelo)
- Entornos de Editor y Display en MS Windows estándar
- Gráficos animados de simulación y modo repetición

Los principales objetivos del simulador en el dominio Airside son:

- Implementación de los beneficios de NextGen y SESAR
- Proponer sectorizaciones alternativas
- Medir la carga de trabajo de los controladores
- Medir la complejidad y densidad de conflictos y del espacio aéreo
- Medir la seguridad del espacio aéreo en relación con las violaciones de separaciones
- Impactos de los conceptos del free-routing y del RVSM
- Impacto del ATM en el consumo de combustible
- Visiones ilimitadas de conceptos de ATM

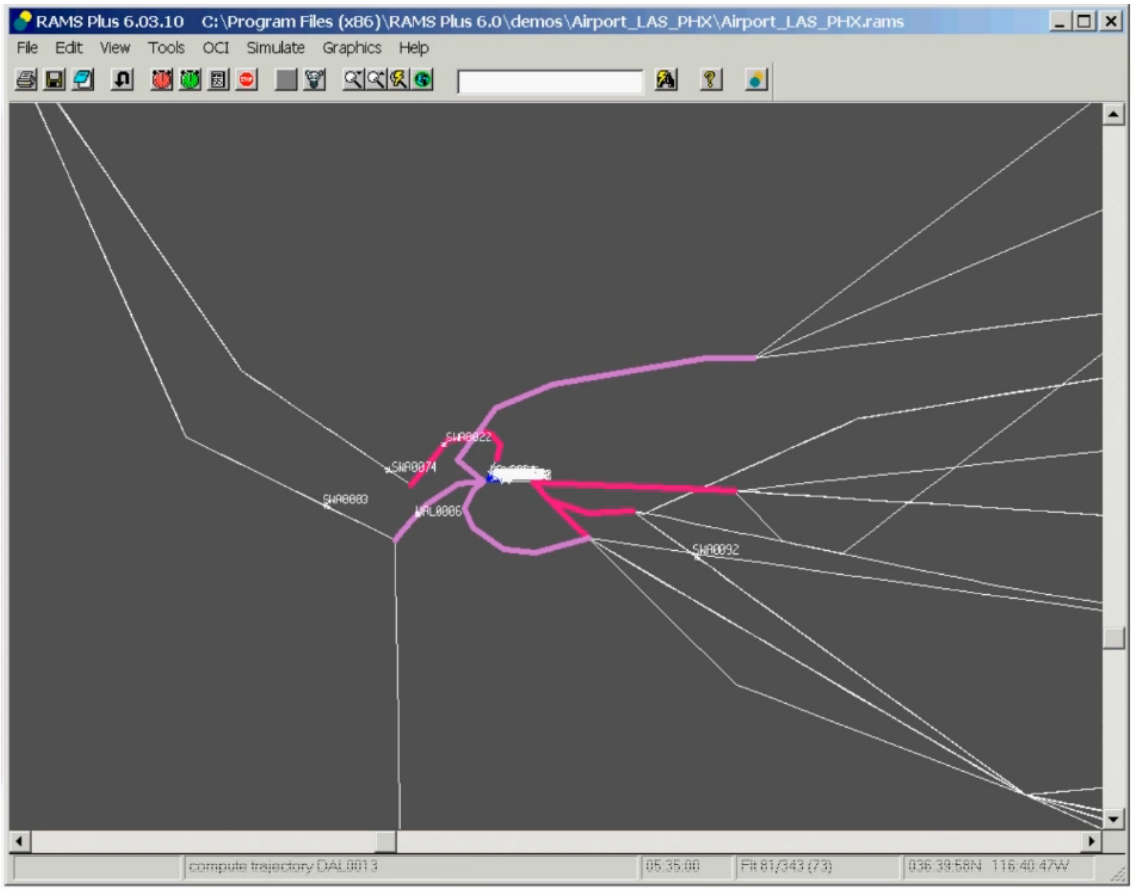

Figura 2.2 RamsPlus Dominio Airside.

Dominio Groundside

El dominio Groundside simula operaciones aeroportuarias de puertas a SIDS y STARS, cuyas funcionalidades incluyen:

- Varios diseños de aeropuertos en un mismo escenario
- Colocación de puertas por aerolínea y modelo de aeronave

- Rutas de taxi más cortas
- Velocidad y separación de las conexiones taxi
- Conexiones taxi por llegada/salida, aerolínea y modelo de aeronave
- Resolución de Conflictos mediante Inteligencia Artificial
- TMA (secuenciación de llegadas y salidas, SIDS/STARS, esperas, vectorización en aproximación, etc…)
- Resolución de Maniobras en 4D (para aproximaciones en TMA)
- Conflictos de planificación y uso de pistas
- Ratios de aceleración/deceleración en pistas por modelo de aeronave
- Salidas de pista de alta velocidad (por modelo/tipo de aeronave y aerolínea)
- Trazabilidad de pistas
- Pistas bloqueando otras pistas cuando se usan
- Entornos de Editor y Display en MS Windows estándar
- Gráficos de simulación Animados
- Informes de retrasos según criterios ilimitados (p.e. por aerolíneas, puertas, pistas, colas de salidas, etc…)

Los principales objetivos del simulador en el dominio Groundside son:

- Proponer configuraciones de pistas alternativas, incluyendo nuevas pistas o pistas cerradas
- Medir operaciones y demandas de pistas
- Medir retrasos en tierra, por ejemplo por tiempos de taxi, aerolíneas, salidas/llegadas, retrasos medios, etc…
- Proponer nuevas configuraciones de vías de taxi
- Usar ambos dominios en conjunto: puerta a puerta
- Compartición de datos entre el Airside y el Groundside

El objetivo principal de usar ambos dominios en conjunto es medir los impactos de las operaciones y los retrasos en tierra en el sistema de ATM completo, dando una vista de puerta a puerta del escenario estudiado.
Datos requeridos para usar RAMS Plus

RAMS Plus funciona enteramente con datos, e incluye por defecto datos sobre modelos y actuación de aeronaves, localización de aeropuertos, normativas comunes, fronteras de países, etc. Los formatos de estos datos son fáciles de crear, donde los datos de RAMS Plus están en formato de texto ASCII, y todas las referencias a localizaciones están en latitud/longitud en cualquier formato reconocido. ISA Software también ofrece un servicio de análisis de ATM y preparación de datos, útil para quienes tengan limitaciones de tiempo y recursos.

Los datos necesarios para estudiar un escenario de espacio aéreo básico son:

- Nombres y lugares geográficos de las ayudas a la navegación
- Horarios de tráfico
- Rutas de tránsito (Ayudas a la navegación y/o Aerovías (compuestas de Ayudas a la navegación))
- Sectorización (básicamente compuesta de la latitud/longitud geográficas de los vértices)

Además de estos, si se quiere estudiar un escenario básico de una TMA, se necesitan los datos de:

- SIDS y STARS
- Localización de las pistas del aeropuerto
- Circuitos de espera (opcional)

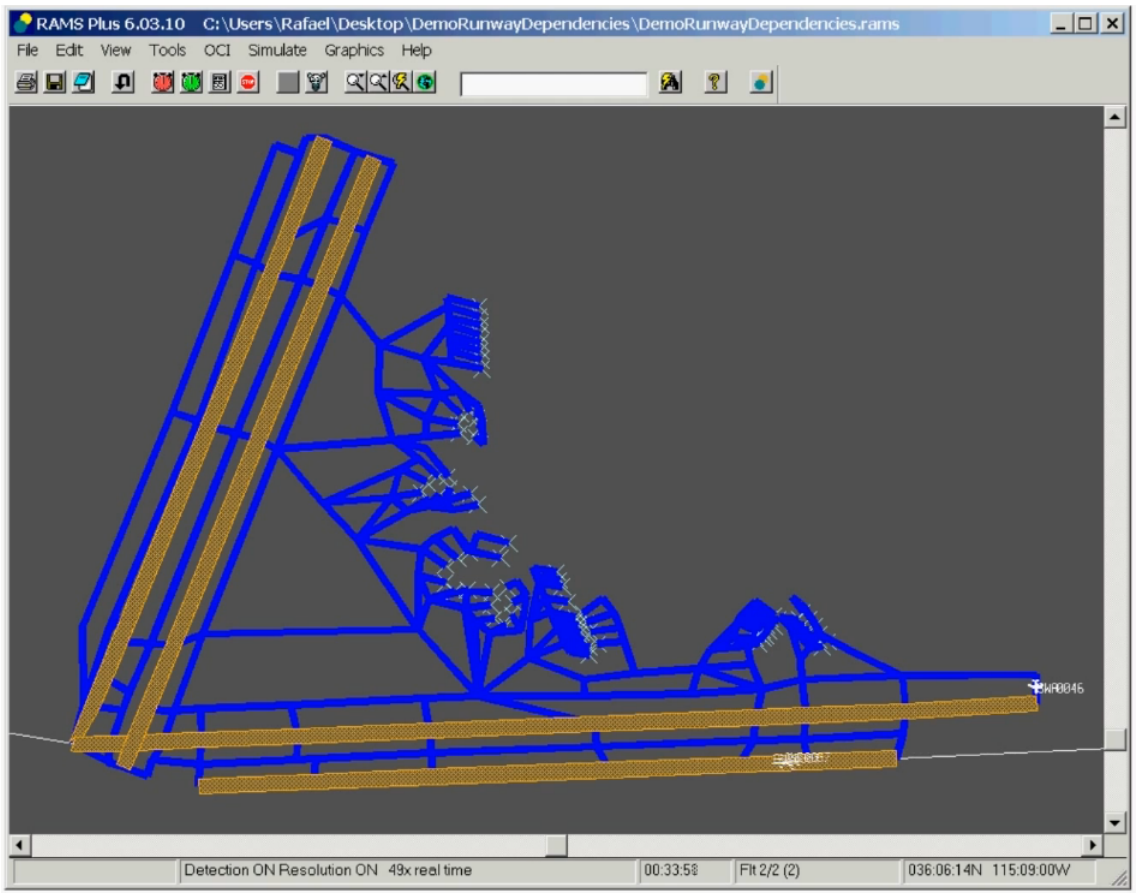

Figura 2.3 RamsPlus Dominio Groundside.

Además de todos los datos anteriores, si se quiere estudiar las Operaciones Aeroportuarias básicas de un escenario, se requieren los datos de:

- Configuración de pistas
- Localización de puertas y su uso (por aerolínea, tipo de aeronave, etc)
- Rutas de taxi y su uso (por salidas y llegadas, tipo de aeronave, etc)

Debido a que este es un simulador profesional de pago no se dispone de más información sobre las herramientas que incluye ni el funcionamiento específico de estas, aun así se puede basar el formato de los datos usados en el simulador propio en el último apartado descrito. Por lo demás se centrará en replicar o implementar herramientas similares a las descritas en el Dominio Airside, dado que de momento, para simplificar el proyecto por las limitaciones de tiempo y recursos, solo se va a contemplar el control de área y en casos concretos, dejando los demás casos para posibles actualizaciones futuras del programa. Aun con esta simplificación, las herramientas de Cálculo de Trayectorias de vuelo en 4D, Sectorización 3D, así como la Detección de Conflictos en 4D serán implementadas, de la mejor manera posible, pues serán necesarias para simular un mínimo de las acciones requeridas por un Controlador.

2.2 Simulador BlueSky ATC

[7][8]El segundo simulador que se ha investigado es el conocido como BlueSky Open Air Traffic Simulator, un simulador de código abierto desarrollado en su origen por el Profesor Jacco Hoekstra de la Universidad Tecnológica de Delft (Países Bajos). Este se creó como una herramienta para realizar investigación sobre Gestión del Tráfico Aéreo (ATM) y Flujos de Tráfico Aéreo (ATF).

El objeto de BlueSky es proveer a todo el que lo desee visualizar, analizar o simular tráfico aéreo, con una herramienta para hacerlo sin ningún tipo de restricciones, licencias o limitaciones. Puede ser copiado,

Figura 2.4 Logo Universidad Tecnológica de Delft y Foto del Prof. Jacco Hoekstra.

modificado, citado, etc. Sin ninguna limitación. Para ello Bluesky está programado y desarrollado en el lenguaje multiplataforma Python, la última versión está escrita en Python 3 (con los módulos de Python, numpy con o pygame o Qt+OpenGL para la visualización).

Las características principales de BlueSky son:

- Es ampliable y actualizable mediante plugins auto-contenidos
- Contiene datos de código abierto sobre Sistemas de Ayuda a la Navegación, datos de actuaciones de aeronaves y geografía
- Datos a nivel global de la provisión de Ayudas a la Navegación y de Aeropuertos
- Contiene simulaciones de actuaciones de aeronaves, de sistemas de gestión de vuelo, autopilotos, detección y resolución de conflictos y sistemas para mantener la separación entre aeronaves
- Compatible con datos BADA (Base of Aircraft Data) 3.x
- Compatible con NLR Traffic Manager (TMX), usado por NLR (Laboratorio Aeroespacial Nacional de Países Bajos) y la NASA
- El tráfico se controla mediante entradas de usuario en una ventana de consola o ejecutando archivos de escenario (.scn) conteniendo los mismos comandos, añadiéndole el tiempo de lanzamiento de cada comando antes de cada comando (en formato “HH:MM:SS.hh>”)
- Los clicks del ratón en la ventana de tráfico se usan en la consola para las entradas de Latitud/Longitud/rumbo y posición

El BADA es una base de datos de actuaciones de aeronaves diseñada para su uso en simulaciones de trayectorias de aeronaves y en algoritmos de predicción dentro del ámbito del ATM. [10]

El NLR Traffic Manager (TMX) es una herramienta de simulación del tráfico aéreo desarrollada por el NLR (Laboratorio Aeroespacial Nacional de Países Bajos) originalmente para el estudio del tráfico aéreo en condiciones de Free Flight o Vuelo Libre, condiciones mediante las cuales las aeronaves deciden su propio camino, sin intervención de ATC, y manteniendo las mínimas de separación con las demás aeronaves mediante herramientas embarcadas. Esta herramienta se creó también como plataforma de simulación escalable para alcanzar las necesidades del tráfico y lo suficientemente flexible para modelar interacciones entre aeronaves en las condiciones de Vuelo Libre. Durante los ocho años siguientes se ha seguido mejorando y actualizando la herramienta TMX por investigadores y desarrolladores profesionales, hasta convertirlo en un simulador de tráfico aéreo propio, generador, editor y reproductor de escenarios, así como en una estación de control de experimentos, herramienta de recopilación de datos y como entorno para un rápido desarrollo de prototipos. Al tener un diseño modular, una gran simplicidad de uso y extensibilidad, TMX se convirtió en un activo muy importante en proyectos de investigación sobre ATM. Esta herramienta se podría haber incluido entre las investigadas para este proyecto, sin embargo, debido a que su diseño está orientado más a la investigación

en ATM, en concreto para condiciones de Vuelo Libre, no se ha tenido en cuenta para el Estado del Arte de este proyecto, pero sí se ha visto necesario mencionarla como base sobre la que se apoya BlueSky ATC.

A parte de toda esta información encontrada en las diversas páginas principales de BlueSky ATC, se ha intentado instalar esta herramienta de simulación para comprobar su funcionamiento y diseño estético, aunque debido a problemas surgidos al estar trabajando con un lenguaje con el que se está poco familiarizado, se ha decidido cesar los intentos de hacer funcionar esta herramienta en pos de dedicar ese tiempo al desarrollo de la herramienta propia de este proyecto. Aun teniendo estas dificultades, se ha encontrado una wiki en el GitHub de BlueSky ATC [9] con suficiente información tal que hace innecesaria la instalación del programa para los objetivos para con ella que se tienen en este proyecto.

[11]Dentro de lo que ofrece esta wiki, se ha centrado principalmente en la sección de Tutoriales, ya que no se disponía de la herramienta en sí, y entre estos tutoriales, se han investigado y tomado como referencia las secciones: “The BlueSky Interface”, “Aircraft Settings”, “Navigation Commands”, “Sim Commands” y “Editing Flight Plans” así como las secciones donde se describe el funcionamiento y la sintaxis de los comandos “ADDWPT”, “CRE” y “MOVE”.

The BlueSky Interface

Al abrir BlueSky, se abre la ventana con la interfaz del simulador vista en la imagen de la **Figura 2.5**, donde se ve la interfaz con varios recuadros de color marcando las distintas secciones de la interfaz, que se describen a continuación.

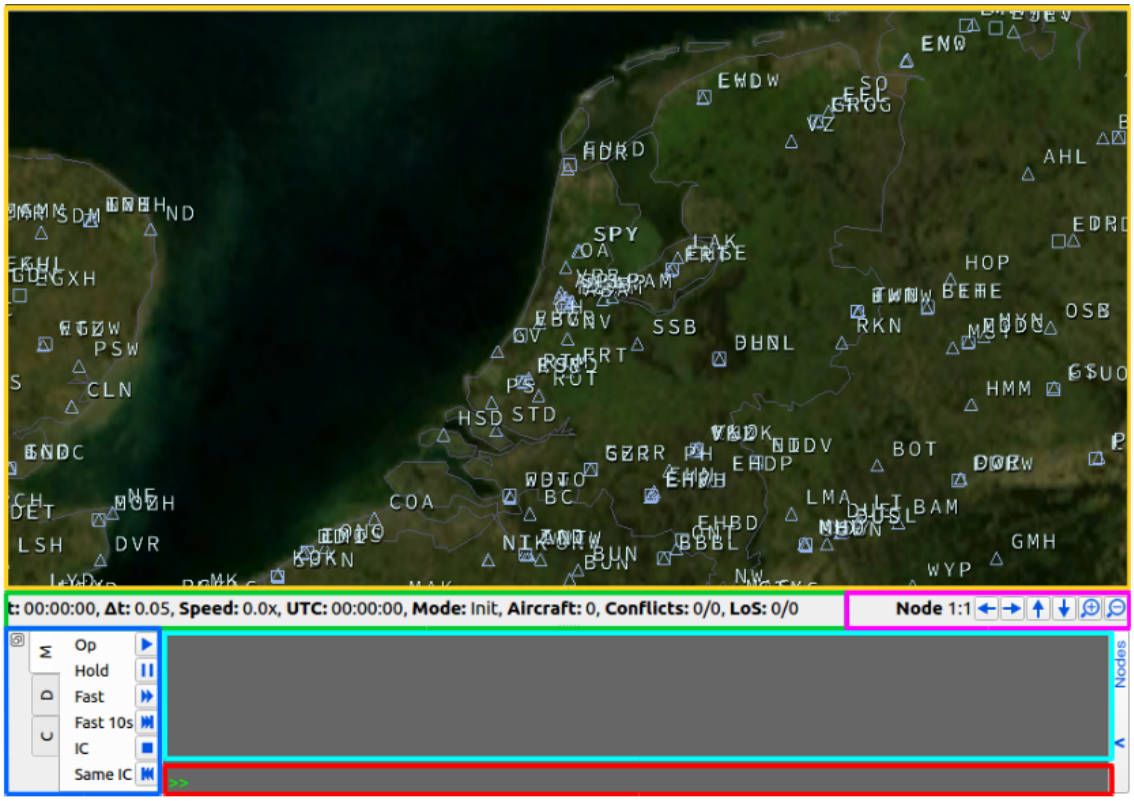

Figura 2.5 Interfaz de BlueSky ATC.

Vista principal del Mapa (amarillo): Esta sección contiene la vista aérea principal de la situación del tráfico. En esta pantalla se muestra toda información relevante sobre aeronaves, puntos de paso de navegación y otros elementos geométricos de interés.

Línea de Comandos (rojo): Aquí es donde se introducen los comandos de BlueSky.

Salida de Texto (cian): Esta sección contiene las salidas de texto de los comandos introducidos, similar a la ventana de comandos de una consola.

Controles de Simulación (azul): Los botones de esta sección dan acceso rápido a varios comandos. Estos se dividen en tres pestañas:

- M contiene los botones para los comandos de control del tiempo y selección de escenario

- D contiene los botones para los comandos de control del display
- C contiene espacio para comandos personalizados definidos por el usuario

Estado de Simulación (verde): Aquí se muestra el estado de 8 parámetros generales de simulación, entre los que se encuentran, por ejemplo, el tiempo de simulación actual (t), el paso de simulación (Δt), la velocidad de la simulación (Speed), entre otros.

Controles de Cámara (magenta): En esta sección se tienen 6 botones para modificar la posición de la cámara, concretamente en orden de izquierda a derecha: mover cámara hacia el oeste, este, norte, sur respectivamente, acercar y alejar la cámara.

Aircraft Settings

En este tutorial se explican los distintos comandos relacionados con la creación eliminación y modificación de aeronaves dentro del espacio de simulación. Para ello, primero se ve cómo crear una aeronave, viendo la ayuda del programa del comando CRE:

Código 2.1 Ayuda del Comando CRE.

```
HELP CRE
CRE acid, type, lat, lon, hdg, alt, spd
Create an aircraft
```

De esto obtenemos que, para crear una aeronave en BlueSky se necesita introducir 7 argumentos después del comando CRE. Estos son el Identificador de la Aeronave (Aircraft Indicator, acid), el tipo de aeronave, las coordenadas (lat, long, alt) y el vector de velocidad (hdg, spd).

En el tutorial, a continuación, se crea una aeronave sobrevolando el Mar del Norte en dirección sur, introduciendo en la entrada de comandos el código siguiente (**Código 2.2**):

Código 2.2 Creación de una Aeronave.

```
CRE KL001 B737 52 3 180 FL100 300
```

Con este comando se crea una aeronave con los siguientes parámetros: Identificador KL001, Tipo Boeing 737, Latitud 52 grados, Longitud 3 grados, Rumbo 180 grados (hacia el Sur), Nivel de Vuelo 100 y velocidad 300 nudos. El resultado de este comando se puede ver en la **Figura 2.6**.

Como se puede ver se crea un icono de una flecha verde que representa a la aeronave con su posición y rumbo, además se muestran los parámetros de identificador, altitud y velocidad al lado del icono. Al crear una aeronave en BlueSky se inicia la simulación, como se ve en el parámetro t del Estado de Simulación, también se observa que el contador de aeronaves aumenta en 1.

A continuación, se crean varias aeronaves con distintos métodos y sintaxis del comando CRE. En la primera, se tiene que las coordenadas no se introducen como Latitud/Longitud, sino que se introduce la cadena de caracteres correspondiente al Punto de Paso sobre el que se quiere situar la aeronave (en este caso EHAM, el aeropuerto de Ámsterdam). En la segunda se tiene que se usa la posición de la primera aeronave creada (KL001), el comando solo toma los datos de Latitud y Longitud de esta, se sigue teniendo que indicar la altitud. En la tercera aeronave se usa un método distinto para introducir la posición y rumbo, que consiste en utilizar una pista de aeropuerto como referencia (en este caso la RW06 del aeropuerto EHAM). Este último método requiere en su sintaxis que se use lo que se llama wildcard (*) como rumbo. También cabe destacar que en el argumento de altitud, para introducirlo como Niveles de Vuelo hay que escribir FL seguido del parámetro, si no se hace, el valor se toma como la altitud en pies (ft), como es el caso de KL004, que está a 100 ft, probablemente debido a que acaba de despegar de la pista RW06 del aeropuerto o pretende aterrizar en dicha pista. En la **Figura 2.7** se ve el resultado de los comandos (**Código 2.3**) en la pantalla.

Código 2.3 Creación de múltiples Aeronaves.

```
CRE KL002 B737 EHAM 0 FL150 300
CRE KL003 B737 KL001 90 FL150 300
CRE KL004, B737, EHAM, RW06,*, 100, 200
```

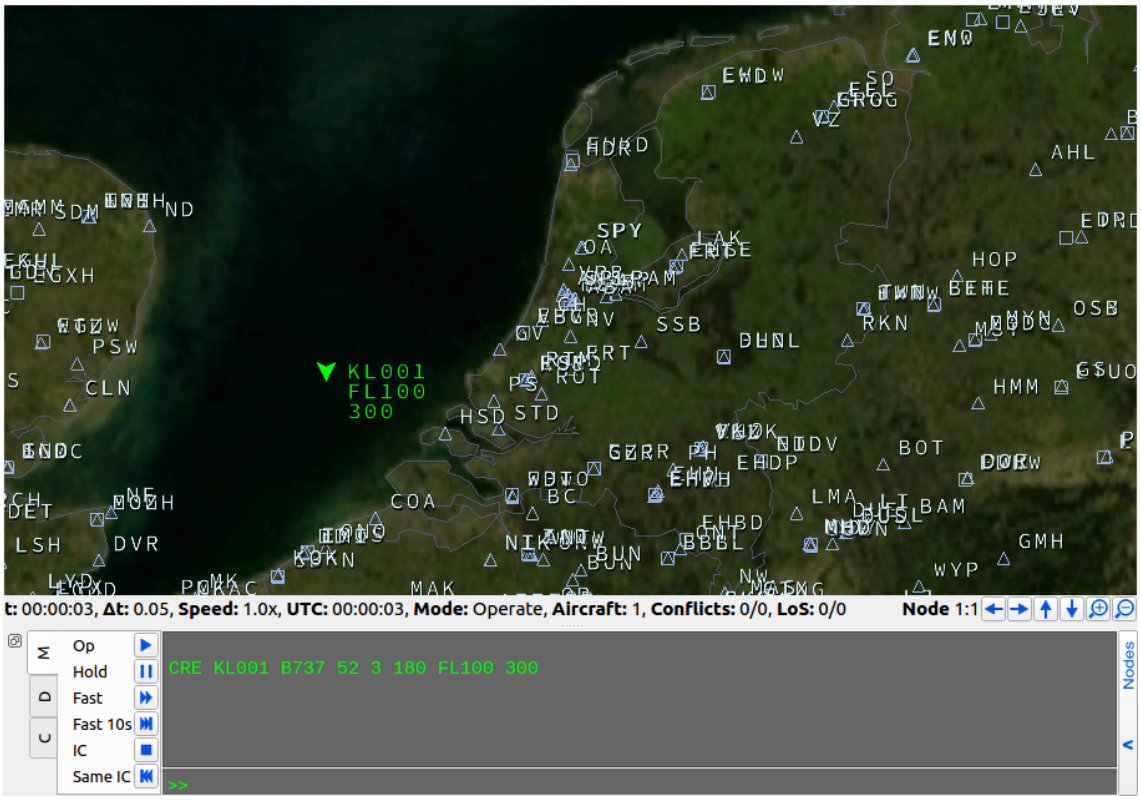

Figura 2.6 Creación de una Aeronave.

Figura 2.7 Creación de múltiples Aeronaves.

Por último, existe un método para crear una aeronave de un modo más interactivo, haciendo click en la pantalla. Para este método se introduce el **Código 2.4** sin darle a enter, tras el cual BlueSky espera que se introduzcan las coordenadas. Haciendo click en la Pantalla del Mapa se introducen automáticamente las coordenadas del ratón dentro del mapa en la línea de comando. Para terminar habría que introducir los demás parámetros necesarios (véase altitud, rumbo y velocidad).

Código 2.4 Creación de Aeronave mediante Entradas de Ratón.

```
CRE KL005 B737
```

Se pueden eliminar aeronaves con el comando DEL seguido del Identificador de la Aeronave que se desea eliminar. En el tutorial se eliminan todas las aeronaves excepto la KL001 con el **Código 2.5**:

Código 2.5 Eliminar Aeronaves.

```
DEL KL002
DEL KL003
DEL KL004
DEL KL005
```

Existe un comando que nos sirve para comprobar los parámetros de una aeronave a mitad de simulación. Este comando es el comando POS, que necesita como argumento el Identificador de la Aeronave en cuestión como en el **Código 2.6** , el resultado se puede ver en la **Figura 2.8**.

Código 2.6 Comando POS.

```
POS KL001
```

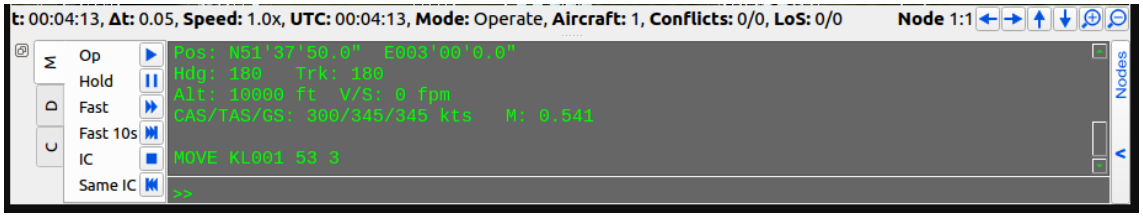

Figura 2.8 Comandos POS y MOVE.

Para modificar los parámetros de la aeronave, se tiene el comando MOVE, que sigue la siguiente sintaxis:

Código 2.7 Sintaxis MOVE.

```
MOVE acid, lat, lon, [alt, hdg, spd, vspd]
```

Donde vspd es la velocidad vertical de la aeronave. Los argumentos entre corchetes no son necesarios para el funcionamiento del comando, y si no se introducen se mantienen los parámetros guardados. Al lanzar el comando MOVE, se traslada automáticamente la aeronave a la posición indicada por los nuevos parámetros introducidos, como se puede ver en la **Figura 2.8** tras aplicar el **Código 2.8**.

Código 2.8 Comando MOVE.

```
MOVE KL001 53 3
```

La sintaxis del comando MOVE se asemeja bastante a la del comando CRE, hasta el punto de que se puede indicar la posición introduciendo un punto conocido (ya sea un Punto de Paso o una Aeronave ya creada), como se ve en el **Código 2.9**, en el que se mueve la aeronave KL001 hasta el FL300. Podemos ver el resultado en la **Figura 2.9**.

Código 2.9 MOVE Altitud.

```
MOVE KL001 KL001 FL300
```

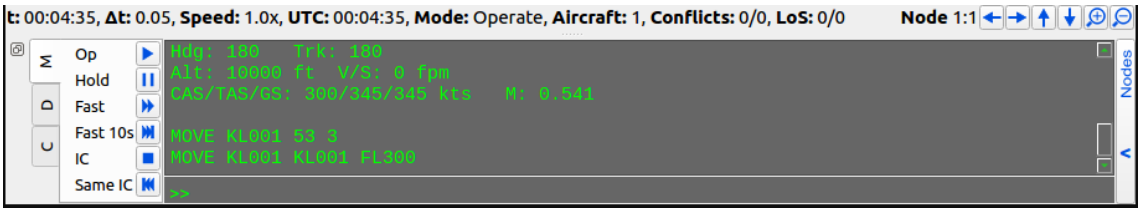

Figura 2.9 Comando MOVE Altitud.

Navigation Commands

En este tutorial se introducen nuevos comandos que sirven para causar actuaciones de las aeronaves, además de otros comandos que afectan a la simulación de la navegación. Entre estos, existe uno que permite ver la trayectoria recorrida por las aeronaves, este es el comando TRAIL, cuya sintaxis requiere determinar si está activado (ON) o está desactivado (OFF). Este comando lo que hace es dibujar una línea cian que sigue a la aeronave y representa la trayectoria recorrida por esta.

Para este tutorial se crea primero una aeronave sobrevolando el aeropuerto de Sciphol rumbo norte (**Código 2.10**). A continuación se usa el comando HDG para cambiar el rumbo de la aeronave. Este comando solo necesita como argumento el rumbo en grados (siendo 0 el Norte e incrementando en sentido horario). En el **Código 2.10** se indica un rumbo 270, que coincide con el Oeste, de modo que en la simulación se vería un giro a la izquierda por parte de la aeronave.

Código 2.10 Código Navigation Comands - Creación de la Aeronave.

```
CRE KL001 B737 EHAM 0 FL150 200
HDG KL001 270
```

También se puede hacer que la aeronave navegue hacia un punto específico. En el **Código 2.11** se utiliza el comando ADDWPT, que sirve para añadir un punto de paso a la ruta seguida por una aeronave, en concreto este comando añade el aeropuerto EHRD (Rotterdam). El comando ADDWPT tiene varias opciones de sintaxis pero en este tutorial se deja la forma más simple, que consiste en determinar primero la aeronave en cuestión y luego el nombre del punto de paso que se va a añadir a su plan de vuelo. En el **Código 2.11** también se añaden otros dos puntos de paso, a la cadena de puntos de paso del avión, en concreto los aeropuertos EBBR (Bruselas) y LFPO (París). Esta cadena hará que la aeronave se dirija primero a Rotterdam, a continuación se dirigirá hacia Bruselas y por último a París.

Código 2.11 Navigation Commands - ADDWPT Ruta.

```
ADDWPT KL001 EHRD
ADDWPT KL001 EBBR
ADDWPT KL001 LFPO
```

Sim Commands

En este tutorial se ven los distintos comandos que se pueden usar para controlar la simulación. En concreto los comandos que te permiten pausar, iniciar, reiniciar, adelantar y parar las simulaciones. Además, también se verá como configurar el reloj de la simulación, así como su paso de simulación (incremento de tiempo entre tomas/muestras de datos de la simulación). De entre los parámetros de simulación presentes en el panel de Estado de la Simulación (recuadro verde en la **Figura 2.10**), en este tutorial se ven solo los siguientes:

- **t** – Tiempo de la simulación
- **Δt** – Paso de simulación

- **Speed** – Factor multiplicador para simulaciones a cámara rápida
- **UTC** – Reloj de la simulación
- **Mode** – Modo de simulación

Figura 2.10 Controles de Simulación.

El tiempo de simulación (t), así como el reloj de simulación (UTC) están ambos en 00:00:00, lo cual es lógico teniendo en cuenta que aún no se ha creado ni una aeronave todavía y, por ende, no hay nada que simular. También se ve que el Modo de Simulación (Mode) es Init, es decir, que la simulación está en su estado inicial. Para iniciar la simulación, se crea una aeronave con el **Código 2.12**.

Código 2.12 Sim Comands - Creación de Aeronave e Inicio de la Simulación.

```
CRE KL001 B737 52 3 180 FL100 300
```

En la **Figura 2.11**, se puede ver la aeronave creada. También se observa que el tiempo de simulación (t) y el reloj de simulación (UTC) empiezan a correr, lo que indica que la simulación está en proceso. Adicionalmente, se puede ver que el modo de simulación (Mode) pasa a ser Operate.

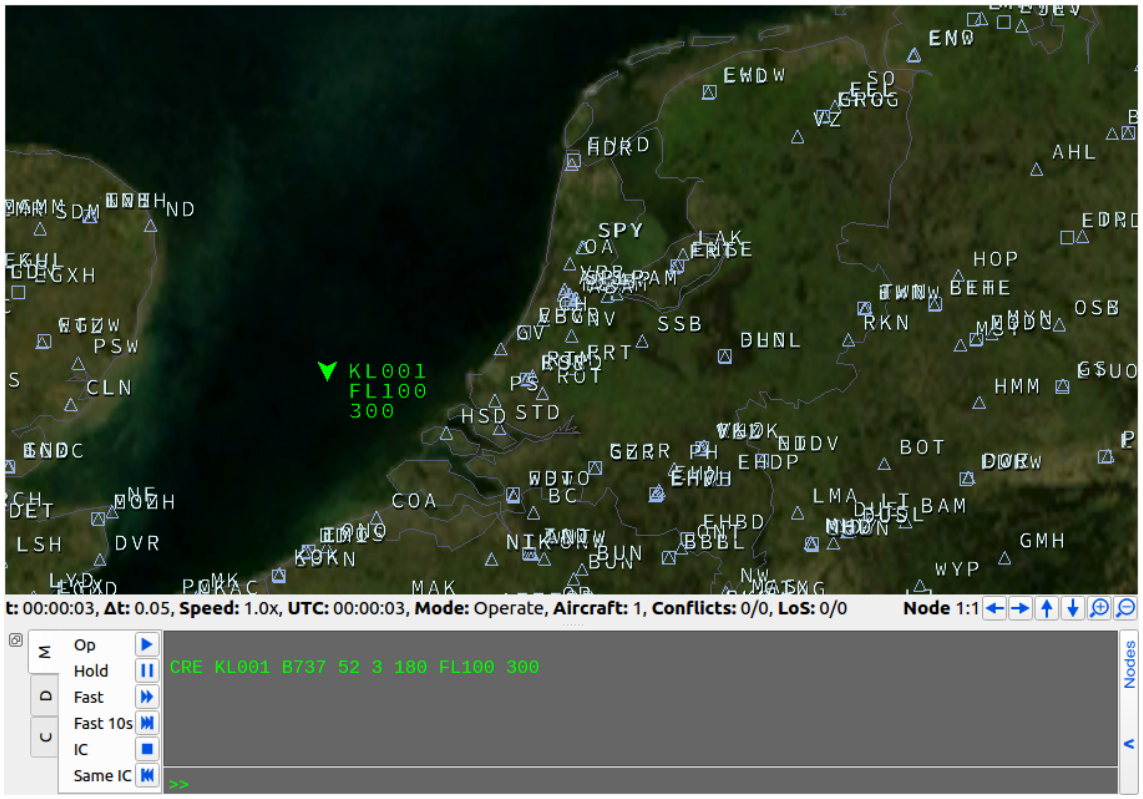

Figura 2.11 Sim Control Después de usar el Comando CRE.

A continuación se exploran los comandos que se pueden usar para controlar la simulación:

Pausar la Simulación: En el tutorial recomiendan dejar que proceda la simulación durante unos 10 minutos e introducir el comando HOLD en la línea de comandos. Esto hace que tanto el tiempo de la simulación (t) como el reloj de la simulación (UTC) se paren, y el modo de simulación (Mode) pasa a ser Hold, lo que significa que se ha pausado la simulación. También se puede pausar pulsando el botón de Hold en el Panel de Controles de Simulación. En la **Figura 2.12** vemos el resultado de introducir este comando.

Figura 2.12 Simulación Pausada.

Iniciar la Simulación: Para Iniciar, o en este caso más bien Reanudar, la Simulación para que continúe no hace falta más que introducir el comando OP en la línea de comandos o pulsar el botón Op en el Panel de Controles de la Simulación, lo que hace que tanto el tiempo de simulación (t) como el reloj de simulación (UTC) se activen y que el modo de simulación (Mode) pase a ser Operate.

Adelantar la Simulación: Debido a que las simulaciones que se establecen pueden, y suelen, ser largas (en torno a 2 horas), interesa poder aumentar la velocidad de simulación, es decir, adelantar la simulación. Para adelantar (o hacer Fast-Forward) la simulación, de manera similar a los comandos anteriores basta con introducir el comando FF en la línea de comandos o pulsar el botón Fast en el Panel de Controles de la Simulación.

Hay que tener en cuenta que solo se puede adelantar una simulación en proceso, de manera que si se quisiera adelantar una simulación en pausa, habría que reanudar la simulación antes de adelantarla. Se puede verificar si se está adelantando la simulación mirando el parámetro Speed en el Panel de Estado de Simulación, o mirando la velocidad a la que la aeronave se mueve fuera de la zona visible de la ventana de mapa, como en la **Figura 2.13**.

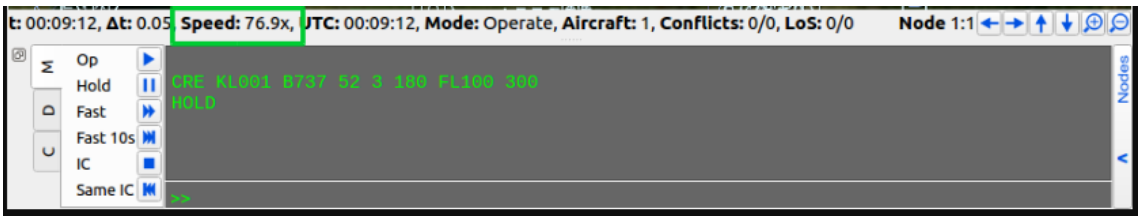

Figura 2.13 Simulación Adelantada.

El valor del parámetro Speed indica el factor de multiplicación para la simulación a cámara rápida. En el ejemplo de la **Figura 2.13** se tiene como 76.9, este valor suele ser grande, aunque puede variar ligeramente. Cuando se usa el comando FF sin argumentos se adelantará la simulación de manera perpetua hasta que se interrumpa/pause por otro comando o se cierre la ventana de BlueSky. Por ello, se puede especificar la cantidad de segundos en los que se adelanta la simulación añadiéndola como argumento al comando FF

al introducirlo en la línea de comandos, como en el **Código 2.13**, que adelanta la simulación durante 10 segundos. Alternativamente, se puede usar el botón Fast 10 del Panel de Controles de la Simulación para este propósito.

Código 2.13 Comando FF.

```
FF 10
```

Reiniciar Simulación: El comando RESET sirve para despejar la vista de mapa y la simulación, sin tener que cerrar BlueSky, para ello se introduce el comando RESET en la línea de comandos.

Configurar el Reloj de Simulación: También es posible modificar el reloj de la simulación (UTC) en el Panel de Estado de Simulación. Inicialmente, el reloj de simulación empieza en 00:00:00, igual que el tiempo de simulación. Para cambiarlo, hay que introducir el comando TIME, seguido de la opción elegida de tiempo en la línea de comandos. Existen 5 opciones. Primera, introducir TIME, seguido de los valores de horas, minutos y segundos en el formato HH:MM:SS para configurar un momento en el tiempo concreto. Segunda, introducir TIME seguido del valor de las horas para configurar una hora específica. Tercera, mostrada en el **Código 2.14**, sirve para configurarlo con la hora del ordenador en ese momento. Cuarta, mostrada en el **Código 2.14**, sirve para configurarlo con la hora UTC, y se puede ver el resultado en la **Figura 2.14**. Quinta, mostrada en el **Código 2.14**, sirve para igualar el reloj de simulación con el tiempo de simulación.

Código 2.14 Comando TIME.

```
TIME REAL
TIME UTC
TIME RUN
```

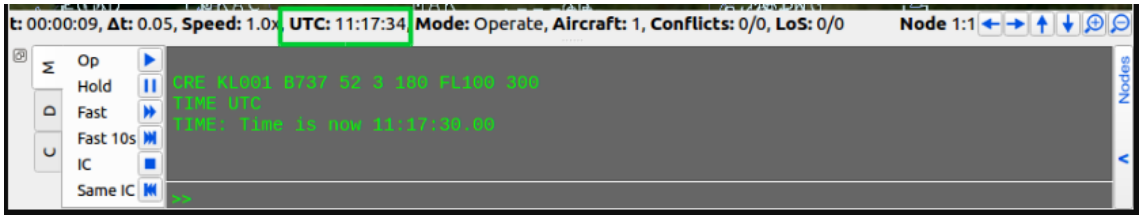

Figura 2.14 TIME UTC.

Cerrar la Simulación: Para cerrar la simulación se puede introducir el comando QUIT en la línea de comandos o simplemente cerrando la ventana de BlueSky. Cabe destacar que usar el comando QUIT es más recomendable pues algunos comandos introducidos pueden no detenerse hasta que se finaliza la ejecución de BlueSky, el comando QUIT hace que estos comandos se detengan antes de cerrar la aplicación.

Editing Flight Plans

Este tutorial muestra cómo crear una aeronave y modificar su ruta. Se empieza creando una aeronave con el **Código 2.15**. Para a continuación hacer que ascienda a FL100 con una velocidad de 250 kt, y luego pausar la simulación. Esto se hace con el **Código 2.15** también.

Código 2.15 Editing Flight Plans - Creación de la Aeronave.

```
CRE KL001, B737, 52, 4, 180, 2000, 220
KL001 ALT FL100
KL001 SPD 250
HOLD
```

En esta situación se está preparado para introducir una ruta en el sistema de gestión de vuelo (FMS) de la aeronave. Primero introducimos el destino usando el identificador OACI del aeropuerto, en este caso Francia, Paris de Gaulle (**Código 2.16**). Si se iniciara la simulación con el comando OP, la aeronave volaría hacia el destino usando la ruta ortodrómica (es decir, la más corta en distancia). Opcionalmente, también se puede

introducir el origen, en este caso el aeropuerto de Ámsterdam, con el **Código 2.16**, donde vemos tres distintas maneras de introducirlo, dos de las cuales especifican la pista de despegue.

Código 2.16 Editing Flight Plans - Añadir Origen y Destino.

```
KL001 DEST LFPG
KL001 ORIG EHAM
KL001 ORIG EHAM RW06
KL001 ORIG EHAM/RW06
```

Ahora se puede hacer que la aeronave vuele directamente a LFPG en línea recta, pero interesa darle una ruta FMS para volar. Al darle doble click a la aeronave (o introduciendo su identificador en la línea de comandos o el comando POS con su identificador) se muestra la ruta actual en la vista de mapa. Ahora también se pueden ir añadiendo puntos de paso a la ruta con el comando ADDWPT, cuya sintaxis es:

Código 2.17 Sintaxis ADDWPT.

```
ADDWPT acid, wpname/position, [alt], [spd], [afterwp]
```

Como se ha podido ver en códigos anteriores, el identificador de la aeronave puede ir antes o después del comando indistintamente, siempre y cuando la aeronave exista (véase, no con el comando CRE). Al igual que con otras sintaxis de comandos, los parámetros en corchetes son opcionales, mientras que los que no están en corchetes es necesario introducirlos para que funcione el comando. Cuando se quiera introducir solo una restricción de velocidad (parámetro spd), se tendrá que introducir dos comas entre el punto de paso y la velocidad, como se ve en el **Código 2.18** más adelante.

La restricción de velocidad es la que se tiene en el tramo de navegación hacia el punto de paso. La restricción de altitud se alcanzará tan pronto como las demás restricciones de altitud lo permitan para subidas, y tan tarde como sea posible para descensos (lógica Top of Climb/Top of Descent). Para el nombre del punto de paso (wpname) también se puede introducir la posición del punto en lat/long, incluso se pueden introducir éstas haciendo click en un punto de la vista de mapa, o también se pueden introducir directamente en la línea de comandos, en formato grados'minutos'segundos (siendo el positivo de la latitud el Norte y el positivo de la longitud el Este). Habiendo visto esto, se introduce una ruta para la aeronave, con varias sintaxis diferentes, en el **Código 2.18**.

Código 2.18 Comando ADDWPT.

```
KL001 ADDWPT TOLEN
KL001 ADDWPT EBBR, FL100, 250
KL001 ADDWPT 50.3, 4.49
KL001 ADDWPT LFAF, , 0.80
KL001 ADDWPT BSN FL200
```

Por defecto, los puntos de paso se añaden al final de la ruta pero antes del destino. Se puede usar el comando AFTER (**Código 2.19**) para insertar un punto de paso en mitad de la ruta, o incluso usar el parámetro afterwp del comando ADDWPT al crear el punto de paso (**Código 2.19**).

Código 2.19 Comando ADDWPT - Orden en la Ruta.

```
KL001 AFTER LFAF ADDWPT LFAH
KL001 ADDWPT LFJS, , , LFAH
```

Si se quiere ver, editar, o suprimir las restricciones de una aeronave en un punto de paso, se usa el comando AT. En el **Código 2.20**, se tienen varios ejemplos de su uso.

Código 2.20 Comando AT.

```
KL001 AT LFAF
KL001 AT TOLEN SPD 240
KL001 AT EBBR DEL ALT
KL001 AT WOODY FL070/250
KL001 AT TOLEN ----/----
KL001 AT WOODY FL070/----
KL001 DELWPT LFJS
```

Durante el vuelo, es decir, con la simulación operativa, es posible controlar el punto de paso al que se debe dirigir la aeronave con el comando DIRECT, **Código 2.21**.

Código 2.21 Comando DIRECT.

```
KL001 DIRECT LFAH
```

A los puntos de paso introducidos con el formato lat-long se les da un nombre usando el identificador de la aeronave, seguido por el número de punto de paso creado de este modo precedido de ceros. Este nombre se puede usar como referencia para el punto de paso en los comandos AT, AFTER, DELWPT o como último argumento en el comando ADDWPT. La ruta se puede eliminar por completo con el comando DELRTE (**Código 2.22**).

Código 2.22 Comando DELRTE.

```
DELRTE KL001
```

Para cualquiera de los comandos, hacer click en la ruta mostrada en la vista de mapa seleccionará el punto de paso de la ruta, e introducirá sus parámetros identificadores en la línea de comandos como argumentos del comando que se esté utilizando. Esto también es posible cuando es más apropiado introducir el nombre del punto de paso en la línea de comandos.

Para asegurar que la aeronave sigue la ruta, el autopiloto debe estar conectado al FMS. Para el rumbo, esto se hace activando el modo LNAV (Lateral NAVigation), para altitud y velocidad (si se especifican restricciones al respecto), se hace activando el modo VNAV (Vertical NAVigation). VNAV solo puede estar activado mientras lo esté LNAV, de modo que habría que activar LNAV primero, como en el **Código 2.23**.

Código 2.23 LNAV y VNAV.

```
KL001 LNAV ON
KL001 VNAV ON
```

En el momento en el que se introducen restricciones de altitud o velocidad (con los respectivos comandos ALT y SPD), VNAV se desactiva y se pasa al modo ALT SEL o SPD SEL, mientras el otro parámetro vertical pasará a SPD/ALT HOLD. Esto significa que al introducir una restricción de este tipo lo que hace el autopiloto es mantener el parámetro no restringido, mientras que se alcanza el valor de la restricción en el parámetro restringido. Usar el comando acid VNAV OFF también desactiva el modo VNAV, continuando la aeronave en modo ALT HOLD y SPD HOLD (véase, mantiene su altitud y velocidad).

De manera similar, acid LNAV OFF también desactiva el modo LNAV, pasando a modo HDG HOLD (véase, mantiene el rumbo), mientras que una restricción con el comando HDG, cambia el modo lateral a HDG SEL (véase, se pone a intentar alcanzar la restricción de rumbo).

Por otro lado, los comandos de DEST y ADDWPT activan el modo LNAV, y el modo VNAV si hiciera falta, para que la aeronave tenga un comportamiento esperable para aquellos usuarios que no manipulan el autopiloto directamente. Cuando la ruta se ha introducido, ejecutar la simulación en modo cámara rápida (**Código 2.24**) mostrará a la aeronave volando la ruta a cámara rápida.

Código 2.24 Ejecutar Simulación a Cámara Rápida.

```
OP
FF
```

Como se ha comentado anteriormente, hay maneras de hacer que la ruta se muestre en la vista de mapa, el resultado de hacer esto se muestra en la **Figura 2.15**, con la ruta de una aeronave distinta.

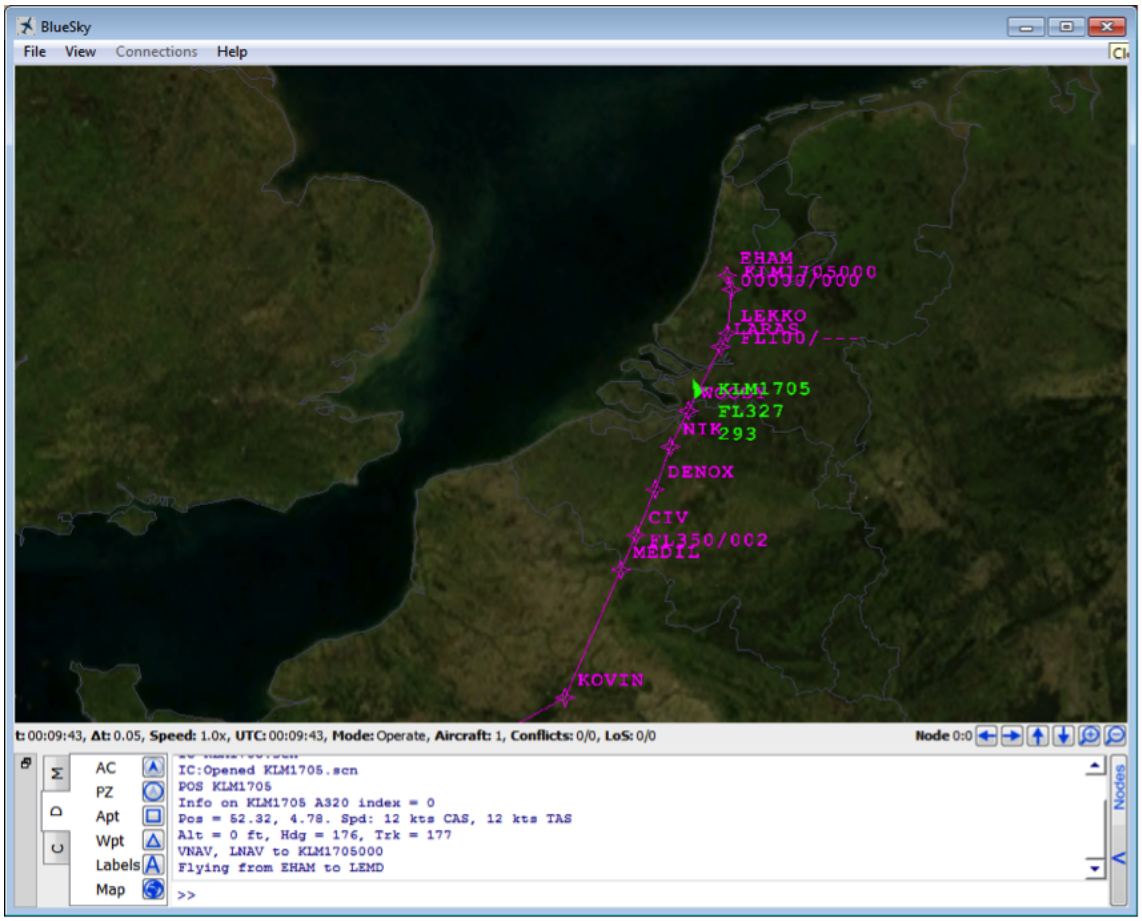

Figura 2.15 Ruta Mostrada.

Como se puede ver este Simulador se asemeja un poco en concepto al proyecto que se está realizando, pues es un simulador de ATC de código abierto desarrollado en una Universidad y con posibilidad de actualizaciones y ampliaciones. Una gran diferencia importante que tiene es que en BlueSky ATC no se modelan ni simulan las acciones que debe tomar el controlador, mientras que en este proyecto ése es uno de los focos principales. Mientras que BlueSky ATC está diseñado para la investigación, ATC Maker está pensado más para práctica de entrenamiento o incluso más orientado hacia el entretenimiento. Sin embargo, hay bastantes cosas que se pueden tomar como inspiración para ATC Maker. Entre ellas, el estilo de almacenamiento de datos, ya en el simulador anterior se ha mencionado que se podía extraer esto como característica de ATC Maker, aunque en ese caso se escogía el método de almacenamiento de datos para elementos del escenario, del simulador BlueSky ATC se pretende tomar el formato de toma de datos para almacenar las aeronaves y sus rutas.

Sí que es verdad que el modo de toma será muy distinto, pues en ATC Maker se va a tener por separado las ventanas de Editor de Escenarios y de Simulación, véase, no se va a poder editar el escenario mientras se está simulando ni se va a poder simular mientras se está editando. De esta manera, ciertas peculiaridades vistas en los tutoriales estudiados sobre como modificar la ruta de la aeronave en mitad de simulación no se van a implementar (con excepción, quizás, de aquellas que se asemejen a operaciones que realizaría un Controlador, como es el caso de la autorización para hacer directos). Del mismo modo, otras peculiaridades que sucedían al editar el "escenario" que se quería simular, causadas por el hecho de tener que hacerlo con la simulación en ejecución, no se aplicarán a ATC Maker.

2.3 Simulador Aurora

[12][13]El Simulador Aurora está desarrollado por IVAO (International Virtual Aviation Organization), una organización sin ánimo de lucro que opera la red de simulación de vuelo realista gratuita en la que se encuentra Aurora. Los usuarios pueden registrarse gratuitamente y a continuación conectarse a la Red IVAO (IVAN), como Controladores de Tráfico Aéreo o Pilotos virtuales, entrando e interactuando con otros usuarios en un Entorno Multijugador Masivo utilizando procedimientos, fraseología y técnicas de la Aviación real.

Figura 2.16 Logo de IVAO.

Esta organización cuenta con más de 240.000 miembros registrados, siendo una de las mayores redes de Simulación de Vuelo que permiten a los usuarios actuar como Pilotos o Controladores. Se apoya en su personal de voluntarios que ronda los 800 trabajadores. Los Controladores se conectan a la red usando el Cliente de Radar de IVAO, IvAc, o Aurora, el Cliente de ATC de IVAO que emula la interfaz de un puesto de Control moderno real. Los Pilotos, por su parte, se conectan usando sus Simuladores de Vuelo Propios y el cliente de piloto IvAp o con Altitude, el Cliente de Pilotos de IVAO. Esto significa que todos los usuarios (Pilotos y Controladores) interactúan en un servidor mundial dedicado que Simula el Tráfico Mundial de la manera más realista posible.

Manual de Usuario de Aurora

[14]Después de tener un resumen sobre quienes llevan el Simulador de Aurora (así como algunos detalles generales sobre este), se van a analizar ciertas partes que se han tomado como referencia para el desarrollo de ATC Maker del Manual de Usuario de Aurora. Dado que es un Manual bastante extenso, solo se analizarán en detalle aquellas secciones de las que se puede obtener referencia o inspiración para el desarrollo de ATC Maker con los objetivos determinados, analizando brevemente y a grandes rasgos aquellas secciones de las que no se pueda obtener referencia o inspiración para ATC Maker.

En este Manual se describe primero el cómo descargar y acceder a un Sector, lo cual no es de mucho interés para la aplicación de ATC Maker en el nivel de desarrollo que se va a llevar a cabo. A continuación se describe la Vista Principal del Cliente de Aurora, la cual consta de tres partes: una Barra Principal, una Barra de Opciones y la Pantalla de Radar. De entre estas tres partes, se estudian las dos últimas, pues la Barra Principal tiene más que ver con la información sobre el Usuario y los datos de uso, así como ajustes de la interfaz y de ciertas funcionalidades del equipamiento, etc. Las cuales no se necesitan en el caso de ATC Maker en su estado de desarrollo actual pues esos ajustes se asumen como fijos e invariables, para simplificar el programa.

Sobre la Pantalla de Radar no hay información específica en el Manual de Usuario, la información que se tiene es de ajustes que vienen en la Barra de Opciones. La Barra de Opciones consta de dos Barras de herramientas, una Barra de Preferencias y una de Menú.

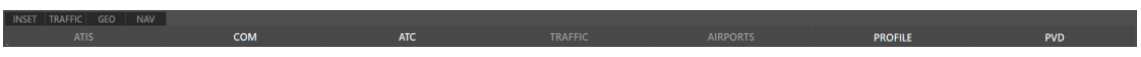

Figura 2.17 Barra de Opciones de Aurora.

Primero se Analizará la Barra de Menú, la cual contiene 7 apartados: ATIS, COM, ATC, TRAFFIC, AIRPORTS, PROFILE Y PVD. De la pestaña de ATIS no hay mucho que se pueda extraer para ATC

Maker, pues la información que contiene consta de información meteorológica (METAR), pistas activas, aproximaciones disponibles, NOTAM y más información que requieran los Pilotos, y esta información se asume fija para simplificar el problema de la Simulación.

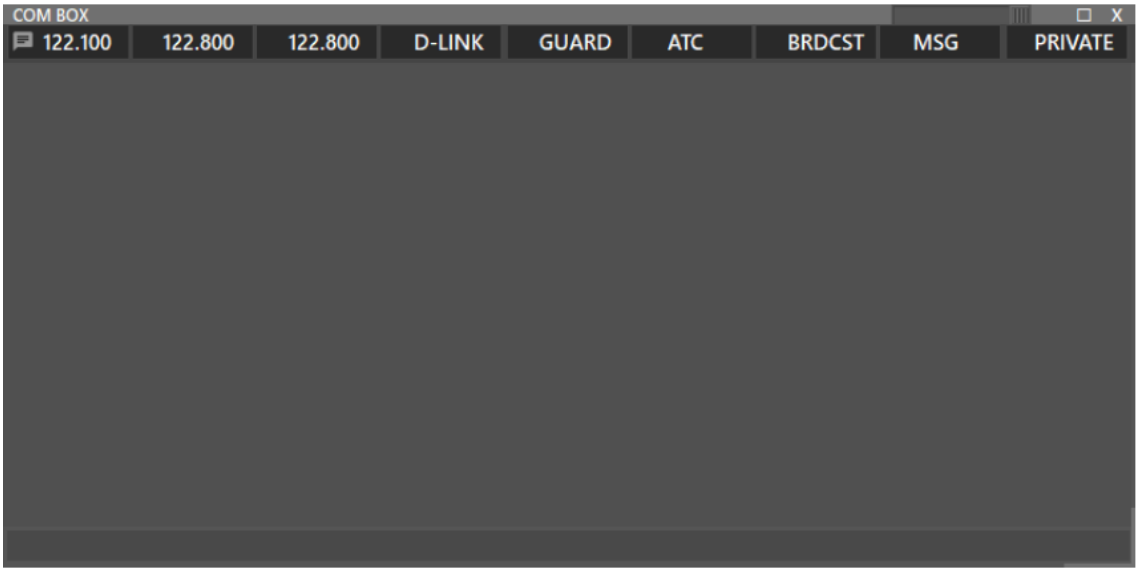

Figura 2.18 Pestaña de COM.

Con la pestaña de COM se abre una ventana en la cual se desarrolla la Comunicación por texto principal. Esta ventana puede tener hasta 8 ventanas distintas, la primera se corresponde a la Frecuencia Primaria del ATC, la cual se puede cambiar. La segunda y tercera ventanas opcionales representan otras frecuencias que el Controlador puede utilizar para monitorizar. La ventana de GUARD representa la frecuencia de emergencia, 121.5, por la que se pueden mandar mensajes a todos los Pilotos y Controladores cercanos, independientemente de la frecuencia que tengan sintonizada. La ventana de ATC contiene los mensajes de los Controladores que contienen información de difusión (como abrir o cerrar dependencia). La ventana de BRDCST (Broadcast) sirve para ver los mensajes que los Supervisores envían a toda la red. A la ventana MSG llegan los mensajes del Servidor, como son los Force Act. Por último, la ventana PRIVATE, que sirve para ver y enviar mensajes privados a diferentes estaciones (Pilotos, Controladores u Observadores). De esta pestaña se va a tomar referencia para el Simulador, pues al no disponer de un software de reconocimiento de voz integrado con MATLAB® , las comunicaciones en la Simulación serán por texto, y se planea introducir solo un Panel de Comunicaciones que haga las funciones de la pestaña de la Frecuencia Primaria, sin dar posibilidad de Cambiar de Frecuencia, pues no sería necesario.

En la pestaña de ATC se abre una ventana en la que se pueden ver a los Controladores u Observadores conectados cerca de la estación en la que se está (basado en el rango y el punto central). Permite ver datos sobre estos como el indicativo, su Frecuencia Primaria, su habilitación de ATC, su Nombre (si lo especifican) además de contar con una serie de indicativos sobre si está controlando, si está disponible para hacer transferencias, y dos botones, uno para mandar un mensaje privado a dicha estación y otro para solicitar su ATIS. De esta pestaña no se extrae nada relevante para ATC Maker, pues no se va a disponer de la posibilidad de comunicarse con otras estaciones de Control (ya que no se van a introducir en el archivo de datos del Escenario, al menos para el objetivo de desarrollo actual).

Pasando a la siguiente pestaña, la de TRAFFIC, abre una ventana en la que se ve una ficha de progresión de vuelo, que se puede ver en la Figura (). En esta ficha se observa información básica que concierne a la aeronave seleccionada, la cual incluye: en la primera columna el aeropuerto de destino (en la ficha de la figura UGGG), el tipo de reglaje de vuelo (I para instrumental, V para visual), y un código de transpondedor seleccionado por el Piloto (en la ficha de la figura 2000), en la segunda columna se tiene el Nivel de Vuelo Final (en la figura F390), el tipo de aeronave y su categoría (en la figura MD11 H), la velocidad en Mach (en la figura M083) y un aeropuerto alternativo (en la figura UGSB), en la tercera columna se tiene el indicativo de la aeronave, seguido de un recordatorio de su indicativo de radio (en la figura CWC552, Challenge Cargo), el aeropuerto de salida y la hora de salida (en la figura LKGR y 2150), por último la cuarta columna muestra la Ruta del Plan de Vuelo, y justo debajo el apartado de más información o Remarks.

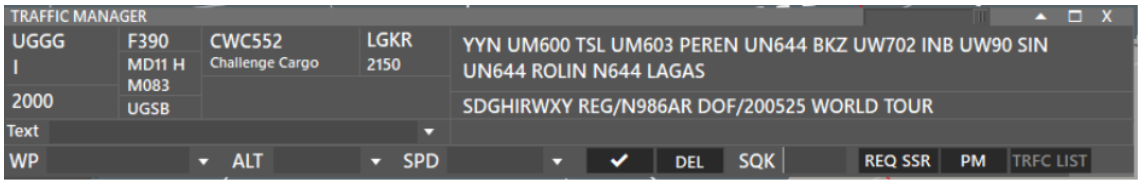

Figura 2.19 Ficha de Progresión de Vuelo de Aurora.

En la fila inferior se tiene una barra para que el Controlador pueda editar la Ficha de Progresión de Vuelo, en los campos WP, ALT y SPD el Controlador puede insertar una Radioayuda, STAR, SID o simplemente texto (en el de WP), y en los otros dos puede insertar Altitud y Velocidad respectivamente. Con los botones de "tick" y DEL inmediatamente al lado de estos apartados se puede enviar o eliminar la información de la etiqueta respectivamente (pulsar la tecla "enter" del teclado hace la misma función que el botón "tick"). En el cajón SQK (Squawk, Transpondedor) el Controlador puede asignar un código de transpondedor, el botón REQ SRR hace que el cliente le asigne automáticamente un código a la aeronave seleccionada. El botón PM permite abrir un chat privado en la ventana de COM para poder hablar con el piloto en privado. Por último, el botón TRFC LIST expande la ventana de TRAFFIC, abriendo la ventana de Gestión de Tráfico (Traffic Manager), y mostrará las aeronaves según los botones inferiores, que se pueden ver en la **Figura 2.20**: DEP para Salidas, ARR para llegadas, OVER para sobrevuelo y en cada uno de estos hay un botón de U, de UNCO, para los no controlados. Esta información se basa en lo seleccionado en la ventana de AIRPORTS que se verá más adelante.

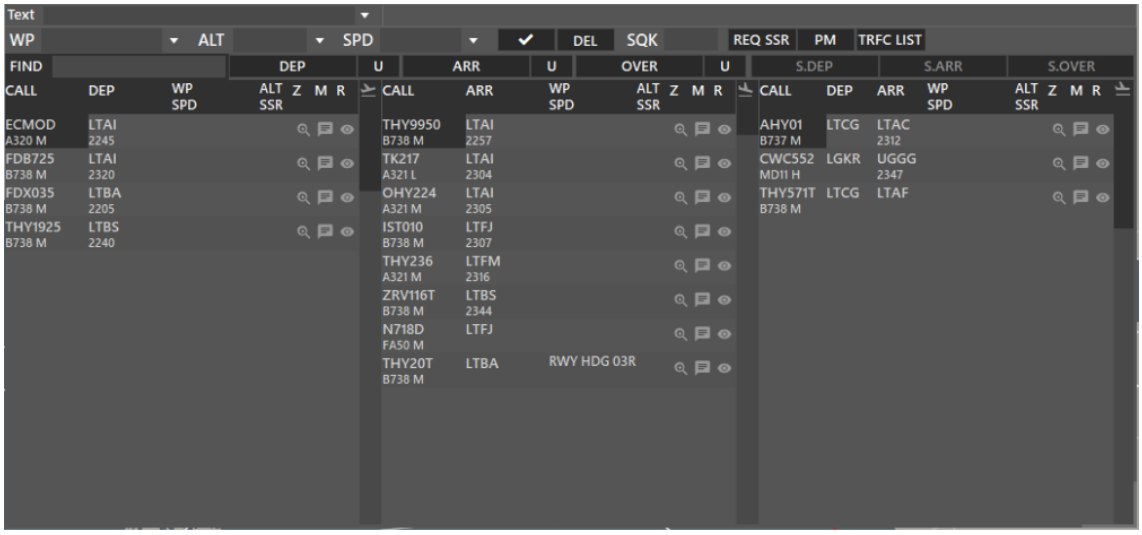

Figura 2.20 Traffic Manager de Aurora.

En la ventana del Traffic Manager se pueden observar tres columnas, que representan las Salidas (DEP), Llegadas (ARR) y Sobrevuelos (OVER), en cada una de ellas se pueden mostrar los vuelos no Controlados activando el botón U al lado del respectivo botón de la columna correspondiente. Adicionalmente si se activa o desactiva el botón de una columna, está se hace visible o se deja de poder ver, así como también se pueden usar los botones S.DEP, S.ARR o S.OVER para ver la columna correspondiente en una ventana a parte de la del Traffic Manager. Estas ventanas por separado contienen todas las mismas 8 columnas, que también las tienen cuando están dentro del Traffic Manager, con la excepción de la columna ARR en la de Salidas y la columna DEP en la de Llegadas. Estas columnas son: CALL, muestra el indicativo, tipo y categoría de la aeronave, DEP, muestra el aeropuerto y hora de salida (la hora solo para Salidas), ARR, muestra el aeropuerto y hora estimada de llegada (en la columna/ventana de Salidas no se muestra la hora, que es calculada por Aurora), WP/SPD, muestra el Punto de Ruta asignado y velocidad, ALT/SSR, muestra la Altitud y el Código de Transpondedor asignados, Z, muestra un icono de lupa que al clicarse hace zoom en el tráfico, M, muestra un icono de diálogo que al clicarse abre una ventana de chat privado, y R, que muestra un icono de un ojo que hace que la Ruta del avión seleccionado sea visible o invisible.

De esta sección del Simulador Aurora se pueden tomar referencias para ATC Maker, como el diseño de las

Fichas de Progresión de Vuelo, así como las listas de Tráfico, aunque de estas listas solo se considerará la de Sobrevuelo, pues es el tipo de Tráfico que se va a Simular.

En la siguiente pestaña, la de Aeropuertos, se tiene información sobre los aeropuertos del Sector, como su informe METAR (si está disponible) el cual, si es válido, proporciona información sobre el viento y el reglaje QNH, también ofrece información sobre si el aeropuerto está siendo controlado e información sobre las pistas disponibles, sus cursos magnéticos, si están seleccionadas como de Salida o de Llegada, el viento en la pista, etc. Además también se puede obtener información sobre las SIDs y las STARs, así como de los procedimientos de Aproximación. De esta pestaña no se puede obtener información relevante para ATC Maker pues no se va a trabajar con Salidas ni Llegadas, ni Aproximación tampoco, además de que se considera tiempo atmosférico en calma.

En la pestaña de Perfil se pueden guardar ajustes para agilizar el proceso de conexión a un puesto de Control en sesiones futuras. De este apartado no se puede obtener nada para ATC Maker, pues la funcionalidad de este tiene que ver con la característica multijugador y online de Aurora.

En la última pestaña de la Barra de Menú, la de PVD, se tienen varias secciones. Una primera sección abre la ventana de Sectores, donde se puede ver una lista de Sectores descargados, así como algunos Sectores que se pueden descargar, en la parte donde se tienen los Sectores descargados están los botones para cargar un Sector o eliminarlo de la lista de descargas (y del equipo), así como un botón para actualizar automáticamente la información de los Sectores. La segunda sección que se tiene es la de Colores, donde se puede establecer un esquema de colores personalizado para los distintos elementos que se representan en la Pantalla de Radar, estos esquemas de colores se guardan en los Perfiles. La tercera sección es la de Sonidos, donde se pueden ajustar los dispositivos de salida, ensordecer sonidos o incrementar el volumen de notificaciones. Por último se tiene una sección de Ajustes generales, que incluye ajustes de Radar, Comunicaciones, Conflictos, Etiquetas, entre otros. De esta Barra se puede tomar referencia para ATC Maker sobre el método de selección de Sector a Simular, así como guías de Colores para representar el Sector y sus distintas partes en la Simulación.

Pasando a la Barra de Preferencias, esta consta de 4 secciones principales que pueden extenderse pulsando sobre los botones: INSET, TRAFFIC, GEO y NAV. El botón de INSET, cuando está activo, permite obtener hasta 8 ventanas de Radar, de manera que, si se está controlando un área muy grande, se puedan ver múltiples aeropuertos simultáneamente. De esta sección no se puede obtener mucho para ATC Maker, pues solo se va a tener una pantalla de Radar.

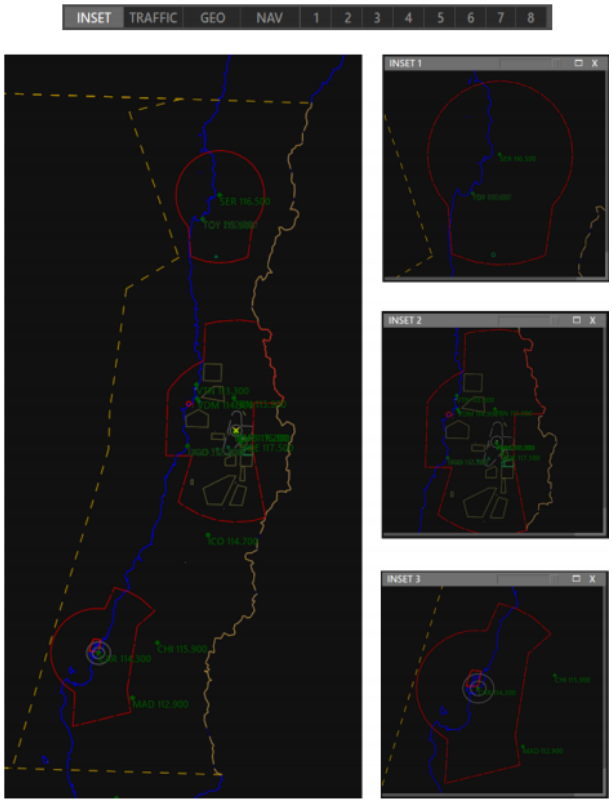

Figura 2.21 Pantallas del INSET.

El botón de TRAFFIC, cuando está activo, permite trabajar con las etiquetas de las aeronaves, añadirles un Halo y Anillos, y controlar la representación del Vector de Velocidad. Lo que permite hacer esta sección es ajustar la distancia de la etiqueta a la representación del avión, crear y editar el radio de un Halo alrededor del avión, crear Anillos alrededor del avión, representar el vector de velocidad en cantidades de tiempo concretas, determinadas en números de minutos (hasta 8 minutos) y por último hay un botón GTR que al activarse representa el tráfico en tierra. De esta sección se puede tomar referencia para la representación de las aeronaves en la Simulación, ver la información que añadir a la etiqueta, así como el representar el vector de velocidad en minutos, adicionalmente, y como se puede ver en las imágenes (**Figura 2.22**), también se puede tomar como referencia el mostrar el rastro del avión con una línea de puntos.

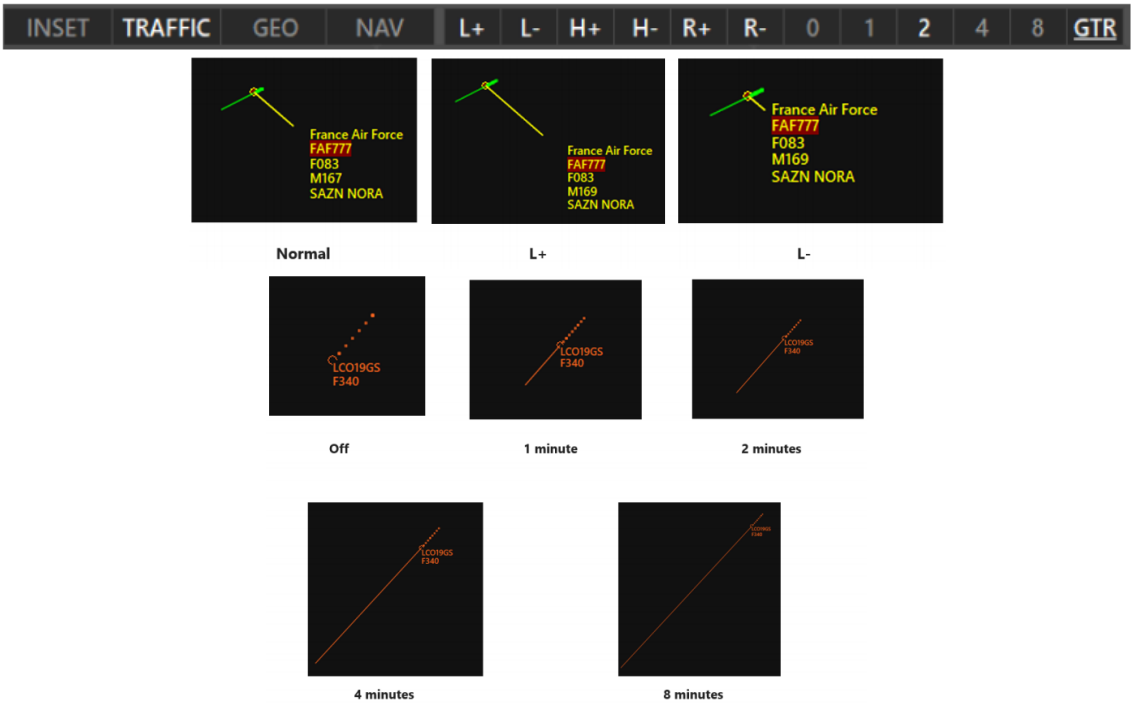

Figura 2.22 Barra de Preferencias - TRAFFIC (1) Barra de TRAFFIC (2) Distancia de la Etiqueta (3) Vector Velocidad.

El botón GEO, cuando está activo, permite mostrar u ocultar cierta información sobre el Sector en la Pantalla de Radar, como aeropuertos, pistas edificios, entre otros, que se pueden ver en la **Figura 2.23**. De esta sección no se puede obtener mucha información para ATC Maker, pues al tener solo una Pantalla de Radar interesa que se pueda ver toda la información en ella, además de que complicaría el proceso de representarlo en MATLAB® , además de que no se va a incluir tanta información como para hacer difícil el verla toda al mismo tiempo.

Figura 2.23 Barra GEO.

El botón NAV, de forma similar al botón GEO, cuando está activo permite mostrar u ocultar información en la Pantalla de Radar, relacionada con la Navegación, como VOR, NDB, Fijos, Frecuencias, Aerovías, entre otros, que se pueden ver en la **Figura 2.24**. Del mismo modo que con GEO, de esta sección no se va a tomar referencia.

Figura 2.24 Barra NAV.

Y con esto se llega al final del Manual de Usuario de Aurora, quedaría un apartado sobre Atajos de Teclado,

del cual no se puede tomar referencia pues en ATC Maker no se van a incluir atajos de teclado debido a la complejidad de Programarlos en MATLAB®

3 Desarrollo del Simulador

En este apartado se va a explicar el desarrollo de los diferentes aspectos del Simulador de ATC que se ha desarrollado. Antes de nada, hay que determinar los casos que se van a contemplar, que son el de control de área, con disponibilidad de multiradar, con cielos despejados, en FIR con RVSM. Se dejan los demás casos como posibles futuras actualizaciones.

Lo siguiente a determinar es el lenguaje y la plataforma de programación, se ha elegido el diseñador de aplicaciones de MATLAB® para ello, dado que MATLAB® es un lenguaje de programación que se ha usado repetidamente a lo largo del Grado, y por ende, se está muy familiarizado con él, además de que se ha usado en varias ocasiones para cálculos relacionados con la navegación, aviónica y aerodinámica, así como para la mecánica de vuelo. Esto hace de MATLAB® una muy buena herramienta para hacer los distintos cálculos requeridos para este proyecto, además de que también se ha usado durante el Grado para representar espacios aéreos con sus rutas y zonas restringidas. En concreto se ha usado la versión R2019a de MATLAB® , que es la que se tenía a mano de las que tiene integrada la herramienta de Diseño de Aplicaciones.

Una vez establecidos los objetivos y los medios para el desarrollo del Simulador, se procede a describir el diseño, desarrollo y funcionamiento del mismo.

3.1 Interfaz de Inicio

La Interfaz de entrada consiste en una Pantalla Inicial, en la que existen dos botones: uno que dice “Editor de Escenarios” y otro justo debajo que dice “Simulador”. Cada uno de estos botones lleva a una Pantalla de Selección de Escenario para el Editor y el Simulador respectivamente. En estas Pantallas de Selección de Escenario se tiene en ambas un objeto tipo árbol, un objeto del Diseñador de Aplicaciones de MATLAB® que permite crear listas jerarquizadas de datos. En estos árboles, que se modifican simultáneamente, se tiene una lista de los distintos escenarios que se carga al iniciar el simulador, se puede modificar en la misma aplicación y se guarda en una variable tipo estructura, un tipo de variable que actúa como un directorio y que permite guardar distintos tipos de datos en una sola variable, y esta se guarda como un archivo .mat (que se ha denominado “escenarioslist.mat”) y se guarda en una carpeta que actúa como librería de la aplicación llamada libapp. En dicha carpeta de librería se guardan también los archivos .mat que contienen las variables tipo estructura en las que se guardan los datos necesarios para simular y editar el escenario en concreto.

En la Pantalla de Selección de Escenarios del Editor, a parte del Árbol ya mencionado, se dispone de una serie de botones que permiten crear, renombrar, eliminar y cargar el escenario seleccionado del Árbol, además de un botón que dice “Atrás”, que devuelve a la pantalla de inicio. El funcionamiento de los distintos botones es el siguiente: los botones de “Nuevo Escenario” y “Renombrar Escenario” abren sendas ventanas para crear y renombrar los escenarios, el botón de “Eliminar Escenario” elimina el nodo del escenario seleccionado en el Árbol, borra dicho escenario del archivo “escenarioslist.mat” y también se borra el archivo .mat donde se guardan los datos del escenario, por último se tiene el botón de “Cargar Escenario” que al pulsarse abre la Pantalla de Editor de Escenarios y se carga la variable del archivo .mat del escenario seleccionado en el Árbol, si no hay un nodo del Árbol seleccionado, este botón no hace nada al ser pulsado.

Las ventanas que se abren al pulsar los botones de “Nuevo Escenario” y “Renombrar Escenario” cuentan ambas con un botón de “Atrás” para cerrarlas y volver a la Selección de Escenario, además de un Campo de Edición de Texto (un tipo de objeto del Diseñador de Aplicaciones de MATLAB® que permite guardar y editar texto en variables del sistema) para escribir y editar el nombre del Escenario en cuestión. En la parte

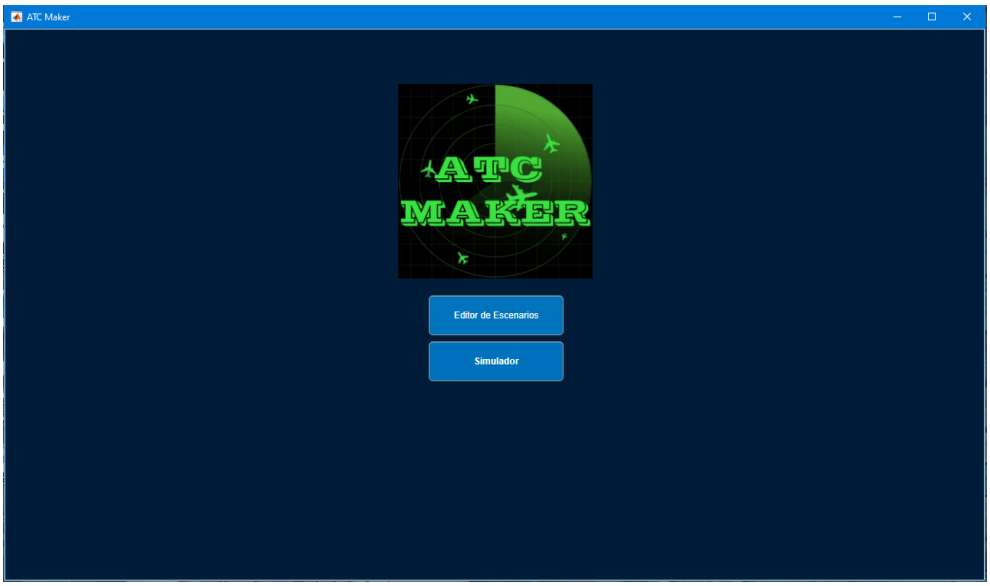

Figura 3.1 Interfaz de Inicio.

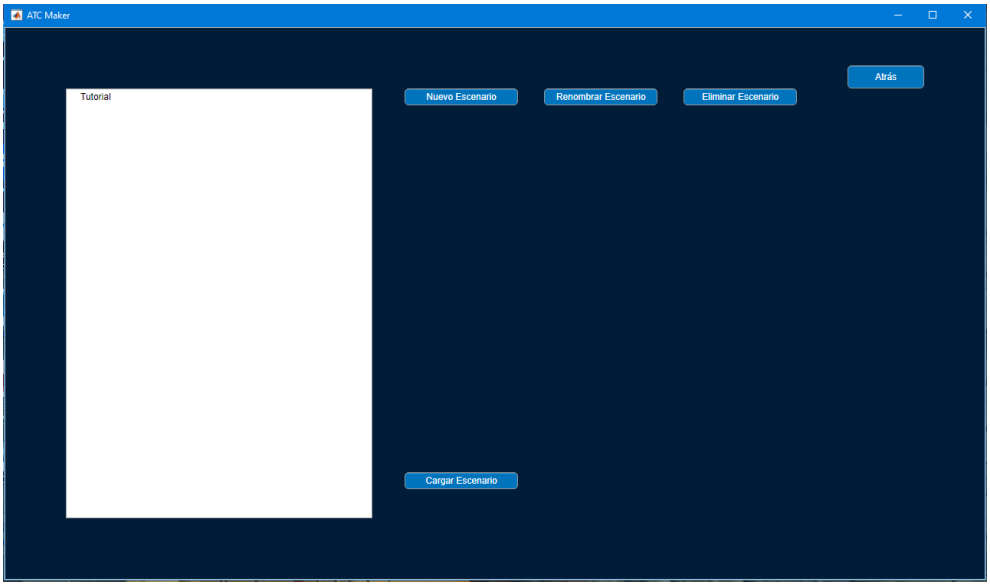

Figura 3.2 Pantalla de Selección del Editor.

en la que se diferencian es en que la de "Nuevo Escenario" cuenta con un botón de "Crear", que cierra la ventana, abre el Editor de Escenarios y crea la variable estructura del nuevo escenario, además de añadir el nombre del nuevo escenario en "escenarioslist.mat" y en los Árboles de las Pantallas de Selección de Escenarios. La ventana de "Renombrar Escenario", por su parte, no se abrirá a no ser que haya un escenario seleccionado en el Árbol, y su botón dice "Renombrar" en vez de "Crear", al pulsarlo, cierra la ventana y cambia el nombre del escenario en escenarioslist.mat, así como en los Árboles de la Selección de Escenario y en la variable estructura del escenario en cuestión, pero no abre la ventana del Editor de Escenarios.

En la pantalla de Selección de Escenarios del Simulador, en cambio, a parte del Árbol y de un botón de "Atrás" como en la del Editor, solo hay un botón de "Cargar Escenario", el cual solo funciona si hay un nodo del Árbol seleccionado, que carga la variable guardada en el .mat del escenario seleccionado en el Árbol y abre la Pantalla del Simulador. En las (**Figura 3.1**, **Figura 3.2**, **Figura 3.3** y **Figura 3.4**) se pueden ver todas las pantallas y ventanas descritas.

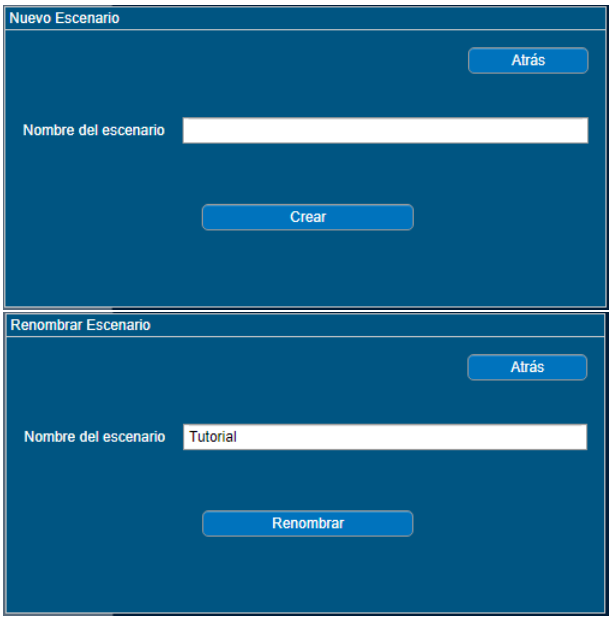

Figura 3.3 Paneles para (1) Crear y (2) Renombrar Escenarios.

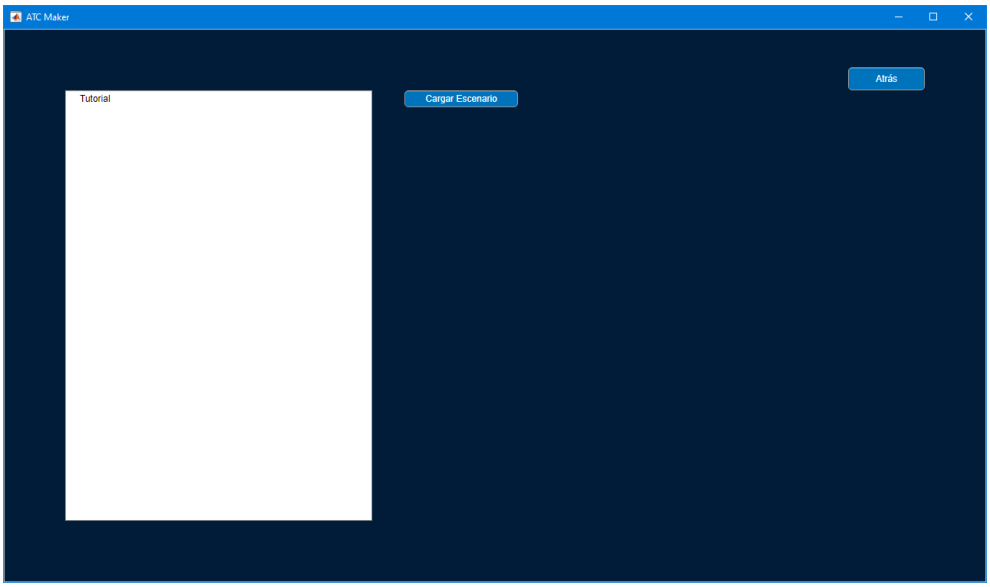

Figura 3.4 Pantalla de Selección del Simulador.

3.2 Editor de Escenarios

En la (**Figura 3.5**) se ve el estado de la Pantalla del Editor de Escenarios al entrar en ella. Como se puede ver cuenta con un objeto Árbol con cinco nodos predefinidos, un botón de "Atrás" y un objeto tipo Ejes (un objeto del desarrollador de aplicaciones de MATLAB® que permite representar datos con el comando plot), el cual representa el Sector del Escenario. Si el Escenario es recién creado o no contiene dato alguno, el objeto ejes está en blanco, sin ningún dato representado. También se pueden observar cuatro Campos de Edición de Texto que no se pueden editar por el Usuario y un Panel de un azul más claro que contiene otros dos Campos de Edición de Texto y un Botón de "Guardar". A parte, en el lateral derecho se aprecia un botón que dice "Guardar Cambios", el cual almacena los datos actuales de la variable estructura "Sector" en el archivo .mat del Escenario en Edición.

Los Campos que quedan fuera del Panel (cuyo título es Horario, como se ve en la figura), sirven para representar los límites verticales del Sector y de las Zonas Restringidas dentro del mismo. Estos Campos están en blanco si no se han introducido o, en el caso de los de las Zonas Restringidas, si no se ha seleccionado

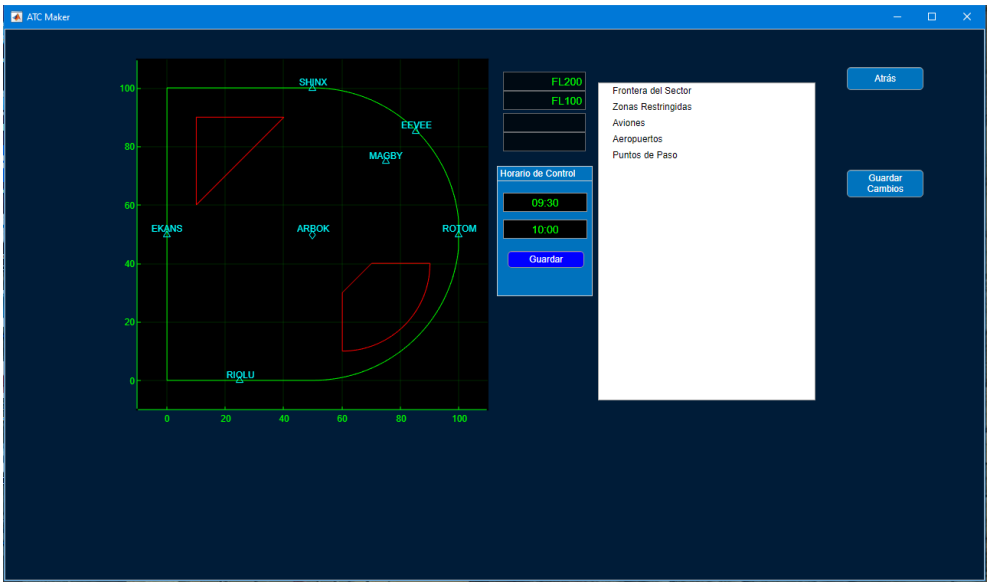

Figura 3.5 Pantalla del Editor.

ninguna Zona Restringida en el objeto árbol correspondiente que se analizará más adelante. El Panel "Horario" sirve para apuntar el período de Control, apuntando las horas de inicio y final en los Campos de Edición de Texto en formato "hh:mm".

También se aprecia en la **Figura 3.5** un espacio por debajo bastante amplio, en este espacio es donde aparecen los Paneles de Edición de los distintos aspectos del Sector, que están asociados a los nodos del Árbol, de modo que cuando se selecciona uno de los nodos, se muestra el Panel correspondiente.

Entre los distintos Aspectos que se pueden Editar se encuentran: La Frontera del Sector, Las Zonas Restringidas, Los Aviones que van a cruzar el Sector, Los Aeropuertos/Aeródromos que hay en el Sector y Los Puntos de Paso que hay en el Sector (Normalmente estos están Asociados a Radioayudas).

En el Panel de Edición de la Frontera (**Figura 3.6**) se pueden determinar los límites horizontales de la Frontera del Sector, que pueden ser de dos tipos: Vértices (Líneas Rectas) o Arcos (Curvas). También se pueden determinar los límites verticales, que se pueden establecer como altitudes en pies (ft) o en Niveles de Vuelo (FL). Ambos límites se representarán, los horizontales en los ejes, representando el plano del sector, y los verticales en los Campos de Edición de Texto Mencionados al principio, que no pueden ser editados por el Usuario. También se tiene un objeto Árbol, en el que se pueden ver y editar los distintos tramos geométricos de la Frontera del Sector. Adicionalmente, cuando se selecciona un nodo del Árbol de la Frontera, en los ejes se remarca el tramo o punto característico del tramo geométrico de la Frontera que representa dicho nodo.

En el Panel de Edición de Zonas Restringidas (**Figura 3.7**) se pueden determinar, de la misma manera que con la Frontera, los límites horizontales y verticales de las Zonas Restringidas. Estos límites se representan del mismo modo que con la Frontera, con la diferencia de los límites verticales, de los cuales solo se muestran los de la Zona Restringida seleccionada en el Árbol del Panel de Zonas Restringidas. Si no hay ninguna Zona Restringida seleccionada (o alguno de sus tramos geométricos), no se muestran los límites verticales. Adicionalmente, se tiene un botón extra con respecto al Panel de la Frontera, que dice "Nueva Zona Restringida", y crea un nodo en el Árbol representando una nueva Zona Restringida, los tramos geométricos de cada Zona Restringida se guardan como subnodos o nodos "hijos" del Nodo de la Zona Restringida. Por último, y del mismo modo que con la Frontera, al seleccionar un nodo del Árbol se remarca el correspondiente tramo geométrico que representa dicho nodo, cuando el nodo representa a una Zona Restringida entera, se remarca dicha Zona Restringida entera en los ejes.

En el Panel de Aviones (**Figura 3.8**) contamos también con un objeto Árbol que enumera los aviones creados, además de botones para crear y eliminar dichos aviones (el de eliminar solo funciona si hay algún nodo del Árbol seleccionado) y a parte un objeto tipo grupo de pestañas (del desarrollador de aplicaciones de MATLAB®), con tres pestañas para rellenar la información del vuelo de cada avión. La primera pestaña es de Información General del Avión (**Figura 3.9**): Matrícula, Nivel de Vuelo, Velocidad de Vuelo (en nudos [kt]), ETO o Tiempo de Entrada al Sector (Estimated Time Over), y Rumbo de Entrada al Sector.

La matrícula actúa como el indicativo de llamada del avión y se ha tomado el formato de las matrículas de aviones españoles para simplificar (pues MATLAB® suele tener problemas a la hora de comparar variables

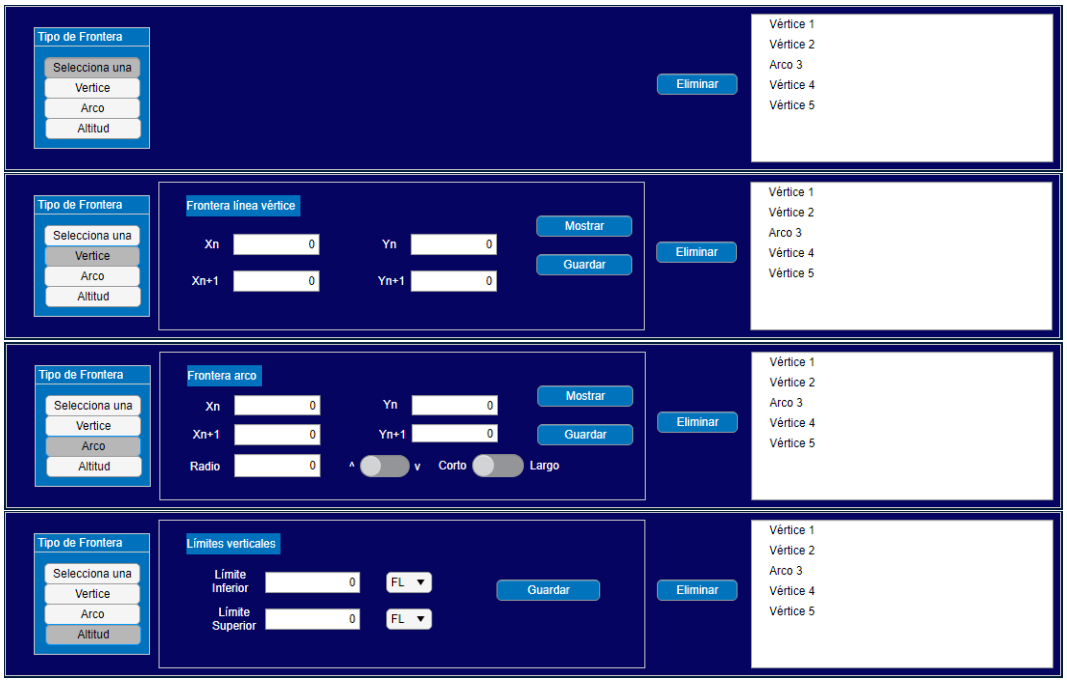

Figura 3.6 (1) Panel de Edición de Frontera (2) Tipo Vértice (3) Tipo Arco y (4) Límites Verticales.

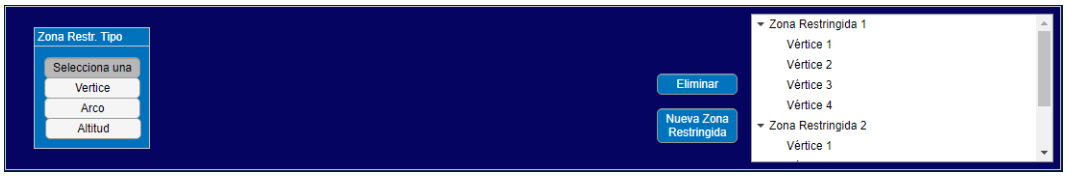

Figura 3.7 Panel de Edición de Zonas Restringidas.

Figura 3.8 Panel de Edición de Aviones.

de tipo cadena de caracteres), el cual incluye, para aeronaves comerciales, los identificadores desde EC-AAA hasta EC-WZZ. La ETO y el Rumbo de Entrada se almacenan para calcular la aparición del vuelo durante la Simulación, la ETO debería estar entre la Hora de Inicio y Final, se guarda como una hora en formato “hh:mm”, y es recomendable que sea por lo menos 1 minuto después de la Hora de Inicio y por lo menos antes de la mitad del tiempo de simulación, si se pretende que salga del Sector, el Rumbo se almacena para poder saber de qué dirección viene el avión cuando está a punto de llegar y así poder representar su posición inicial.

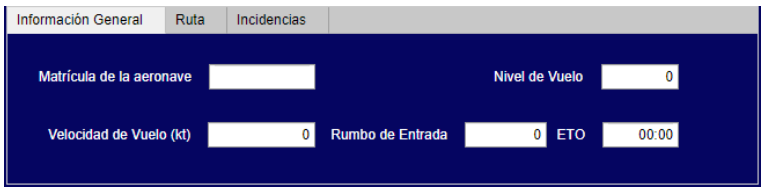

Figura 3.9 Pestaña de Información General.

También se almacenan los datos sobre el Nivel de Vuelo de entrada y la velocidad de Vuelo, los cuales

son muy necesarios para la Simulación del Vuelo. Cabe destacar que no se tiene ninguna forma de toma de datos sobre el tipo de aeronave en cuestión, esto se debe a que el proyecto ya era lo suficientemente complejo de por sí para la extensión que suele tener un TFG, además de para la cantidad de personas trabajando en él (en este caso, solo una), de modo que se ha asumido que todas las aeronaves tienen el mismo modelo de actuaciones y se han tomado datos concretos del A320 cuando han sido necesarios [15].

La segunda pestaña consta de los botones y medios para almacenar la Ruta del Vuelo (**Figura 3.10**), con un objeto Árbol que almacena los Puntos de Paso de la Ruta, un Campo de Edición de Texto en el que escribir los identificadores de dichos puntos de Paso (si no coincide con ninguno, da un Error y no lo guarda), así como botones para "Guardar" y "Eliminar" dichos Puntos de Paso en la Ruta. Si se tiene un nodo del Árbol de Rutas seleccionado, el Punto de Paso que se guarda se guarda como el siguiente en la Ruta al del nodo seleccionado.

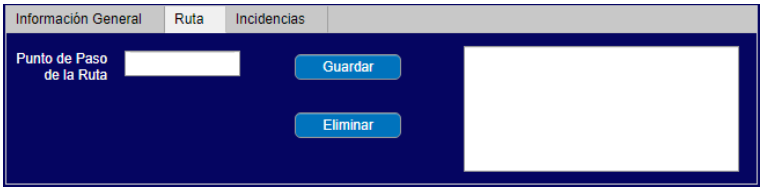

Figura 3.10 Pestaña de Ruta.

La tercera pestaña se llama Incidencia (**Figura 3.11**), esta pestaña está pensada para almacenar un tipo de Incidencia que pueda surgir en Ruta para la Aeronave seleccionada. Entre las posibles Incidencias, se han seleccionado tres posibilidades que se consideran de entre las más probables que se puede encontrar un Controlador: Una Solicitud de Cambio de Nivel de Vuelo (Tipo 1), Una Solicitud de Directo (Tipo 2) y Un Cambio de Rumbo No Programado (Tipo 3). La selección del tipo de la Incidencia se hace mediante un objeto tipo desplegable, y se tiene en cuenta la posibilidad de que no haya una incidencia de ningún tipo. Para cada Tipo de Incidencia se tiene un Punto Detonante o "Trigger", que se establece mediante un Campo de Edición de Texto, este Punto debe ser un Punto de la Ruta, también se guarda un objetivo o "Target", cuyo tipo varía según el tipo de Incidencia (Nivel de Vuelo para las de Tipo 1, un Punto de Paso para las de Tipo 2, y un Rumbo para las de Tipo 3), por último, se tiene un desplegable en el que se determina el "Tiempo" de ejecución de la Incidencia, véase, si se efectúa antes o después del "Trigger" (por defecto está en antes, y se recomienda que así sea, pues es más fácil de simularlo así).

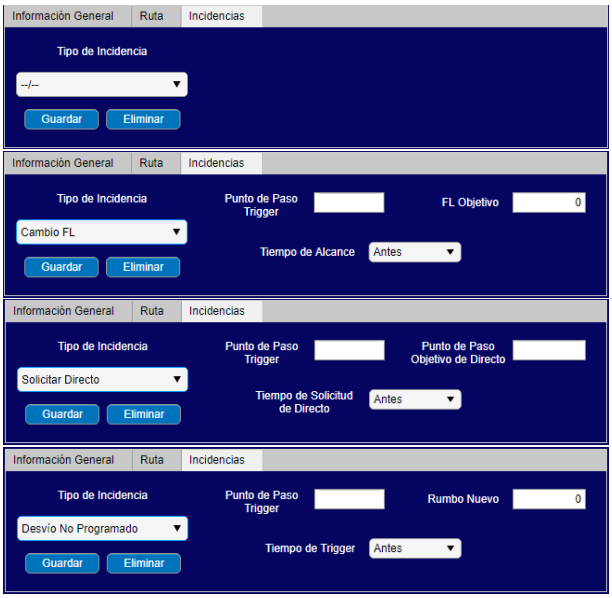

Figura 3.11 Pestaña de Incidencias, (1) para Tipo 0, (2) para Tipo 1, (3) para Tipo 2 y (4) para Tipo 3.

Todos estos datos se guardan en la variable del Sector, bajo el subgrupo de las Aeronaves/Aviones, en concreto bajo el Avión seleccionado en el Árbol de Aviones. Adicionalmente, cuando se selecciona un Avión

en este Árbol, se representa en los ejes la Ruta que dicho Avión pretende seguir.

En el Panel de Aeropuertos (**Figura 3.12**) se tiene una configuración relativamente simple, un Campo de Edición de Texto en el que poner el Identificador del Aeropuerto, así como dos Campos de Edición Numéricos para escribir la posición del mismo, y tres botones, uno para guardar el Aeropuerto, otro para eliminar un Aeropuerto seleccionado y un tercer botón que sirve para añadirle Pistas de despegue y aterrizaje al Aeropuerto, abriendo un Panel para ello en el que se puede añadir un par nuevo de pistas o editar un par de pistas ya existente (se habla de pistas por pares dado que tienen dos sentidos posibles de uso). Cuando se guarda un Aeropuerto se crea un nodo en un objeto Árbol que almacena los Aeropuertos creados y les crea un nodo "hijo" con la posición del Aeropuerto.

Figura 3.12 (1) Panel de Aeropuertos (2) Panel de Pista Nueva (3) Panel de Edición de Pista.

Para los identificadores de Aeropuertos se ha seguido un criterio que usa OACI para los Puntos de Paso, véase, Fijos relevantes o estaciones de Radioayudas, llamado el 5LNC (5-Letter Name Code) [17], que consiste en asignarle un código de 5 letras que sea fácilmente pronunciable, de manera que se pueda decir por los canales de Radio de Control (que suelen tener bastante distorsión) y se puedan identificar inequívocamente. De esta manera se asegura que MATLAB® no tendrá problemas a la hora de comparar las variables tipo cadena de caracteres de los Nombres, ya que tienen siempre la misma longitud, y es relativamente fácil detectar si un nombre es válido (se comprueba si está entero en mayúsculas), el límite de que sea fácilmente pronunciable no es muy estricto pues es difícil de comprobar dado que la pronunciación es algo bastante subjetivo, de manera que se deja a manos del Usuario el que considere si un Nombre es pronunciable o no. El criterio real usado por OACI para los identificadores de Aeropuertos consiste en un Nombre de longitud variable que cuenta con un prefijo que determina la región en la que se encuentra, seguido normalmente de un indicador de tres caracteres que distinguen claramente al Aeropuerto en cuestión, pero no tiene en cuenta en ningún momento la pronunciación de dicho identificador, pues usualmente se usa el Alfabeto Radiofónico para este tipo de casos.

En el Panel de Puntos de Paso (**Figura 3.13**) tenemos una configuración similar, sino casi igual que con los Aeropuertos, con la diferencia de que en vez de un botón para añadir Pistas de despeje y aterrizaje, se tiene un botón que mueve el Punto de Paso al Punto de la Frontera más cercano, convirtiendo al ûnto en un Punto Fronterizo, y abre un Panel en el que se puede asociar una Frecuencia de Radio que represente la Frecuencia de Radio del Sector Adyacente que comparte el tramo de Frontera en el que está el Punto con el Sector Editado. Este Dato se almacena entre los datos de Simulación pues sirve para realizar una de las funciones que un Controlador hace más a menudo, los cambios de Sector, con este dato de la Frecuencia del sector, se puede mandar dicha Frecuencia al Avión durante la Simulación, y considerar este hecho como la salida del Avión del Sector. Para los nombres de los puntos de paso se sigue el mismo criterio que para los Aeropuertos, pues de esta manera se pueden comparar ambos grupos a la vez sin temor a que MATLAB® cause errores, además de que el criterio lo utiliza OACI para dar nombre a los Puntos Relevantes. El criterio de OACI para identificar los Puntos de Paso, tiene en consideración también el que hay un número limitado de posibilidades, por ello se coordina entre oficinas regionales para que los puntos con nombres iguales estén lo suficientemente separados como para que no haya confusiones. Para los Puntos Asociados a Radioayudas se usa el código identificador de dicha Radioayuda, el cual puede variar mucho tanto en longitud como en cualidad de pronunciable, estas diferencias de longitud en los identificadores es lo que nos lleva a usar el criterio 5LNC [17], pues MATLAB® no tiene un proceso fácil y rápido para comparar cadenas de caracteres de longitudes distintas, además, debido a que los Sectores que se crean solo existen en el Escenario de

Simulación, no debería haber problemas de duplicidad de nombres en este caso concreto.

Figura 3.13 (1) Panel de Puntos de Paso (2) Panel de Frecuencia.

3.3 Simulador

Figura 3.14 Pantalla del Simulador.

En la **Figura 3.14** se ve el estado de la Pantalla del Simulador cuando se entra en ella, en concreto cuando se carga un escenario de Ejemplo llamado Tutorial que se ha creado para demostrar las capacidades del programa y sobre el cual se entrará más en detalle en Apartados más adelante.

En esta Pantalla se puede ver un Panel a la Izquierda (**Figura 3.15**), este Panel hace las veces de una herramienta que usan comúnmente los Controladores que consiste en una serie de Raíles en los que colocan las distintas Fichas de Progresión de Vuelo, generalmente en un orden que representa los niveles de vuelo autorizados para cada Aeronave, de manera que pueden distinguir claramente cuales aeronaves podrían causar conflictos entre ellas. En este programa lo que se ha hecho es crear "botones de estado" (un objeto del desarrollador de aplicaciones de MATLAB®) que actúan en esencia como interruptores, cuando cambia su "valor" (on/off) se mantienen en él. Estos botones son distintos de los que se han usado hasta este punto en el programa, los cuales son "botones de presión", los cuales siempre se mantienen en valor off y se accionan al ser pulsados.

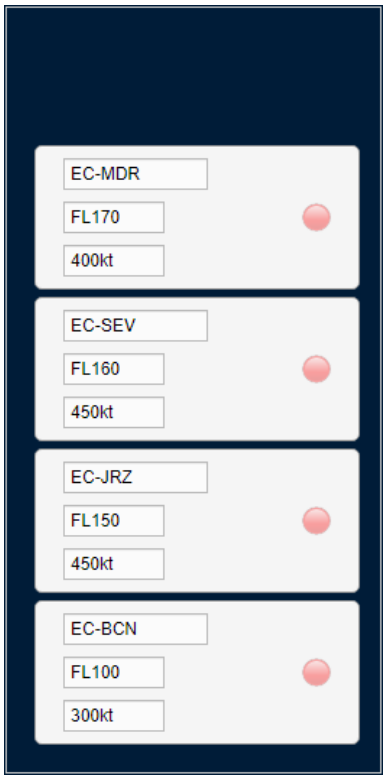

Figura 3.15 Panel de Raíl de Fichas.

Estos Botones de Estado se crean con una serie de Campos de Edición de Texto por encima (los cuales no pueden ser editados por el Usuario), que muestran el indicador de la aeronave, su Nivel de Vuelo (FL) y su Velocidad de Vuelo en Nudos (kt). Además, se crea también por encima del botón un objeto tipo Lámpara de Color Rojo (un objeto del desarrollador de aplicaciones de MATLAB®), la cual se deja apagada por defecto y se ilumina cuando se detecta una alerta o conflicto con el Avión correspondiente. La funcionalidad de estos botones es que cuando se activan (pasan a estar en on), se desactivan todos los demás y se rellena el Panel Inmediatamente Debajo del Panel de Raíl de Fichas, llamado Ficha de Progresión de Vuelo, con los datos del Avión correspondiente. Adicionalmente, cuando se crean estos botones, se ordenan por Altitud de más alta a más baja, además, en el caso de que un avión cambie de Altitud durante el Vuelo, se representa en el Campo correspondiente y si este Cambio de Altitud lleva que los botones se tengan que reordenar, ocurre automáticamente.

En el Panel de Ficha de Progresión de Vuelo (**Figura 3.16**) se cargan los datos del avión seleccionado, entre los datos que se cargan está el Identificador del Avión, su Nivel de Vuelo (FL), su Velocidad de Vuelo en Nudos (kt), su ETO en formato “hh:mm” y la ruta que va a seguir en el Sector. Estos datos se cargan en sendos Campos de Edición de Texto que no pueden ser editados por el Usuario. Dentro del Panel de Ficha se tiene un Panel extra para la Ruta, y en este panel se crean Campos de Edición de Texto según la Ruta del Avión seleccionado, estos campos tampoco se pueden Editar, sin embargo, se crean también campos en blanco justo debajo de cada punto de la Ruta para apuntar la hora de paso por dicho punto. Esta es una de las funciones que tienen los Controladores y consiste en apuntar la hora por la que el Avión pasa por cada punto de la Ruta para poder hacer un seguimiento detallado del Vuelo.

Justo a la derecha del Panel de Ficha de Progresión de Vuelo se tienen los Controles de la Simulación (**Figura 3.17**), véase, un botón de Play y un botón de Pausa, así como dos Campos de Edición de Texto (que no se pueden Editar por el Usuario) que representan el reloj de la Simulación y la hora de fin de la Simulación, puestos como horas en formato “hh:mm:ss”. Los botones son botones de estado, igual que los del Panel de Raíl de Fichas, y cuando uno está activo se desactiva el otro (de hecho incluso se deshabilita, de modo que solo se puede interactuar con el que está desactivado). El botón de Play inicia la Simulación desde la hora inicial o, en el caso de que la Simulación se haya pausado previamente, continúa desde ese punto en el tiempo de Simulación. El botón de Pausa, pausa la Simulación, la cual se detiene automáticamente cuando se llega a la hora de fin igualmente.

(Imagen Controles de Simulación)

Figura 3.16 Ficha de Progresión de Vuelo (1) Vacía (2) Rellenada.

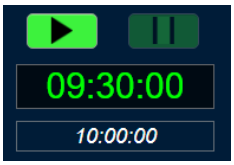

Figura 3.17 Controles de la Simulación.

El tiempo de Simulación se ha establecido que avance en intervalos de 10 segundos, debido a que la velocidad media en Nudos de una aeronave comercial en crucero está entre los 300 y los 500 kt, esto significa que un Avión comercial avanza entre 0.83 y 1.38 nm cada 10 s, lo cual, teniendo en cuenta que las distancias mínimas horizontales se consideran en 5 nm es una buena tasa de muestreo. También ayuda a que la Simulación avance a un ritmo similar al tiempo real, pues a cada muestreo se tienen que hacer muchos cálculos de Simulación, además de comprobaciones de Comunicaciones.

A la derecha de los Controles tenemos el Panel de Comunicaciones del Controlador (**Figura 3.18**). Este es el medio por el cual el Controlador puede influir en el Tráfico, dando instrucciones a los pilotos para autorizar ciertas maniobras o corregir errores de los pilotos. Debido a la complejidad que pueden alcanzar estas instrucciones si se da libertad absoluta, y teniendo en cuenta que en la Edición del Escenario se establecen Incidentes para que ocurran durante la Simulación, se ha limitado la Comunicación con botones que cargan mensajes pre-hechos con campos variables que se rellenan automáticamente según ciertos parámetros. Estos parámetros varían según el Avión, de modo que se tiene un desplegable que permite seleccionar el Avión al que se quiere mandar el mensaje concreto, y en base a este Avión y sus datos de Simulación se rellenan los parámetros variables de los mensajes. Cuando se selecciona uno de los mensajes, se muestra en un Campo de Edición de Texto que no se puede Editar directamente por el Usuario, para poder comprobar que es el mensaje que se quiere mandar, y en caso afirmativo se tiene un botón de Enviar, que pasa el mensaje a la Zona de Comunicaciones.

Figura 3.18 Panel de Comunicaciones del Controlador.

La Zona o Panel de Comunicaciones (**Figura 3.19**) es un objeto tipo Área de Texto (del desarrollador de aplicaciones de MATLAB®) que permite escribir varios grupos de texto de seguido. En este Panel aparecerán mensajes con solicitudes de los Pilotos a los que el Controlador debe Responder, así como las respuestas de dichos Pilotos a las instrucciones que da el Controlador. Estos mensajes también están pre-hechos y disponen también de campos variables que rellenar con los datos de Vuelo de cada Avión. Cada mensaje está precedido con un identificador del que ha mandado dicho mensaje, en caso de los Aviones su Identificador y en caso del Controlador dice “Control”.

Los mensajes pre-hechos se muestran a continuación, tanto los del Controlador como los del Piloto, para referencia: Callsign se refiere al Identificador del Avión involucrado, WPTrigger refiere al Punto de Paso

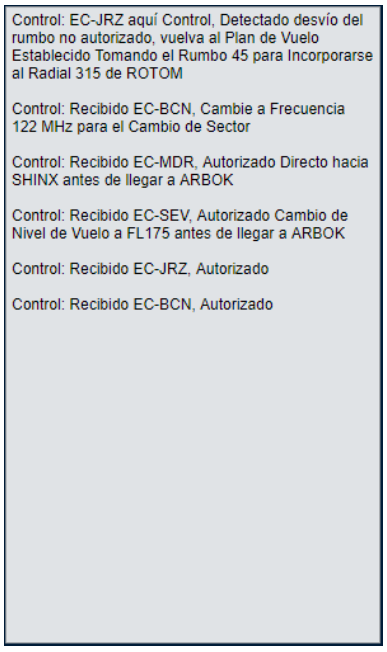

Figura 3.19 Zona o Panel de Comunicacionesr.

"Trigger" o Detonante de la Incidencia asociada al Avión en cuestión (si no hay incidencia queda vacío), FLObjetivo y WPObjetivo hacen referencia a los objetivos de las Incidencias que conllevan una solicitud de Autorización (Cambio de Nivel de Vuelo y Solicitud de Directo), Time refiere al tiempo de activación de la incidencia (antes o después del Trigger), Freq refiere a la Frecuencia del Sector fronterizo con el Punto de Paso final de la Ruta del Avión en cuestión y por último RADAutorizado refiere al Radial del Punto de Paso Trigger que se autoriza a tomar a una aeronave que tenga la Incidencia tipo 3 (Cambio de Rumbo no Autorizado) y el HDGAutorizado es el Rumbo que debe tomar la aeronave en cuestión para retornar a la Aerovía correspondiente a dicho Radial. Este último se calcula automáticamente en base a la ruta del Avión en cuestión. Cada uno de los mensajes del Control está asociado a su botón correspondiente en el Panel de Comunicaciones del Controlador.

Líneas de Comunicación:

Aviones:

1. Control aquí Callsign solicitamos autorización para realizar el Plan de Vuelo establecido en el Sector
2. Control aquí Callsign solicitamos autorización para realizar un Cambio de Nivel de Vuelo a FLObjetivo Time de llegar a WPTrigger
3. Control aquí Callsign solicitamos autorización para realizar un Directo hacia WPObjetivo Time de llegar a WPTrigger
4. Recibido Control seguiremos instrucciones
5. Control aquí Callsign solicitamos Frecuencia de Radio para el Cambio de Sector

Control:

1. Recibido Callsign, Autorizado
2. Recibido Callsign, Autorizado Cambio de Nivel de Vuelo a FLObjetivo Time de llegar a WPTrigger
3. Negativo Callsign, Manténgase en el Nivel de Vuelo Autorizado
4. Recibido Callsign, Autorizado Directo hacia WPObjetivo Time de llegar a WPTrigger
5. Negativo Callsign, Siga con el Plan de Vuelo Establecido
6. Callsign aquí Control Detectado desvío en el rumbo, Vuelva al Plan de Vuelo establecido tomando el Rumbo HDGAutorizado para incorporarse al radial RADAutorizado del WPTrigger

7. Recibido Callsign, Cambie a frecuencia Freq para el Cambio de Sector

Por último, el botón asociado al mensaje que da la frecuencia del Sector Adyacente sirve para activar la salida del Avión, véase, cuando este Responde al mensaje (con un "Recibido", mensaje 4 de Aviones) el avión deja de representarse en los Ejes.

Por último, pero no por ello menos importante, el objeto de Ejes, donde se representa el Sector y los aviones cuando están dentro de él. Es en este objeto donde se observa el funcionamiento principal de la Simulación, los aviones se representan con un triangulito verde de 0.5 nm de altura y 0.4 nm de base, orientado según el rumbo de la aeronave. Adicionalmente se representa el "vector" de velocidad, como una línea recta de color magenta que acaba en una cruz blanca, que sigue la orientación del rumbo del Avión y tiene una longitud equivalente a la velocidad en nm/minuto. También se representan las posiciones que ha tomado la Aeronave en los últimos 50 segundos, marcado como una línea de puntos que sigue a la aeronave.

Como último añadido cabe mencionar una lámpara roja que se tiene en lo Alto del Panel de Raíl de Fichas, fuera de este y entre este y los ejes, esta lámpara sirve como aviso general de que se ha detectado una alerta por conflicto, se ha creado para la posibilidad de que se tengan tantos aviones que no se puedan ver todos los botones del Raíl a la vez (El Panel tiene habilitada la propiedad "Scrollable", que permite tener objetos dentro de él que queden fuera de sus dimensiones, creando barras de movimiento dentro del Panel), de modo que si salta una alerta pero no está visible el botón de la aeronave en conflicto, siempre se puede saber si hay una aeronave en alerta gracias a esta lámpara de alerta general. Además de esto, se tiene el botón de Salir de la Simulación, el cual limpia la Variable del Escenario para Simulación y devuelve a la Pantalla de Selección de Escenario del Simulador. Este botón de Salir queda inhabilitado durante la Simulación, véase, la Simulación debe estar pausada o terminada para poder Salir.

Una vez explicados todos los elementos, solo queda explicar cómo Funciona la Simulación, pues la Simulación empieza con el Reloj de Simulación igual a la Hora de Inicio y va avanzando con una tasa de muestreo de 10 segundos, como se ha explicado antes, para ello se crea un objeto Timer (o Temporizador) de MATLAB® , que sigue al reloj del ordenador y ejecuta Funciones periódicamente hasta que se le diga que pare. Una de las ventajas que tiene el usar el Timer es que tiene una propiedad de Interrumpibilidad, que se puede configurar a modo "queue" o "poner en cola", lo que hace esto es que si una función o interacción del Usuario fuera a interrumpir una Función en ejecución, en vez de Interrumpir se pone en una cola de ejecución que se va completando a medida que se completan los comandos en orden de interrupción o llamada.

Entre las funciones que ejecuta el Temporizador se tienen: una que efectúa la Navegación, una que detecta si ha llegado la ETO de un Avión (calcula si queda menos de un minuto y en ese caso activa la representación del Avión) y por último una que determina si los Aviones efectúan comunicaciones y las manda al Panel de Comunicaciones.

La función de Navegación hace que la aeronave siga la ruta establecida, haciendo virajes en los puntos en los que debe y navegando en Aerovías de 10 nm de ancho centradas en el Radial que conecta un Punto de Paso con el Siguiente. Se ha elegido una anchura de 10 nm para las Aerovías debido a que la distancia mínima lateral establecida por los protocolos de OACI es 5nm [1][4], de modo que con una Aerovía de este tamaño se pueden cumplir dichos estándares. El funcionamiento específico del código que ejecuta esta función se explicará más adelante en un apartado dedicado.

3.4 Análisis y explicación del código en Detalle

En esta sección se va a analizar y explicar en detalle el funcionamiento específico del código creado, que se ha incluido en el Apéndice A. Para ello se va a utilizar terminología explícita de MATLAB® y, en concreto, terminología específica de su Desarrollador de aplicaciones. Se pueden encontrar definiciones de dicha terminología en el Glosario de Términos al principio del Documento.

El código desarrollado se divide en tres secciones: una primera de Propiedades (properties), que consisten en ciertas variables que se almacenan en la variable de la Aplicación (app) y se pueden llamar desde cualquier función de la misma escribiendo 'app.' seguido del nombre de la Propiedad, la segunda sección contiene las Funciones (functions) creadas dentro de la Aplicación y que deben tomar como primer argumento de entrada la variable de la Aplicación (app) aunque no se use en su código (en esos casos puede ser sustituida por el símbolo ' '), estas son fragmentos de código que puede ser repetitivo y por ello se escriben a parte adjudicándoles un comando de llamada, que puede ser usado por cualquier otra función de la Aplicación, por último está la sección de Callbacks o Funciones de Respuesta, estas son funciones que se activan por acciones del Usuario (Pulsar un botón, editar un Campo de Texto, etc…), estas están asociadas a objetos creados en la Aplicación y sus argumentos de entrada son fijos, no manipulables y son siempre las mismas dos variables,

la variable de la Aplicación (app) y una variable event, que se genera automáticamente y guarda información sobre el evento que ha causado la activación del Callback (Tipo de evento, Objeto que lo ha causado, Valor que éste ha tomado y Valor previo al evento).

Adicionalmente, en el **Código A.1** del **Anexo A**, se puede ver que hay una cuarta sección, llamada Create Components, la cual define los Objetos que conforman la Aplicación, los cuales no se van a desarrollar pues son objetos de MATLAB® cuyas propiedades y funcionamiento se puede buscar en las ayudas de MATLAB® .

Propiedades

Esta sección comienza declarando las Propiedades de la Aplicación asociadas a Componentes de la misma, véase los Objetos que se han creado en la Aplicación. Estas propiedades se crean automáticamente por el desarrollador de aplicaciones de MATLAB® y no son editables por el programador. Estos objetos se describirán en más detalle en la última sección, donde se crean estos Objetos. Justo después de esta sección se tienen las Propiedades que se crean por el programador, las cuales se van a describir a continuación.

SectorEditor, esta propiedad se ha creado para almacenar la variable estructura 'Sector', que se almacena en los archivos .mat de cada escenario, cuando se carga o crea un Escenario en el Editor de Escenarios. De esta manera se pueden manipular y editar los distintos parámetros del Escenario y luego almacenarlos en el archivo .mat correspondiente.

SectorSimulador, esta propiedad tiene una función similar a la de SectorEditor, pero para cuando se carga y Simula un Escenario en el Simulador. De esta manera se puede acceder a toda la información del Sector desde otras funciones durante la Simulación.

S, esta propiedad se usa para Editar la lista de escenarios, de manera que se puedan editar el número y los nombres de los escenarios y guardarlos en el archivo escenarioslist.mat.

NombreNuevoEscenario, esta propiedad se usa para almacenar el valor del Campo de Edición de Texto del Nombre del Escenario, ya sea para Crear un Escenario o Renombrarlo.

pmat y **libapp**, estas dos propiedades se han creado para almacenar las cadenas de caracteres '.mat' y 'libapp\', de manera que al guardar variables en archivos .mat se usan para crear la cadena de caracteres que referencia al archivo de un Sector concreto para poder usarlo en el comando save y load, que sirven para guardar y cargar archivos .mat respectivamente.

SectorEditorSelec y **SectorSimuladorSelec**, estas propiedades se han creado para almacenar los Nombres de los Escenarios Seleccionados en los Objetos tipo Árbol de las Pantallas de Selección de Escenarios. De esta manera se pueden referenciar desde cualquier función (por ejemplo la de Cargar Escenario) y usar el Escenario concreto.

Vert, esta propiedad se ha creado para almacenar los datos correspondientes a un tramo de frontera tipo Vértice en el Editor, tanto para la Frontera del Sector como para las respectivas Fronteras de las Zonas Restringidas. Esta se usa para almacenar datos provisionales que quedan sin guardar en la variable Sector hasta que se pulsa el respectivo botón Guardar del panel de Vértices. En concreto permite guardar las coordenadas de dos puntos, que formarían la línea recta del tramo de la Frontera que se quiere Crear o Editar. Estas coordenadas se guardan en el respectivo campo de la variable Sector (Border para la Frontera y RestricZone para las Zonas Restringidas), en sendos campos dentro de estos que contienen variables tipo celda (X e Y). Las distancias se toman en millas náuticas (nm).

Arch, esta propiedad tiene la misma función que la anterior, pero para tramos de Frontera tipo Arco, véase, curvas. Esta propiedad almacena las coordenadas de los puntos de inicio y final del arco, así como el Radio de dicho Arco y dos variables que sirven para determinar exactamente el tramo del Arco entre esos dos puntos con ese Radio. En concreto se determina si el arco es Cóncavo o Convexo (se considera cóncavo si el Arco se extiende por encima o a la derecha de la cuerda del Arco, véase la línea recta entre los dos puntos) además de si es el Arco Largo o el Arco Corto que pasa por esos dos Puntos y con ese Sentido o Radio, en la **Figura 3.20** a continuación se ve claramente estos cuatro Arcos posibles para dos puntos con un Radio concreto.

Alt, esta propiedad tiene una función similar a las dos anteriores pues sirve para almacenar la información sobre los límites verticales del Sector o las Zonas Restringidas que estén seleccionados. Estos límites se guardan en pies (ft), y se convierten a Nivel de Vuelo cuando corresponde a un Nivel de Vuelo válido (el umbral se considera en 10000 ft).

Airport, esta propiedad se ha creado para almacenar los datos de un Aeropuerto, ya sea uno que está por guardar o uno guardado que se ha seleccionado en su respectivo objeto tipo Árbol. Esta variable tipo Estructura (la mayoría de estas propiedades lo son, quitando algunas que se indicarán más adelante y aquellas que almacenan solo cadenas de caracteres de las que se han mencionado ya) almacena en sus diferentes campos el Nombre y Posición del Aeropuerto, así como las Pistas de Despegue y Aterrizaje si se incluyen.

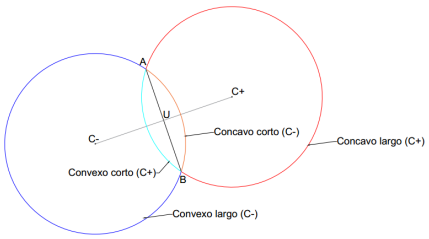

Figura 3.20 Curvaturas y Longitudes para un arco que pasa por 2 puntos con un Radio.

Waypoint, similar a la anterior esta propiedad se usa para almacenar la información relevante de un Punto de Paso, véase, su Nombre, Posición y la Frecuencia de Control del Sector con el que hace Frontera, si es ese el caso.

Airplane, esta propiedad sirve para almacenar la información sobre un avión en forma de variable Estructura. Esta variable almacena datos tales como la Matrícula de la Aeronave (que actúa como su identificador de Radio), el Nivel de Vuelo en el que entra al Sector, la Velocidad de Vuelo en Nudos (kt), su ETO u Hora Estimada de Entrada al Sector, la Ruta que va a seguir y, por último, una sección de Incidencias, en la que se puede programar una Incidencia de entre tres tipos Concretos, que se explicará más en detalle un poco más adelante. Al pulsar el botón de Guardar en el Panel de Aviones se almacena esta información en su respectivo Campo de la variable Sector.

RouteWP, esta propiedad sirve para complementar el llenado de la anterior, pues se usa para almacenar el nombre de un Punto de Paso/Aeropuerto del Sector que se pretende añadir a la Ruta del Avión Seleccionado, al pulsar el botón Guardar en su respectivo panel, se añade dicho Nombre al Campo de Ruta de la variable Airplane.

Incident, en esta propiedad se almacena la información relativa al Incidente programado para el Avión Seleccionado. Esta información consta del Tipo de Incidente, que puede ser uno de tres: Cambio de Nivel de Vuelo (Tipo 1), Solicitud de Directo (Tipo 2) o un Desvío no Programado (Tipo 3), o ninguno de ellos (Tipo 0). También se almacenan el Punto de Paso Detonante de la Incidencia (véase, el punto entorno al cual ocurre, ya sea antes o después), el Objetivo de la Incidencia, la cual cambia según el Tipo de Incidencia (Nivel de Vuelo Objetivo, Punto de Paso Objetivo del Directo o Rumbo Objetivo), y por último el Tiempo de "activación" de la Incidencia (antes o después de llegar al Punto Detonante). Al Pulsar el botón Guardar del Panel de Incidencias se almacena esta variable en el campo correspondiente de la variable Airplane.

TimeSector, terminando con las propiedades relacionadas con el Editor de Escenarios, esta propiedad se usa para guardar las Horas de Inicio y Final de la Simulación, antes de guardarse en la variable Sector al pulsar el respectivo botón de Guardar.

ATCCommLine, esta es una variable tipo Celda, que se usa para almacenar las partes no variables de los 7 mensajes posibles que puede mandar el Controlador en el Simulador, por esta razón se usa una variable tipo Celda, pues esta permite crear matrices cuyos elementos tienen dimensiones no concordantes, que darían error en MATLAB® si se tratara de una variable tipo double y estos mensajes varían bastante en la cantidad de secciones Editables, así como en longitud de caracteres de cada sección NO Editable.

AirmenCommLine, esta propiedad se usa para lo mismo que la anterior descrita, pero para los 5 mensajes posibles que pueden mandar los Pilotos. La manera en la que se han organizado ambas matrices de celdas es la siguiente: Se tiene una columna de celdas para representar el número de mensajes (dimensión 5x1 en la de mensajes de Pilotos y dimensión 7x1 en la de mensajes de Controladores), cada fila de estas matrices de Celdas contiene una Celda de dimensión 1xn, donde n es el número de secciones NO Editables del mensaje en concreto, por último, dentro de cada espacio de esa Celda Fila, cada sección del mensaje se guarda en una Celda en sí misma, de manera que cada sección de mensaje puede tener la longitud que sea sin afectar a la coherencia de dimensiones de la variable asociada a la Propiedad. De este modo, para llamar a una sección concreta de un mensaje, se llama de la siguiente manera: Nombre de la Propiedadnúmero del mensajenúmero de la sección del mensaje1.

AirplaneSim, esta propiedad se usa para almacenar los datos de Todos los aviones en ella y manipularlos durante la Simulación para que representen las actuaciones de las aeronaves, en esencia es una variable estructura con dimensión igual al número de Aviones y Campos iguales a la Propiedad Airplane descrita anteriormente. Adicionalmente, a AirplaneSim se le ha añadido un Campo llamado Z, en el que se guarda la Altitud en pies (ft), pues en la variable Airplane solo se guarda el Nivel de Vuelo.

TimeSim, esta Propiedad es de Tipo Duration, un tipo de variable que sirve para almacenar tiempo. Esta propiedad se usa entonces para almacenar el Tiempo de Simulación, que avanzará en intervalos de 10 segundos, y usando una variable Duration es muy sencillo seguir datos de Tiempo.

FlightStrip, esta propiedad se rellena con los datos de las Aeronaves que se rellenarían en una Ficha de Progresión de Vuelo, en este caso son: La Matrícula (que actúa como identificador de Radio de la Aeronave), Nivel de Vuelo, Velocidad en Nudos (kt), ETO y la Ruta. Adicionalmente, se le ha añadido un Campo donde almacenar las Horas de Paso por cada Punto de la Ruta, pues es una de las Funciones que se le ha puesto al Controlador del Simulador. Estos datos se representan luego en el Panel de Ficha de Progresión de Vuelo del Avión.

Comminput, esta propiedad es una variable de tipo estructura que se usa para montar los mensajes de texto de las Comunicaciones del Controlador, para luego poder introducirlas en el Área de Texto donde se ve el mensaje montado antes de ser mandado al Panel de Comunicaciones. Los dos Campos que tiene esta variable son un primer campo llamado Data, donde se almacenan los datos variables de los mensajes de la Aeronave seleccionada en el Panel de Comunicaciones del Controlador, el segundo campo se llama msg, y es donde se monta el mensaje a mandar por el Controlador, usando los datos almacenados en el campo Data y las secciones correspondientes de la propiedad ATCCommLine.

SimTimer, esta es una propiedad que se crea vacía, pues se usa para crear en ella el objeto Timer o Temporizador que se usa para llevar el tiempo real de la simulación, estos objetos siguen al reloj del equipo y ejecutan funciones, que se pueden definir, en tiempos y periodos determinados, que también se pueden definir, y un número de veces que también se puede definir. Cuando se pausa o se termina la Simulación, esta variable se vuelve a declarar como una Variable Vacía.

SimData, llegando a la última Propiedad creada, esta se usa para almacenar varios Datos de Simulación sobre los distintos Aviones que no tenían cabida en otras Propiedades y el acceso a los cuales es necesario en varias funciones distintas. Entre estos Campos que se crean al cargar un Escenario en la Simulación se tienen: Autoriz, un campo tipo estructura que consta de los campos Type y Value, que sirve para almacenar la información sobre Autorizaciones ya sea su tipo o si se han dado o no (Value es un campo de variables lógicas, cuyos valores deben ser o 1 lógico o verdadero o 0 lógico o falso, al declararlo se deja en falso); ETOCheck, una variable que puede valer 0, 1 o 2, representando si el Avión está aún por entrar en el Sector (0), dentro del Sector (1) o ya ha salido del Sector (2), con esta variable se pueden ignorar a los Aviones que no estén dentro del Sector y de ese modo acelerar los cálculos de Simulación; Turning, este campo se usa para determinar si la Aeronave está girando o no, sirve para poder representar mejor la Navegación, puede tener valor 0 (no está girando), 1 (está girando hacia el siguiente Punto de Paso de la Ruta) o 2 (está girando hacia un rumbo determinado por una Incidencia de Tipo 3 o de Desvío no Programado); Comms, es una variable que puede tomar valores entre 0 y 6, que se usa para determinar qué mensaje manda el Piloto correspondiente (1 a 5), si no ha mandado mensaje y está a la espera de mandar uno (0) o si ya ha mandado un mensaje y está esperando respuesta del Controlador (6); Tracks, este es un campo que se usa para almacenar los últimos 5 puntos por los que ha pasado cada Avión y poder representarlos como una línea de puntos blancos que siguen al Avión correspondiente; RouteTrack, este campo se usa para determinar la posición del siguiente Punto de la Ruta dentro de la lista de la Ruta; Climbing, este campo se usa para determinar si el Avión está ascendiendo (1), descendiendo (-1) o manteniendo su altitud (0); por último, DoneIncident, es otro campo en el que se almacenan variables lógicas y se usa para determinar si el Incidente programado en la Aeronave ha sucedido o no, de manera que si ya ha sucedido, no se hacen los cálculos con respecto al Incidente de esa Aeronave y se aceleran los cálculos de Simulación.

Funciones

plotEditorgraph(app), esta función no devuelve ningún valor tras su llamada y toma como argumento de entrada únicamente la variable de la Aplicación (app). Por lo general, a no ser que se especifique, la mayoría de las funciones creadas en esta Aplicación no devuelven ningún valor tras su llamada. Esta función en concreto se usa para representar en el Objeto Ejes del Editor de Escenarios los datos invariables del Sector (véase, su Frontera, la Frontera de las Zonas Restringidas, los Puntos de Paso y los Aeropuertos). Esta función reconoce si dichos campos de la variable Sector están vacíos y los ignora en ese caso (con el comando isempty). Al representarlos, se usa el siguiente esquema de colores: línea verde para la Frontera del Sector, línea roja para las Fronteras de las Zonas Restringidas, rombos cian con el nombre en cian encima del rombo para los Aeropuertos y triángulos cian con el nombre en cian encima del triángulo para los Puntos de Paso. Los puntos se representan usando el comando plot, y los nombres usando el comando text. Adicionalmente, al final de la función, se especifica la propiedad del objeto Ejes DataAspectRatio con el vector [1 1 1], de modo que la representación no se deforma. También, aparte de esto último, se representa un cuadrado cuyas

dimensiones hacen que encuadre la Frontera del Sector, con un esquema de colores con el cual solo se representan los vértices como puntos negros.

resetBorderXYeditfieldvert(app), con esta función se declaran en 0 los Campos Numéricos Editables de las coordenadas de los puntos de un tramo de Frontera tipo Vértice en el Panel de vértice dentro del Panel de Frontera del Sector. Con esto se restaura a valores de inicio estos objetos y se pueden vaciar los datos de la propiedad Vert en la siguiente función.

updateBorderVert(app), con esta función se pasan los datos de los Campos Numéricos Editables de las coordenadas de los puntos de un tramo de Frontera tipo Vértice en el Panel de Vértice del Panel de Frontera del Sector, a la propiedad Vert, que tiene un campo para el punto de origen del tramo y uno para el punto de destino del tramo. Si estos Campos Numéricos están en 0, se considera restaurada a valores de inicio a la propiedad Vert.

LoadScenario(app), esta función se usa para cargar los datos del Escenario en los distintos objetos del Editor de Escenarios, representando el Sector con la función mencionada antes, rellenando los objetos tipo árbol correspondientes de cada Panel con los Datos del Sector y rellenando los Campos de límites verticales (si estuvieran definidos) y de Horarios de Simulación. Por último, también limpia cualquier selección que haya en cualquier objeto Árbol de la parte del Editor de Escenarios.

CharNum = dec2char(app,Num), esta función si devuelve un resultado, siendo este la cadena de caracteres correspondiente al número que se introduce com el argumento de entrada Num. Esta función permite obtener la cadena de caracteres asociada a un número entero decimal de hasta 4 dígitos, pues no se espera necesitar expresar números mayores en campos de Texto. Esta función se ha visto como necesaria debido a que no hay un comando de MATLAB® que convierta a un número decimal en una cadena de caracteres que contenga a dicho número decimal, existe el comando char, pero este transforma un vector de números en una cadena de caracteres cuyos códigos ASCII se corresponden con los del vector de números.

clearSector(app), está función vacía la variable Sector dentro de la propiedad SectorEditor, se pueden ver los distintos campos que se tienen: Nombre (Name) es una variable tipo celda que almacena el nombre del Sector, esta función la declara como Celda Vacía; Tiempo (Time), una variable tipo estructura que almacena la hora de inicio y la hora final de la Simulación como variables Duration, esta función la declara como Matriz Vacía; Frontera (Border), es una variable estructura que guarda las coordenadas de los puntos de la Frontera, así como los límites verticales, esta función la declara como Matriz Vacía; Zonas Restringidas (RestricZones), es una variable tipo estructura de longitud el número de Zonas Restringidas, y que cuenta con los mismos campos que la de Frontera, esta función la declara como Matriz Vacía; Aviones (Airplane), es una variable tipo estructura de longitud el número de Aviones, y que cuenta con los mismos campos que la Propiedad Airplane, esta función la declara como Matriz Vacía; Aeropuertos (Airport), es una variable tipo estructura de longitud el número de Aeropuertos, y que cuenta con los mismos campos que la Propiedad Airport, esta función la declara como Matriz Vacía; y por último, Puntos de Paso (Waypoint), es una variable tipo estructura de longitud el número de Puntos de Paso, y que cuenta con los mismos campos que la Propiedad Waypoint, esta función la declara como Matriz Vacía.

clearSelectedNodesEditor(app), esta función limpia las selecciones en todos los objetos tipo Árbol del Editor de Escenarios, dejando sus respectivas propiedades SelectedNodes como Matrices Vacías y ejecutando sus respectivas funciones de respuesta.

clearTreesEditor(app), esta función elimina todos los nodos de los objetos Árbol de los Paneles de Edición de los parámetros del Sector, usando la función delete. Esta función excluye el objeto Árbol TreeSector, pues sus nodos son predefinidos en la Aplicación, y son los que sirven para seleccionar el ámbito del Sector que se quiere Editar.

clearButtonGroupsEditor(app), esta función devuelve a los objetos tipo Button Group, que se usan para determinar los Tipos de Frontera en los Paneles de Frontera y Zonas Restringidas, a sus valores iniciales, véase, al estado “Selecciona Una”. También ejecuta sus respectivas Funciones de Respuesta.

I = encuentraI(app,Arco,A,B), esta función sirve para encontrar el punto medio de un Arco ya creado, el cual se ha decidido llamar I, de manera que se pueda usar para calcular los parámetros del Arco. Los argumentos de entrada son la variable de la Aplicación, una variable que contenga las Coordenadas de todos los puntos del Arco, y dos variables, A y B, que contienen las coordenadas de los puntos que forman la cuerda del Arco. La manera en que se calcula este punto es la siguiente: primero se generan los vectores AB y su perpendicular (se usa el método de intercambiar sus coordenadas y cambiarle el signo a una de ellas), así como el punto medio de la cuerda, que se ha llamado U, a continuación se crea una variable que contenga los puntos de una recta que pase por U y tenga como vector director el vector perpendicular a AB (esta recta sería la que pasa por el centro del Arco y por los puntos U e I). Una vez obtenida esta recta, se obtienen los puntos más centrales (dentro de sus dimensiones) de la variable Arco y se almacenan en una

variable nueva llamada arco (con minúscula, MATLAB® distingue entre mayúsculas y minúsculas), de esta forma se reducen los cálculos que hay que hacer, pues el punto I es más probable que esté entorno a la mitad de la dimensión de la variable Arco. A continuación, se inicia un bucle for que barra todos los puntos de la variable arco, y a cada iteración se calcula la distancia de ese punto con todos los puntos de la recta (que se almacenó en una variable llamada H), y se almacena esas distancias en una variable que se ha llamado d. Con esta variable, se calcula cuales de estas distancias son inferiores a 0.01 nm, usando el comando find, y se almacenan en una variable tipo Celda llamada P las distancias, los índices de la variable distancia (obtenidos del comando find) y el índice de la variable arco correspondiente al punto estudiado. Una vez barrida toda la variable arco, se convierte a la variable P en una Matriz, usando el comando cell2mat, y luego se calcula la distancia mínima de entre las calculadas, usando el comando min con la primera columna de P (que contiene las distancias), para obtener el índice de la fila de P que la contiene y, de ese modo, poder obtener el índice del punto de mínima distancia de arco, y declarar dicho punto como el punto I.

Centro = encuentracentro(app,A,B,I,R), esta función sirve para encontrar el centro de un Arco ya creado, se toman como argumentos de entrada la variable de la Aplicación, los puntos A, B (extremos del Arco) e I (que se obtiene con la función descrita anteriormente) y el Radio del Arco (que se calcula con otra función). El cálculo de la posición del Centro con estos datos es muy simple, se calcula el punto U, intermedio de la cuerda, y con ese punto se calcula el vector IU (que va de I a U). Con IU calculado, la posición del Centro se obtiene aplicando un vector paralelo a IU, con módulo igual al Radio del Arco, al punto I.

R = Calcrad(app,X,Y), está función sirve para calcular el Radio del Arco, toma como argumentos de entrada las variables X e Y que almacenan sendas coordenadas de los puntos del Arco. En este cálculo primero se calculan los puntos A y B (que son los extremos del Arco) fácilmente obtenibles pues son el primer y último punto del Arco respectivamente, seguidos del punto U (el punto medio de la cuerda), y a continuación se almacenan X e Y en una variable Arco, compatible con la función encuentraI, para luego hacer una llamada a esta función y obtener el punto I. Una vez obtenido I se tienen los suficientes Datos del diagrama de la **Figura 3.21**, que nos sirven para calcular el Radio usando las **Fórmulas 3.1** a **3.3**. La primera fórmula sirve para calcular el ángulo alfa de la **Figura 3.21**, para lo cual se tienen todos los datos suficientes, véase C (la distancia entre A e I) y CT (la distancia entre U e I) en la **Figura 3.21**. Calculado alfa se tienen todos los datos para calcular gamma con la segunda fórmula, que lo calcula en radianes, esta fórmula se basa en la propiedad de un triángulo de que todos sus ángulos suman 180º o 2pi radianes, para el caso de un triángulo isósceles, como es el caso pues dos de los lados deben medir R. Una vez obtenido gamma se puede aplicar el Teorema del coseno para un triángulo isósceles (**Fórmula 3.3**), y despejando el Radio se obtiene la cuarta fórmula, para la cual se tienen todos los Datos. Este razonamiento paso a paso se aplica en la función y resulta en la obtención del Radio del Arco.

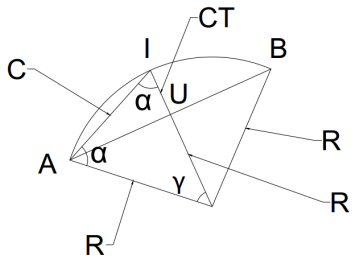

Figura 3.21 Diagrama del Arco para Calcular R.

$$\alpha = \arccos\left(\frac{CT}{C}\right), \tag{3.1}$$

$$\gamma = 2\pi - 2\alpha(rad), \tag{3.2}$$

$$C^2 = 2R^2 - 2R^2\cos\gamma,$$

$$R = \frac{C}{\sqrt{2(1-\cos\gamma)}}, \tag{3.3}$$

[Value,len] = ConcaveConvexShortLongCheck(app,X,Y,R), esta función sirve para determinar los parámetros de curvatura (Cóncavo o Convexo) y longitud (Largo o Corto) del Arco en base a los criterios que

se establecieron al describir la Propiedad Arch. Para ello se toman como argumentos de entrada X e Y, que contienen sendas coordenadas de los puntos del Arco y R, que es el Radio del Arco. Con estos argumentos se calculan los puntos A, B, I y U como en funciones anteriores. Usando las posiciones relativas entre I y U, y entre A y B, se determina si el Arco es Cóncavo (acento circumflejo) o Convexo (v), y se almacena en Value, y si es Largo o Corto, y se almacena en len.

Centro = Calccentre(app,Arco), esta función se usa para calcular la posición del Centro cuando se va a representar o guardar un Arco nuevo, por ello, la variable Arco que esta función tiene como argumento de entrada tiene un formato igual al de la Propiedad Arch. Para este cálculo se tiene un Sistema de Ecuaciones de Segundo Grado con dos Variables (coordenada x e y del Centro), que se puede ver en las **Fórmulas 3.4** a **3.6**. Teniendo como Datos las coordenadas de A (xa e ya) y B (xb e yb), así como el Radio y los parámetros de curvatura y longitud. Teniendo este Sistema de Ecuaciones se ha resuelto despejando la coordenada y del Centro como se puede ver en la segunda fórmula, y aplicándola al Sistema se obtiene la tercera fórmula, la cual es una ecuación de segundo grado con una sola variable. Cabe destacar que para simplificar la fórmula se han creado las constantes en la cuarta fórmula. Resolviendo la ecuación se obtienen dos soluciones para la coordenada x del Centro, lo cual deviene en tener dos soluciones para la coordenada y del Centro, teniendo entonces dos Centros posibles (C+ y C-). Se selecciona uno de estos centros en base a los criterios de curvatura y longitud descritos anteriormente y se devuelve dicha posición a la llamada de la función.

$$\begin{cases} (x_A - C_x)^2 + (y_A - C_y)^2 & = R^2 \\ (x_B - C_x)^2 + (y_B - C_y)^2 & = R^2 \end{cases} \tag{3.4}$$

$$C_y = \frac{AB - 2AB_x C_x}{2AB_y} \tag{3.5}$$

$$(4AB_y^2 + 4AB_x^2)C_x^2 + (8y_A AB_X AB_y - 8x_A AB_y^2 - 4AB_x AB)C_x + (AB^2 - 4y_A AB_y AB + 4AB_Y^2 S) = 0 \tag{3.6}$$

$$\text{Donde: } \begin{array}{rclcrcl} AB & = & x_A^2 - x_B^2 + y_A^2 - y_B^2 & ; & AB_y & = & y_A - y_B \\ AB_x & = & x_A - x_B & ; & S & = & x_A^2 + y_A^2 - R^2 \end{array}$$

[X,Y] = Calcarch(app,Arco,Centro), esta función se usa para calcular todos los puntos del Arco a partir de los datos de la variable Arch y el Centro calculado con la función descrita anteriormente. Esto se hace usando la parametrización polar de la ecuación de la circunferencia, y se calculan los ángulos polares de los puntos A y B (con una función que se describirá más adelante, llamada Calctheta) y seleccionando en base a los parámetros de curvatura y la posición relativa entre A y B el intervalo de ángulos polares que representar. Para referencia, se muestra la parametrización polar de la ecuación de la Circunferencia en la **Fórmula 3.7**.

$$C(\theta) = \begin{cases} x & = C_x + R\cos\theta \\ y & = C_y + R\sin\theta \end{cases} \tag{3.7}$$

theta = Calctheta(,P,C,R), con esta función se calcula el ángulo polar del punto P que está en una circunferencia con centro C y radio R. Este se obtiene usando la función acos para el cociente entre la diferencia de coordenadas x entre C y P, y dependiendo de si la diferencia entre coordenadas y de C y P es positiva o negativa, se almacena el ángulo obtenido o su opuesto (véase, 2pi – el ángulo, en radianes).

resetBordereditfieldarch(app), esta función se usa para lo mismo que la función resetBorderXYeditfield-vert, pero con los Campos de Edición del Panel de Arcos del Panel de la Frontera del Sector. Los datos numéricos de inicio siempre son 0 y los datos de curvatura y longitud de inicio se consideran Cóncavo y Corto respectivamente.

updateBorderArch(app), esta función sirve para lo mismo que la función updateBorderVert, pero para la Propiedad Arch y con los Campos de Edición del Panel de Arcos del Panel de la Frontera del Sector.

FLchar = FLChar(app,Altft) y **ftchar = ftChar(app,Altft)** ambas cumplen funciones similares, que consisten en convertir a cadenas de caracteres valores de Altitud, en formato de Nivel de Vuelo para el caso de FLChar y en formato de pies para el caso de ftChar. Dichos formatos consisten en dividir por 100 la Altitud en pies y añadirle 'FL' antes del número y un '0' si es menor de tres dígitos en el caso de FLChar y en simplemente añadir 'ft' después del número. Para ello se hacen sendas llamadas a la función dec2char.

resetBordereditfieldalt(app), esta función se usa para restaurar a los valores iniciales (0) a los Campos de Edición del Panel de Altitud del Panel de la Frontera del Sector. Adicionalmente, se restauran los valores de los objetos tipo Desplegable que se usan para determinar si el número introducido es en pies (ft) o en Nivel de Vuelo (FL), y cuyo valor de inicio es el asociado a Nivel de Vuelo.

updateBorderAlt(app), esta función almacena los datos del Panel de Altitud del Panel de la Frontera del Sector en la Propiedad Alt, si estos han sido restaurados a los valores iniciales, se considera restaurada a valores iniciales a la Propiedad Alt. El método que se usa es llamar a las Funciones de Respuesta de sendos Campos de Edición.

resetRZXYeditfieldvert(app), **updateRZVert(app)**, **resetRZeditfieldarch(app)**, **updateRZArch(app)**, **resetRZeditfieldalt(app)**, **updateRZAlt(app)**, estas funciones se usan del mismo modo que sus homólogas para la Frontera (estas tienen RZ en vez de Border) descritas anteriormente, pero para los respectivos Paneles dentro del Panel de Zonas Restringidas.

clearFLeditfields(app,param), esta función sirve para restaurar a valores iniciales los Campos de Edición de Texto que se usan para representar los límites verticales del Sector y de la Zona Restringida Seleccionada en su respectivo objeto Árbol. El valor inicial de estos Campos es el carácter espacio en blanco (‘ ’). El argumento de entrada param es un parámetro que se indica al llamar a esta función que consiste en una cadena de caracteres que puede tomar valor ‘RZ’ o ‘All’, de manera que si éste es ‘RZ’ solo se restauran los Campos correspondientes a las Zonas Restringidas, y si es ‘All’ se restauran todos.

resetAirporteditfield(app) y **updateAirport(app)**, estas funciones hacen lo mismo que sus homólogas anteriormente descritas, pero para los datos de Aeropuertos en el Panel de Aeropuertos y para la Propiedad Airport. Los valores de inicio para los datos del Aeropuerto son carácter espacio en blanco (‘ ‘) para el Nombre, y 0 para las coordenadas de su posición, para el Campo de Pistas el valor inicial es una Matriz Vacía.

resetRWYPanel(app), esta función sirve para actualizar los valores del Panel de Pistas de un Aeropuerto, si se tiene seleccionado un Aeropuerto cuando se llama a esta función, se añaden los datos sobre las pistas del Aeropuerto en el Panel, véase, se añaden a un objeto tipo desplegable de la Sección para Editar una pista, que permite seleccionar un par de pistas y editarlas. El valor inicial del interruptor que permite pasar de la sección para Añadir una Pista a la sección de Editar Pistas y viceversa es el que permite ver la sección de Añadir una nueva Pista y el valor inicial del Desplegable es el 0, que equivale a no haber seleccionado ninguna pista. Los Campos de Edición numéricos, que en este caso son rodantes (véase, contienen dos botones con flechas pequeñas que indican hacia arriba o hacia abajo y permiten aumentar o disminuir el número guardado en una unidad).

clearRWYspinners(app), esta función devuelve los rodantes a 0 y llama a sus respectivas Funciones de Respuesta.

resetWaypointeditfield(app) y **updateWaypoint(app)**, son funciones que, como sus homólogas, restauran a los valores iniciales los datos presentados de los Puntos de Paso, que son iguales que para los Aeropuertos, con la excepción de que en vez de Pistas, los Puntos de Paso pueden tener un Campo de Frecuencia (Freq), cuyo valor inicial es 0.

resetWPfreqPanel(app), esta función deja el valor del Campo de Edición para la Frecuencia de un Punto de Paso a 0 y llama a su Función de Respuesta.

h = isvalidcallsign(,value), esta función sirve para detectar si el valor que se tiene como argumento de entrada es un indicador de llamada de aeronave válido o no, según los criterios que se han establecido anteriormente en el documento, que consisten en considerar válido un identificador que concuerde con una matrícula de aeronave civil española (que cubre desde EC-AAA a EC-WZZ). Esta función devuelve un valor lógico verdadero si es válido o falso si no lo es, en la variable h. Para comprobar si es válido el identificador, se comprueba si los tres primeros caracteres son ‘EC-’ y si lo son, se comprueba si los siguientes caracteres entran dentro del intervalo válido.

h = isvalidETO(,value), esta función sirve para detectar si el valor que se tiene como argumento de entrada es una hora válida escrita en cadena de caracteres. Esta función se usa para esos Campos de Edición de Texto que almacenan horas y tiempos, para comprobar si se ha introducido una hora válida o no. Se considera hora válida aquella que está en el formato de hora 24h (de 00:00 a 23:59). Si el valor introducido contiene la cadena de caracteres equivalente a una hora entre 00:00 y 23:59, se devuelve un valor lógico verdadero, si no entra dentro de esos parámetros, se devuelve un valor lógico falso, a través de la variable h. Para comprobar esto, primero se comprueba si el carácter central es el de ‘:’, y luego si cada carácter de los demás cumple sus respectivas restricciones.

Num = char2dec(,char), esta función, como su propio nombre indica, hace el proceso inverso de dec2char, convierte una cadena de caracteres que representa a un número, y lo devuelve como el número en una variable tipo double.

h = iswp(app,wp), esta función sirve para determinar si una cadena de caracteres (wp) es un Punto de Paso o Aeropuerto del Sector. Para ello, primero se comprueba si todos sus caracteres son mayúsculas (pues así es como se guardan los nombres de los Puntos), para a continuación, si su longitud es de 5 caracteres, se compara con los nombres de los Puntos de Paso y Aeropuertos creados y almacenados en los objetos Árbol de los Paneles de Aeropuertos y de Puntos de Paso (usando el comando find), si encuentra una coincidencia, devuelve un valor lógico verdadero y rompe el bucle con el comando break, de manera que no busca innecesariamente y se agilizan los cálculos. Si recorre ambas variables y no encuentra coincidencia con el valor de la variable wp, devuelve un valor lógico falso.

pos = findwp(app,wp), esta función sirve para devolver las coordenadas de la posición de un Punto de Paso o Aeropuerto que se busca en la cadena de caracteres wp, que se tiene como argumento de entrada. El proceso que sigue esta función es igual al de la función iswp, solo que cuando encuentra una coincidencia devuelve las coordenadas de posición del punto.

clearIncident(app), está función restaura la Propiedad Incident a sus valores iniciales, que son: carácter '0' para el campo Type, carácter vacío '' para el campo Trigger, matriz vacía para el campo Target y carácter '1' para el campo Time.

updateAirplaneTab(app), esta función carga los datos guardados en la Propiedad Airplane en el Grupo de Pestañas dentro del Panel de Aviones. Para ello llama a varias funciones que se describirán a continuación.

loadRouteTree(app), esta función se usa para rellenar el objeto Árbol de la Pestaña de Ruta dentro del Panel de Aviones. Se rellena con la Ruta (si la hay) del Avión almacenado en la Propiedad Airplane.

updateIncidentTab(app), esta función rellena los datos de la Pestaña de Incidencias del Panel de Aviones con los datos almacenados en la Propiedad Incident. Esto incluye hacer visible el Panel correspondiente según el tipo de incidencia y rellenar sendos Campos de Edición en esos casos, así como seleccionar los valores que toman los respectivos objetos tipo Desplegable.

plotRoute(app,Route,hdg), esta función representa la Ruta introducida como argumento de entrada en el Objeto tipo Ejes del Editor de Escenarios como una línea magenta. A parte de la Ruta, también se le añade un punto a representar que representa el punto en a partir del cual la Aeronave tardaría 1 minuto en alcanzar el primer Punto de la Ruta, para lo cual se usa el Rumbo de Entrada, que se tiene como argumento de entrada de la función, y la velocidad, que se obtiene de la variable Airplane.

clearAirplaneTab(app), esta función se usa para restaurar a valores iniciales los campos del Grupo de Pestañas del Panel de Aviones. Para ello se llama a una función que se describirá a continuación, que se llama clearAirplane, y luego se llama a la función ya descrita llamada updateAirplaneTab(app).

clearAirplane(app), con esta función se restaura a valores iniciales a la Propiedad Airplane, los cuales son: una cadena de caracteres vacía '' para el Identificador de la aeronave (Callsign), 0 para el Nivel de Vuelo de entrada (FL), 0 para la Velocidad de Vuelo (FSPD), 0 para el Rumbo de Entrada (HDG), una variable Duración de valor 0 para la ETO, una Variable Celda con una cadena de caracteres vacía '' para la Ruta (Route), y para el campo de Incidencias (Incident) se usa la función ya descrita clearIncident y se almacena la Propiedad Incident restaurada.

plotSimuladorgraph(app), esta función hace exactamente lo mismo que su homóloga (descrita la primera de esta lista), pero en el Objeto de Ejes de la Pantalla del Simulador, y con la variable Sector guardada en la Propiedad SectorSimulador.

LoadScenarioSim(app), esta función hace algo similar a lo que hace su homóloga descrita anteriormente (LoadScenario), pero para la Pantalla de Simulación y con los datos almacenados en la Propiedad Sector-Simulador. Aunque el objetivo de la función es similar, el método es distinto, debido a la diferencia de composición a nivel de objetos entre la Pantalla del Simulador y la del Editor de Escenarios. En concreto, lo primero que hace esta función es representar al Sector en el Objeto Ejes del Simulador (que en adelante se llamará Pantalla de Radar, pues es lo que representa estar haciendo). A continuación carga los datos de Aviones en la Propiedad AirplaneSim, la cual se usará para rellenar los demás datos de los Aviones. Una vez cargada AirplaneSim, si esta no está vacía se llama a una función que se describirá más adelante llamado OrdenaAirplaneSim, y después de esto se empiezan a llenar los campos que quedan por rellenar sobre los Aviones. Primero se rellenan los Datos sobre el primer Avión almacenado en la variable (el correspondiente al índice 1) en el Botón del Panel de Raíles de Fichas, así como en la Propiedad FlightStrip. También es aquí donde se le añaden los campos X, Y y Z a la Propiedad AirplaneSim, dejando X e Y como matrices vacías y Z como la Altitud en pies (ft) de la primera Aeronave. Después de haber lidiado con la carga de la primera Aeronave, se pasa a las siguientes, con un bucle for, que abarca la longitud de la Propiedad AirplaneSim. En

este bucle for, antes de rellenar los datos de la Aeronave de la iteración, se crea un Botón del Panel de Raíles de Fichas igual al primero y justo por encima del botón más alto, además se guarda este objeto en la misma Propiedad de la Aplicación donde se guarda el Botón Original, también se crean y rellenan todos los demás objetos asociados al Botón original y se almacenan en sus respectivas Propiedades de la Aplicación. En la misma iteración también se rellenan los datos de la Propiedad Flightstrip en su índice correspondiente a la iteración, y también se añaden los respectivos datos sobre los campos X, Y y Z de la Propiedad AirplaneSim también en el índice correspondiente a la iteración. Una vez barrida la Propiedad AirplaneSim y rellenados esos Datos, se crean los Campos de la Propiedad SimData y se rellenan con sus valores iniciales de manera que tengan dimensiones acordes al número de Aviones. Por último se rellena la Propiedad TimeSim con el campo Time.init de la variable Sector, y se rellenan los Campos de Edición de Texto que muestran el Reloj de la Simulación y la Hora de fin de la Simulación, además de limpiar las Áreas de Texto del Simulador.

OrdenarAirplaneSim(app), esta función se usa para ordenar los Aviones almacenados en la Propiedad AirplaneSim según su Altitud de Entrada, de más bajo a más alto. De esta manera, al crear los Botones en el Panel de Railes de Fichas, se quedan Ordenados según este criterio.

clearFlightStrip(app), esta función se usa para limpiar el Panel de Ruta del Panel de Ficha de Progresión de Vuelo, este Panel tiene la propiedad Scrollable activada, y en él se crean un número variable de Campos de Edición de Texto de modo que se represente la Ruta del Avión cuyo Botón en el Raíl de Fichas está activado, así como los tiempos de paso por cada Punto. Esta función elimina esos Campos adicionales, dejando solo los originales, de manera que se pueden rellenar de nuevo al pulsar otro Botón del Panel de Raíles de Fichas.

clearSectorSim(app), esta función restaura a valores iniciales a la variable Sector de la Propiedad Sector-Simulador, del mismo modo exacto que lo hace la función clearSector con la Propiedad SectorEditor.

clearlamps(app), esta función desactiva los objetos tipo lámpara, dejándolas apagadas.

clearTimers(app), esta función declara la Propiedad Sim como matriz Duration con valor 0, y los Campos del Reloj de Simulación y Hora final de Simulación también se declaran en 0.

clearFLSR(app), esta función elimina con el comando delete todos los Botones del Panel de Rail de Fichas que no sean el original, junto con los objetos asociados a ellos, y restaura a los valores iniciales a los objetos asociados al primer botón. Estos valores iniciales son: Botón desactivado, los Campos de Texto se quedan con un carácter de espacio en blanco (‘ ‘) y la lámpara desactivada.

Rad = encuentraRadial(,postrig,postarg), esta función calcula el rumbo asociado al Radial del Punto postrig, que lo une con el Punto postarg.

wpi = locatewp(app,wp), esta función barre el campo de Puntos de Paso de la variable Sector de la Propiedad SectorSimulador, buscando una coincidencia con el Punto de Paso wp que se introduce como argumento de entrada. Al encontrar una coincidencia, devuelve el índice del campo de Puntos de Paso en el que se encuentra y rompe el bucle con el comando break.

ETOCheck(app), esta función es llamada durante las Funciones de Respuesta del objeto Timer o Temporizador que se usa para seguir el tiempo real de Simulación, la funcionalidad de esta función consiste en detectar si queda menos de 1 minuto para la ETO de cada avión que aún no ha entrado en el Sector, y en caso de que esto sea así cambia el valor del Campo ETOCheck de la Propiedad SimData asociado a dicho avión para indicar que ha entrado en el Sector. También manipula el Campo Comms de la misma Propiedad para que la aeronave mande el mensaje de Piloto 1, además de calcular la posición del avión en este punto, para lo que se usa el Rumbo de Entrada y la Velocidad de Vuelo. Por último hace una llamada a la función plotAirplanes que se describirá más adelante.

SimNav(app), esta función es el núcleo del Funcionamiento de la Simulación, se la llama igual que la anterior a Cada Periodo del Temporizador y calcula las actuaciones de cada aeronave. La funcionalidad principal de esta función es calcular la siguiente posición del avión, en intervalos de 10 segundos. Sin embargo hay que hacer que Navegue la Ruta establecida y realice las Incidencias programadas adecuadamente. Por ello se usa un parámetro llamado Turning, que se almacena en la Propiedad SimData, e indica si una aeronave concreta está virando o no, toma valores 0 si no está virando, 1 si está virando acorde a la Ruta y 2 si vira acorde a una Incidencia de Tipo 3 (desvío no Programado). Adicionalmente, se tienen otros tres parámetros que determinan el comportamiento de la aeronave: RouteTrack, Climbing y DoneIncident. Los tres se almacenan en la Propiedad SimData, y determinan respectivamente el índice dentro de la variable Ruta del Punto de Ruta siguiente, si se está ascendiendo (+1), descendiendo (-1) o manteniendo altitud (0) y por último si la Incidencia asociada a la aeronave ha ocurrido (true) o todavía no (false). Si se está dirigiendo al último punto de la Ruta, no se tiene en cuenta la variable Turning, pero si no, esta determina el régimen de actuación en el que se encuentra la aeronave, y por ende el método para calcular la posición siguiente.

En modo Turning 0 (no vira), se tiene a la aeronave volando en dirección al siguiente Punto de la Ruta, y a cada llamada calcula la posición que alcanzaría si empezara un viraje coordinado hacia el rumbo entre

el Punto de Ruta al que se dirige y el siguiente Punto de Ruta en la lista en la posición siguiente, una vez calculada esta posición futura, se comprueba si esa posición estaría dentro de una aerovía de 10 nm de ancho con centro el Radial que une dichos Puntos de Ruta, si fuera así, la variable Turning pasa a valer 1. En caso de que no haga falta viraje, cuando la aeronave alcanza una posición a menos de 10 segundos del Punto al que se dirige yendo a su Velocidad de Vuelo, aumenta la variable RouteTrack en 1.

En Turning 1 (vira según la Ruta establecida), se calcula el rumbo objetivo y se va variando el Rumbo de la Aeronave en intervalos de 30 grados (pues un viraje coordinado [16] es aquel en que el rumbo varía a una velocidad angular de 3º/seg). Cabe destacar que se toman medidas para que el viraje siempre sea el más corto de realizar, véase, si la diferencia entre los dos Rumbos es mayor de 180º, se le suma o se le resta en función del sentido de ese primer giro 360º al Rumbo objetivo, de esta manera, al calcular el sentido de giro se toma el más corto, adicionalmente, una vez terminado el bucle de virajes de la función, si el rumbo de la aeronave no está comprendido entre 0 y 359º, se le suma o se le resta 360º acordemente para que tome un rumbo válido, esto se hace para contrarrestar el hecho de que al variar el Rumbo objetivo para obtener el sentido de giro más corto, se puede salir el rumbo de la aeronave de los márgenes válidos, pero al contrarrestar esta posibilidad, una vez ocurre esto la diferencia entre los dos rumbos no puede superar los 180º, de modo que no será necesario ajustar el Rumbo objetivo. Estando en Turning 1, si la posición siguiente de la aeronave queda dentro de la aerovía correspondiente, se cambia el Rumbo de la Aeronave al Rumbo final y se vuelve a dejar la variable Turning en 0.

En Turning 2 (viraje no programado) se hace lo mismo que en Turning 1, con la diferencia de que el Rumbo objetivo no se calcula en base al siguiente tramo de Ruta, sino que se obtiene del Campo Incident.Target de la Propiedad AirplaneSim. Cuando el Controlador lidia con el hecho de que una aeronave se ha desviado sin autorización, se le autoriza un rumbo perpendicular al del tramo de Ruta sobre el que se ha desviado y se introduce este Rumbo en el campo Incident.Target, también se modifica el valor del Campo Autoriz de la Propiedad SimData, para indicar que ese es un rumbo autorizado. Cuando se alcanza el rumbo objetivo en Turning 2 y la Autorización está activa, se pasa a Turning 0 de nuevo, de modo que el avión vuelve a su Ruta establecida.

Cuando el campo RouteTrack alcanza el final de la Ruta, la aeronave se dirige hacia ese punto sin cambios hasta que llega a una distancia que le llevaría 1 minuto y medio recorrer con su Velocidad de Vuelo, se varía el campo Comms para que la aeronave mande el mensaje de Piloto 5, que solicita la Frecuencia para el cambio de Sector, cuando se recibe respuesta del Controlador y esta es reconocida por el piloto (véase, ha mandado el mensaje de Piloto 4), se cambia la variable ETOCheck de la aeronave a valor 2, representando así que la aeronave ha salido del Sector, y por ende, ya no debe tenerse en cuenta para los cálculos ni se tiene que representar.

Una vez se termina el bucle de Navegación Horizontal, se almacena en el campo Tracks de la Propiedad SimData, la posición anterior de la aeronave, de manera que este campo se puede usar para representar el rastro de la aeronave. Inmediatamente después de esto, comienza el bucle de Navegación vertical. Si el campo Climbing de la Propiedad SimData es 0, ni siquiera se entra en el bucle. En cambio, si el campo Climbing es distinto de 0, se entra en el bucle que varía la Altitud en Pies de la Aeronave (campo Z de la Propiedad AirplaneSim), en el sentido que indica Climbing, a una razón de 1000/6 ft (la velocidad de subida de un A320 en crucero, que es de donde se obtienen los datos de actuaciones de aeronaves para la Simulación, es de 1000 ft/min). Cuando la Altitud alcanza el Objetivo marcado por el campo Incident.Target de la Propiedad AirplaneSim, se cambia el valor de Climbing a 0. Adicionalmente, durante el proceso de ascenso o descenso, en los Campos de Texto del Botón del Panel de Raíles cambia según cambia la Altitud de la aeronave, y se le añade el carácter de acento circumflejo o ‘v’ dependiendo de si está ascendiendo o descendiendo, a parte, se modifica también el campo correspondiente de la Propiedad FlightStrip así como el del Panel de Ficha de Progresión de Vuelo si la Ficha activa es la de la aeronave en cuestión.

Por último se tiene un último bucle que gestiona las Incidencias. En este bucle no es entra a no ser que se tenga una Incidencia programada y esta no se haya resuelto, comprobando el Campo DoneIncident. La gestión de las Incidencias varía según si su momento de ocurrencia es Antes o Después de alcanzar el Punto de Ruta detonante, además de variar según el tipo de Incidencia.

Para las Incidencias Tipo 1 (Cambio de Nivel de Vuelo), si ocurren antes, se calcula la distancia necesaria para alcanzar el Nivel de Vuelo objetivo, y se le suma 1 minuto de recorrido, con esta distancia respecto al Punto Detonante se lanza el mensaje de Piloto de Solicitud de Cambio de Vuelo (mensaje 2), y si es autorizado y reconocido, se cambia al Climbing adecuado. Si no se autoriza, se cambia a DoneIncident verdadero y se sigue la Ruta. Ocurre lo mismo para cuando ocurre Después, solo que en ese caso se solicita a minuto y medio de alcanzar el Punto Detonante, y no se cambia al Climbing adecuado hasta que no se ha terminado de hacer el viraje al siguiente punto de la Ruta.

Para las Incidencias Tipo 2 (Solicitud de Directo), si ocurren antes, se solicita el Directo a minuto y medio de alcanzar el Punto Detonante y si se autoriza se modifica el campo Ruta para no incluir aquellos puntos que se salte el Directo. Si ocurre Después, se sigue solicitando el Directo a minuto y medio de alcanzar el punto detonante, pero no se aplica el cambio en la variable ruta hasta después de haber completado el viraje. Si no se Autoriza se cambia a DoneIncident verdadero y se sigue el Plan de Vuelo establecido. Tanto para este tipo de incidencias como para el anterior, en cuanto se efectúan los cambios asociados a la incidencia se cambia a DoneIncident verdadero.

Por último, en Incidencias Tipo 3 no se requiere de autorización del Controlador para iniciar la Incidencia, de modo que si ocurre antes de alcanzar el Punto Detonante, cuando se alcanzan minuto y medio de recorrido hasta el Punto, se activa la Incidencia y se pasa a Turning 2, cambiando DoneIncident a verdadero. Si ocurre Después, en cuanto se termina el viraje en el Punto Detonante se pasa a Turning 2, se reduce el RouteTrack en 1 (de manera que al resolverse la incidencia vuelva al tramo de Ruta del que se ha desviado) y se cambia DoneIncident a verdadero. Para terminar con la función SimNav, se hace una llamada a la función plotAirplanes.

CommCheck(app), esta función monta y gestiona los mensajes de los Pilotos en base al campo Comms de la Propiedad SimData, que toma valores de 0 a 6, representando el 0 que aún no se ha mandado el siguiente mensaje, los valores del 1 al 5 representan el mensaje que se manda y el valor 6 representa que ya se ha mandado un mensaje. De esta manera solo se entra al bucle de mandar mensaje si se encuentra entre los valores del 1 al 5. Existen tres tipos de mensaje de los Pilotos: 1 y 5, que requieren de un valor variable, concretamente el indicador de la Aeronave que manda el mensaje, 2 y 3, los cuales requieren que se añada información relativa a una Incidencia, pues son los mensajes de solicitud de autorización, y por último el 4, que es el mensaje de reconocimiento de respuesta 'Recibido Control, seguiremos instrucciones', y no necesita de ningún dato específico de la aeronave. Cuando se entra en el bucle de mandar mensajes se monta el respectivo mensaje y luego se manda al Área de Texto que actúa como Panel de Comunicaciones, introduciéndose arriba del todo, igual que el resto de mensajes nuevos, de ese modo se evita el tener que "Scrollear" hacia abajo por parte del Controlador constantemente para ver los mensajes nuevos. Adicionalmente, cuando se manda un mensaje se cambia Comms a valor 6 para que no se vaya mandando el mismo mensaje cada 10 segundos.

plotAirplanes(app), esta función representa los Aviones que están dentro del Sector enla Pantalla de Radar, usando el Esquema de Colores siguiente: el Avión es un triángulo orientado según el Rumbo de la Aeronave de 0.5 nm de altura y base de 0.4 nm, cuyo vértice opuesto a la base es tiene las coordenadas X e Y del Avión, el vector Velocidad de Vuelo se representa con una línea magenta que se extiende a la Velocidad de Vuelo por minuto, y termina en una cruz blanca, por último, el rastro de la aeronave se representa con una línea de puntos blancos. Si un avión tiene su ETOCheck distinto de 1, no se representa. Adicionalmente, en cada Avión se representan los datos básicos del Vuelo, como son el Indicativo de la Aeronave, su nivel de Vuelo y su velocidad en Nudos, siempre a la derecha de la aeronave y de forma que no estorbe al vector Velocidad de Vuelo, además se representa en color verde.

pos = finwpsim(app,wp), esta función hace exactamente lo mismo que su homóloga, pero en vez de buscar en los árboles correspondientes del Editor de Escenarios, busca en la Variable Sector de la Propiedad SectorSimulador.

inawy = inawycheck(,hdgtarg,wptpos,acpos), esta función calcula si la posición de la aeronave (acpos) está dentro del área de una Aerovía que parte del Punto de Ruta wptpos con Rumbo hdgtarg con una anchura de 10 nm. Si es así la función devuelve un valor lógico verdadero, y si no devuelve un valor lógico falso.

Funciones de Respuesta (CallbackFcn)

startupFcn, esta Función se ejecuta justo después de haberse creado los Componentes de la Aplicación, y su funcionalidad consiste en: primero, mover la figura de la Aplicación a una posición central en la Pantalla del Ordenador; segundo, cargar los datos del archivo 'escenarioslist.mat' en la Propiedad S, y crear los Nodos de los respectivos Árboles de la Selección de Escenarios; por último, declara los valores iniciales de las distintas Propiedades de la Aplicación. Adicionalmente, se modifican las propiedades Tag de los objetos Árbol de las Zonas Restringidas, los Aeropuertos y los Puntos de Paso al valor 'Tree', y las propiedades UserData de los objetos Botón del Raíl de Fichas, Lámpara y Campo de Edición de Texto para el Tiempo de Paso por un Punto de Ruta al valor 1.

EditordeEscenariosButtonPushed y **SimuladorButtonPushed** son las funciones de respuesta asociadas a los botones de la Pantalla de Inicio que llevan a sendas Pantallas de Edición de Escenarios.

EditorseleccionbackbuttonButtonPushed y **SimuladorseleccionbackcuttonButtonPushed** son las funciones de respuesta asociadas a los respectivos botones de 'Atrás' en ambas Pantallas de Selección de Escenarios, los cuales devuelven a la Pantalla de Inicio.

TreeEditorSelectionChanged es la función de respuesta a un cambio de Selección en el objeto Árbol de la Pantalla de Selección de Escenarios del Editor, la cual almacena el Nombre del Nodo en la Propiedad SectorEditorSelec.

NuevoEscenarioButtonPushed es la función de respuesta a pulsar el botón de Nuevo Escenario en la Pantalla de Selección de Escenarios del Editor, el cual abre un Panel en el que gestionar la Creación de un Nuevo Escenario. Al abrir este Panel, se desactiva la funcionalidad de todos los objetos de la Pantalla de Selección del Editor.

NombredelnuevoescenarioEditFieldValueChanged es la función que almacena en la Propiedad NombreNuevoEscenario el valor que se ha introducido en el Campo de Texto del Panel de Creación de Escenario.

AtrsButtonPushed es la función de respuesta asociada al botón de Atrás del Panel de Creación de Escenario, esta cierra dicho Panel y reactiva los objetos de la Pantalla de Selección del Editor.

CrearButtonPushed es la función de respuesta asociada al botón Crear del Panel de Creación de Escenario, esta añade el Nombre del Nuevo Escenario al archivo escenariosslist.mat, crea los Nodos que representan el Nuevo Escenario en los Árboles de Selección, crea el archivo .mat del Escenario, Cierra el Panel de Creación y lleva a la Pantalla del Editor de Escenarios.

Las siguientes 4 Funciones de Respuesta que se ven en el Código del Apéndice A tienen las mismas respectivas funcionalidades que sus homólogas descritas anteriormente, con la diferencia de que abre un Panel distinto en el que se gestiona el Cambio de Nombre del Escenario, al abrirse se rellena el Campo de Texto con el Nombre del Escenario Seleccionado (si no hay ningún Escenario Seleccionado no se abre el Panel) que se puede editar y al Renombrar el Escenario, en vez de llevar a la Pantalla del Editor se queda en la Pantalla de Selección del Editor.

EditorbackbuttonButtonPushed es la función de respuesta asociada al botón de Atrás de la Pantalla del Editor, la cual llama a todas las funciones descritas que restauran a los valores iniciales y limpian los distintos objetos del Editor de Escenarios, además de limpiar el objeto ejes representando el punto [0 0] con la propiedad hold de dicho objeto desactivada. Por último, lleva a la Pantalla de Selección del Editor.

EliminarEscenarioButtonPushed es la función de respuesta asociada al botón Eliminar Escenario de la Pantalla de Selección del Editor, esta solo se activa si hay un Escenario Seleccionado, y lo que hace es eliminar dicho Escenario, quitándolo del archivo escenariosslist.mat y eliminando su propio archivo .mat así como sus respectivos Nodos de los Árboles de Selección.

CargarEscenarioButtonPushed es la función de respuesta asociada al botón Cargar Escenario de la Pantalla de Selección del Editor, esta carga el archivo .mat del Escenario Seleccionado (no se activa si no hay un Escenario Seleccionado) en la Propiedad SectorEditor y hace una llamada a la función LoadScenario para a continuación, llevar a la Pantalla del Editor.

TreeSectorSelectionChanged es la función de respuesta asociada al objeto Árbol del Sector de la Pantalla del Editor, esta abre el Panel de Edición correspondiente al Nodo Seleccionado, en caso de no haber ningún Nodo Seleccionado se dejan cerrados todos los Paneles de Edición. Adicionalmente, se llama a la función clearSelectedNodesEditor.

TipodeFronteraButtonGroupSelectionChanged y **TipodeFronteraRZButtonGroupSelectionChanged** son las funciones de respuesta asociadas a los objetos Grupo de Botones de los respectivos Paneles de Frontera del Sector y Zonas Restringidas, estas tienen como funcionalidad el hacer visible el correspondiente Panel de Vértices, Arcos o Altitudes de sus respectivos Paneles cuando el botón seleccionado de su Grupo cambia. Si el botón seleccionado es el de Selecciona Una, ningún Panel de los descritos se deja visible.

XnvertEditFieldValueChanged, **YnvertEditFieldValueChanged**, **Xn1vertEditFieldValueChanged**, **Yn1vertEditFieldValueChanged** y sus homólogas con RZ, son las funciones de respuesta de los Campos Numéricos Editables de los respectivos Paneles de Vértices. Estas rellenan los correspondientes Campos de la Propiedad Vert con los datos introducidos.

MostrarBorderVButtonPushed y **MostrarRZVButtonPushed** son las funciones de respuesta asociadas a los botones de Mostrar de los respectivos Paneles de Vectores de la Frontera del Sector y de las Zonas Restringidas. Estas se usan para representar en la "Pantalla de Radar" del Editor los tramos de Frontera tipo Vértice introducidos sin llegar a Guardarlos en la variable Sector.

GuardarBorderVButtonPushed y **GuardarRZVButtonPushed** son las funciones de respuesta asociadas a los botones de Guardar de los respectivos Paneles de Vectores de la Frontera del Sector y de las Zonas Restringidas. Estas tienen la funcionalidad de almacenar en la variable Sector el Vértice introducido, si no se ha seleccionado ningún tramo de Frontera en el objeto árbol del Panel de Edición correspondiente, se almacenan ambos Vértices introducidos al final de la lista que enumera los tramos de Frontera (en el caso de las Zonas Restringidas se añade a la última Zona Restringida creada o en la Zona Restringida Seleccionada si se ha Seleccionado un Nodo correspondiente a una Zona Restringida y no a un tramo de

Frontera, si no hay ninguna creada, el respectivo botón no hace nada). Si se tiene un Nodo Seleccionado en el Árbol correspondiente, se añade solo el segundo Vértice introducido inmediatamente después del Tramo de Ruta Seleccionado. También crea un Nodo en el respectivo Árbol y le atribuye como nombre 'Vertice X' donde X es el número en la lista de tramos de frontera, y almacena X en la Propiedad NodeData del Nodo correspondiente, en el caso de las Zonas Restringidas el Nodo creado es un Nodo "hijo" del Nodo de la Zona Restringida Correspondiente.

BorderTreeSelectionChanged y **RZTreeSelectionChanged** son las funciones de respuesta asociadas a los Respectivos objetos Árbol de los Paneles de Frontera del Sector y de Zonas Restringidas. Estas cargan Datos en el Panel de Vértices o Arcos que esté activo en el momento de cambio de Selección, y almacena en los Campos que representan el origen del tramo de Frontera las coordenadas del último Punto del tramo de Frontera Seleccionado (en caso de un Vértice el tramo solo tiene un punto así que se toma ese). En caso de Seleccionar un Arco teniendo el Panel de Arcos activo, se almacena en los Campos los datos del Arco (para lo cual se llama a las funciones necesarias para calcular aquellos datos que no se almacenan directamente). Adicionalmente, se representa un círculo sobre los puntos significativos del tramo de Frontera Seleccionado en la Pantalla de Radar (para el Vértice es solo un punto, pero para un Arco se consideran como puntos significativos los extremos del Arco y su punto medio, que se ha llamado I). El Árbol de las Zonas Restringidas distingue entre Nodos de tramos de Fronteras y Nodos de Zonas Restringidas (gracias a la Propiedad Tag del objeto "padre" del Nodo Seleccionado) de manera que si se Selecciona un Nodo de una Zona Restringida se carga en los Campos correspondientes los datos del último tramo de Frontera de esa Zona y en la Pantalla de radar se resalta la Frontera entera de la Zona Restringida Seleccionada.

EliminarBorderButtonPushed y **EliminarRZButtonPushed** son las funciones de respuesta asociadas a los respectivos botones Eliminar de los Paneles de Frontera del Sector y de Zonas Restringidas. Estas eliminan el tramo de Frontera Seleccionado (si no hay ninguno seleccionado no hace nada) de la variable Sector y eliminando también el correspondiente Nodo. En el caso del Panel de Zonas Restringidas, si se tiene Seleccionado el Nodo de una Zona Restringida, se elimina la Zona Restringida al completo de la variable Sector y del Árbol.

XnarchEditFieldValueChanged, **YnarchEditFieldValueChanged**, **Xn1archEditFieldValueChanged**, **Yn1archEditFieldValueChanged**, **BARadiusEditFieldValueChanged**, **BAConcaveConvexSwitchValueChanged**, **BACortoLargoSwitchValueChanged**, y sus homólogas con RZ, son las funciones de respuesta asociadas a los Campos Numéricos Editables y los Interruptores que contienen los parámetros de un Arco. Estas funciones rellenan los correspondientes Campos de la Propiedad Arch.

MostrarBorderAButtonPushed y **MostrarRZAButtonPushed** son las funciones de respuesta asociadas a los respectivos botones Mostrar de los Paneles de Arcos de la Frontera del Sector y de Zonas Restringidas. Estas, de modo similar a sus homólogas para Vértices, sirven para representar los tramos de Frontera introducidos en la Pantalla del Radar del Editor (para lo cual llaman a la función CalcArch) sin guardarlo en la variable Sector ni en el Árbol correspondiente.

GuardarBorderAButtonPushed y **GuardarRZAButtonPushed** son las funciones de respuesta asociadas a los respectivos botones Mostrar de los Paneles de Arcos de la Frontera del Sector y de Zonas Restringidas. Estas, de modo similar a sus homólogas para Vértices, sirven para almacenar las coordenadas de los puntos del tramo de Frontera en la variable Sector y crear el Nodo del tramo en el Árbol correspondiente (en el caso de Zona Restringida, como Nodo "hijo" de la Zona correspondiente), con el Nombre 'Arco X' donde X es el número en la lista de tramos de frontera, y almacena X en la Propiedad NodeData del Nodo correspondiente.

ZinfEditFieldValueChanged, **ZsupEditFieldValueChanged**, **RZZinfEditFieldValueChanged** y **RZZsupEditFieldValueChanged** son las funciones asociadas a los Campos Numéricos Editables de los respectivos Paneles de Altitud de la Frontera del Sector y las Zonas Restringidas. Estas rellenan los datos de límites verticales en Altitudes en pies (ft) en os respectivos Campos de la Propiedad Alt.

GuardarBorderAltButtonPushed y **GuardarRZAltButtonPushed** son las funciones de respuesta asociadas a los botones Guardar de los respectivos Paneles de Altitud de la Frontera del Sector y las Zonas Restringidas. Estas guardan los datos almacenados en la propiedad Alt en la variable Sector y rellenan los correspondientes Campos de Edición de Texto que representan los límites verticales del Sector y las Zonas Restringidas.

NuevaZonaRestringidaButtonPushed es la función de respuesta asociada al botón Nueva Zona Restringida del Panel de Zonas Restringidas, esta crea un Nodo en el Árbol de Zonas Restringidas con el Nombre 'Zona Restringida X' donde X es el número en la lista de Zonas Restringidas, también le añade a la Propiedad NodeData X y en la Propiedad Tag almacena 'Node'.

NombreAirportEditFieldValueChanged y **NombreWPEditFieldValueChanged** son las funciones de respuesta asociadas a los Campos de Edición de Texto de los respectivos Paneles de Aeropuertos y Puntos de

Paso. Estas detectan si el texto introducido, que se pretende que sea el Nombre de un Aeropuerto o un Punto de Paso, sea de 5 letras y en mayúsculas (para seguir el criterio 5LNC) y lo almacena en el campo correspondiente de la Propiedad Airport o Waypoint, en caso de que no sea así, el valor del campo correspondiente de la Propiedad Airport o Waypoint se rellena con 'FAIL'. Adicionalmente, se coloca de base el fondo del Campo de color blanco al cambiar su valor, independientemente de si se cambia a un nombre válido o no.

XairportEditFieldValueChanged, **YairportEditFieldValueChanged**, **XwaypointEditFieldValueChanged** y **YwaypointEditFieldValueChanged** son las funciones de respuesta asociadas a los Campos Numéricos Editables de los respectivos Paneles de Aeropuertos y Puntos de Paso. Estas almacenan los datos de las coordenadas en los correspondientes campos de la Propiedad Airport o Waypoint.

GuardarAirportButtonPushed y **GuardarWaypointButtonPushed** son las funciones de respuesta asociadas a los botones Guardar de los respectivos Paneles de Aeropuertos y Puntos de Paso. Estas detectan primero si el valor del campo Name de la Propiedad correspondiente Airport o Waypoint tiene una longitud de 4 o 5 (si tiene 5 es que es un nombre válido, si tiene 4 es que es FAIL), y en caso de tener longitud 4 cambia el valor del Campo de Texto correspondiente a 'ERROR' y su fondo a color rojo. Si tiene longitud 5 se almacena en el Nodo Seleccionado del correspondiente Árbol o, si no hay ningún Nodo Seleccionado, se crea uno, con Nombre el nombre del Aeropuerto o Punto de Paso y valor de la Propiedad NodeData el índice dentro de la variable Sector, además de almacenar los respectivos datos en sendos campos de la variable Sector. Adicionalmente se crea un Nodo "hijo" que contiene las coordenadas de la posición del Aeropuerto o Punto de Paso.

AirportTreeSelectionChanged y **WaypointTreeSelectionChanged** son las funciones de respuesta asociadas a los Árboles de los respectivos Paneles de Aeropuertos y Puntos de Paso. Estas cargan en los respectivos campos los datos del Aeropuerto o Punto de Paso Seleccionado, si no hay ninguno se restauran estos Campos a valores iniciales. Si se tiene un Nodo Seleccionado en el correspondiente Árbol, adicionalmente, se representa un cajetín entorno al nombre representado del Aeropuerto o Punto de Paso Seleccionado en la Pantalla de Radar del Editor. Si el Aeropuerto o Punto de Paso Seleccionado contiene Pistas definidas (en el caso del Aeropuerto) o es Fronterizo y tiene un valor de Frecuencia del Sector Adyacente (en el caso del Punto de Paso), se hace visible el respectivo Panel de gestión de Pistas o de Frecuencia y se rellena con los datos almacenados en la variable Sector.

EliminarAirportButtonPushed y **EliminarWaypointButtonPushed** son las funciones de respuesta asociadas a los botones Eliminar de los respectivos Paneles de Aeropuertos y Puntos de Paso. Estas eliminan el Aeropuerto o Punto de Paso Seleccionado en el correspondiente Árbol, tanto de dicho Árbol como de la variable Sector. Si no hay un Nodo Seleccionado, este botón no hace nada.

AddRWYButtonPushed y **WaypointFronterizoButtonPushed** son las funciones de respuesta asociadas a los botones Añadir Pista del Panel de Aeropuertos y Fronterizo del Panel de Puntos de Paso respectivamente. Estas abren los respectivos paneles de gestión de Pistas en el caso del Panel de Aeropuertos y de gestión de Frecuencia de Cambio de Sector en el caso del Panel de Puntos de Paso. En el caso del Panel de Puntos de Paso, adicionalmente, se llevan las coordenadas del Punto al Punto de la Frontera más cercano, para tramos de Frontera tipo Vértice, se crea la Recta de dicho tramo y se busca el punto más cercano.

RWYNewEditSwitchValueChanged es la función de respuesta asociada al Interruptor del Panel de gestión de Pistas de un Aeropuerto que determina si se está añadiendo una pista o editando una ya añadida. El valor que toma este Interruptor decide si se abre la sección de Nueva Pista o Editar Pista.

NewRWYSpinnerValueChanged y **RWYEditSpinnerValueChanged** son las funciones de respuesta asociadas a los respectivos objetos tipo Rodante de las secciones de Nueva Pista o Editar Pista. El valor de estos objetos se almacena en el Campo Runway de la Propiedad Airport.

RWYEditDropDownValueChanged es la función de respuesta asociada al objeto Desplegable de la sección de Editar Pista. En este objeto se almacena la lista de pares de Pistas, y al seleccionar un par de Pistas se muestra en el rodante de la Sección de Edición de Pista el Rumbo de la Pista, siendo este el que está más orientado hacia el Este.

GuardarNewRWYButtonPushed y **GuardarRWYEditButtonPushed** son las funciones de respuesta asociadas a los botones Guardar de las respectivas secciones de Nueva Pista y Editar Pista. Estas almacenan en la variable Sector, en el campo respectivo al Aeropuerto Seleccionado en el Árbol, el rumbo de la Pista almacenada en el campo Runway de la Propiedad Airport y el de su Pista Pareja, guardando primero la que tiene un rumbo más hacia el Este. Dependiendo de si se está en la sección de Nueva Pista o Editar Pista, se almacenan al final del campo Runway del Aeropuerto en la variable Sector o se almacenan en el espacio donde estaba el par de Pistas Editado.

EliminarRWYEditButtonPushed es la función de respuesta asociada al botón Eliminar de la sección de Editar Pista, esta elimina el par de Pistas seleccionado en el Desplegable de la variable Sector y del objeto Desplegable.

FreqMHzEditFieldValueChanged es la función de respuesta asociada al Campo Numérico Editable del Panel de gestión de Frecuencia del Panel de Puntos de Paso. Esta almacena el valor del Campo, que representa la Frecuencia del Sector Adyacente en MHz, en el campo Freq de la Propiedad Waypoint.

GuardarWPFreqButtonPushed es la función de respuesta asociada al botón Guardar del Panel de gestión de Frecuencia del Panel de Puntos de Paso. Esta almacena el valor del campo Freq de la Propiedad Waypoint en la variable Sector.

NiveldeVueloEditFieldValueChanged, **MatrculadelaaeronaveEditFieldValueChanged**, **VelocidaddeVueloktEditFieldValueChanged**, **ETOEditFieldValueChanged** y **EHDGEditFieldValueChanged** son las funciones de respuesta asociadas a los Campos Editables de la Pestaña de Información General del Grupo de Pestañas en el Panel de Aviones. Estas almacenan los Datos en los respectivos campos de la Propiedad Airplane. Para la Matrícula de la Aeronave y su ETO se hacen previamente a su almacenaje sendas llamadas a las funciones isvalidcallsign y isvalidETO para comprobar si son válidos, en caso de que no lo sean, no se almacenan. La ETO además se almacena en formato de variable Duration.

RouteWPEditFieldValueChanged es la función de respuesta asociada al Campo de Edición de Texto de la Pestaña de Ruta del Grupo de Pestañas en el Panel de Aviones. Esta almacena el valor del Campo en la Propiedad RouteWP.

GuardarRouteWPButtonPushed es la función de respuesta asociada al botón Guardar de la Pestaña de Ruta del Grupo de Pestañas en el Panel de Aviones. Esta comprueba si el valor almacenado en RouteWP es un Punto de Paso o un Aeropuerto almacenado en la variable Sector y lo añade al Campo Route de la Propiedad Airplane, así como al final del Árbol de Ruta que está en esa misma Pestaña (si no lo es no se hace nada). En caso de que haya un Nodo Seleccionado en el Árbol de Ruta, el Punto de Ruta se añade inmediatamente después del Seleccionado.

EliminarRouteWPButtonPushed es la función de respuesta asociada al botón Guardar de la Pestaña de Ruta del Grupo de Pestañas en el Panel de Aviones. Esta Elimina el Punto de Ruta del Árbol de Ruta y del Campo Route de la Propiedad Airplane.

RouteTreeSelectionChanged es la función de respuesta asociada al Árbol de Ruta de la Pestaña de Ruta del Grupo de Pestañas en el Panel de Aviones. Esta deja la Propiedad RouteWP y el Campo de Texto de la Pestaña de Ruta en valores iniciales y representa en la Pantalla de Radar del Editor un cajetín entorno al Nombre del Punto de Paso o Aeropuerto Seleccionado en el Árbol (si no hay ningún Punto de Ruta Seleccionado esto no se hace).

TipodeIncidenciaDropDownValueChanged es la función de respuesta asociada al Desplegable de la Pestaña de Incidencias del Grupo de Pestañas en el Panel de Aviones. Esta sirve para determinar el tipo de incidencia y hacer visible el Panel correspondiente a dicha Incidencia. Adicionalmente, restaura a valores iniciales a la Propiedad Incident antes de determinar el tipo de Incidencia, y una vez determinado lo almacena en el campo Type de la Propiedad Incident.

WaypointTriggerCFLEditFieldValueChanged, **WaypointTriggerSDEditFieldValueChanged** y **WaypointTriggerCRNPEditFieldValueChanged** son las funciones asociadas a los Campos de Edición de Texto donde se introduce el Punto Detonante de la Incidencia en sus respectivos Paneles de Incidencias dentro de la Pestaña de Incidencias del Grupo de Pestañas en el Panel de Aviones. Estas almacenan el Punto en el campo Trigger de la Propiedad Incident.

FLObjetivoEditFieldValueChanged, **WPDirectObjetivoEditFieldValueChanged** y **RumboNuevoEditFieldValueChanged** son las funciones de respuesta asociadas a los Campos Editables donde se introduce el Objetivo de la Incidencia en sus respectivos Paneles de Incidencias dentro de la Pestaña de Incidencias del Grupo de Pestañas en el Panel de Aviones. Estas almacenan el Dato en el campo Target de la Propiedad Incident.

CFLDropDownValueChanged, **SDDropDownValueChanged** y **CRNPDropDownValueChanged** son las funciones de respuesta asociadas a los objetos tipo Desplegable donde se indica el Tiempo de Ocurrencia de la Incidencia en sus respectivos Paneles de Incidencias dentro de la Pestaña de Incidencias del Grupo de Pestañas en el Panel de Aviones. Estas almacenan el Valor Seleccionado (que puede ser '1' para antes o '2' para después) en el campo Time de la Propiedad Incident.

GuardarIncidenciaButtonPushed es la función de respuesta asociada al botón Guardar de la Pestaña de Incidencias del Grupo de Pestañas en el Panel de Aviones. Esta hace comprobaciones de los datos almacenados en la Propiedad Incident en base al Tipo de Incidencia y almacena esos datos en el campo Incident de la Propiedad Airplane si son válidos. Para Incidencia tipo 0 (no hay incidencia) no hacen falta comprobaciones,

para Incidencia tipo 1 o 3 (Cambio de Nivel de Vuelo o Desvío no Programado) comprueba si el Punto Detonante (Trigger) es un Punto de Paso o un Aeropuerto del Sector, para Incidencia tipo 2 (Solicitud de Directo) comprueba si tanto el Punto Detonante (Trigger) como el Objetivo (Target) son Puntos de Paso o Aeropuertos del Sector.

EliminarIncidenciaButtonPushed es la función de respuesta asociada al botón Eliminar de la Pestaña de Incidencias del Grupo de Pestañas en el Panel de Aviones. Esta restaura al objeto Desplegable a valor inicial (tipo 0) y llama a su función de respuesta, restaurando así a valores iniciales a la Propiedad Incident, y almacenando a esta en el campo Incident de la Propiedad Airplane.

GuardarAirplaneButtonPushed es la función de respuesta asociada al botón Guardar del Panel de Aviones, esta comprueba primero que los campos Callsign y ETO de la Propiedad Airplane no están vacíos y si no lo están almacena la Propiedad Airplane en la variable Sector y crea un Nodo en el Árbol de Aviones con Nombre el Identificador de la Aeronave. Si se tiene un Avión seleccionado en dicho Árbol, en vez de almacenarse en un espacio nuevo se almacenan los Datos de la Propiedad Airplane en el espacio del Avión Seleccionado.

AirplaneTreeSelectionChanged es la función de respuesta asociada al Árbol de Aviones del Panel de Aviones. Esta carga los datos de la Aeronave Seleccionada a la variable Airplane (si no se tiene Seleccionada ninguna, se restaura a los valores iniciales) para a continuación hacer sendas llamadas a las funciones updateAirplaneTab y plotRoute.

EliminarAirplaneButtonPushed es la función de respuesta asociada al botón Eliminar del Panel de Aviones. Esta Elimina al Avión Seleccionado de la variable Sector y del Árbol de Aviones.

GuardarSectorButtonPushed es la función de respuesta asociada al botón Guardar Cambios de la Pantalla del Editor, esta guarda la variable Sector actual en el Archivo .mat del Escenario.

GuardarHorarioButtonPushed es la función de respuesta asociada al botón Guardar del Panel de Horario de la Pantalla del Editor. Esta almacena la Propiedad TimeSector en el campo Time de la variable Sector si sus campos no están vacíos (si alguno de ellos lo está, el campo Time de la variable Sector se declara como matriz vacía).

HorainicioEditFieldValueChanged y **HorafinalEditFieldValueChanged** son las funciones de respuesta de los respectivos Campos de Edición de Texto del Panel de Horario. Estas comprueban si el valor introducido es una hora válida (llamando a la función isvalidETO) y, en caso afirmativo, las almacena en su correspondiente campo de la Propiedad TimeSector. En caso de que no sea una hora válida el correspondiente campo de la Propiedad TimeSector pasa a ser una matriz vacía y el valor del Campo de Texto pasa a ser '00:00'.

CargarEscenarioSimButtonPushed es la función de respuesta asociada al botón Cargar Escenario de la Pantalla de Selección del Simulador, esta carga el Archivo .mat correspondiente al Escenario seleccionado en el Árbol de la misma Pantalla en la Propiedad SectorSimulador. Una vez cargado se llama a la función LoadScenarioSim y se pasa a la Pantalla del Simulador.

TPWPEFValueChanged es la función de respuesta asociada al Campo de Edición de Texto del Panel de Ruta del Panel de Ficha de Progresión de Vuelo en la Pantalla del Simulador (que representa el Tiempo de Paso por el Punto de la Ruta correspondiente) y a las subsiguientes copias de este Campo que se crean cuando se Carga una Ficha de Progresión de Vuelo. Esta primero determina cuál de sus Campos se ha modificado, mediante la variable event y su campo Source.UserData. Una vez determinado esto, si el valor introducido parece una hora pero contiene un carácter espacio en blanco al principio porque al Usuario se le ha olvidado o no ha pensado en eliminarlo, se elimina dicho espacio. A continuación se determina, en caso de que el valor tenga longitud 5 (véase, cabe la posibilidad de que sea una hora válida), se pasa por la función isvalidETO. El siguiente paso es encontrar el Avión al cual esta Ficha de Progresión de Vuelo pertenece, para lo cual se barre la Propiedad de la Aplicación asociada al objeto de los Botones del Panel de Raíl de Fichas y se busca aquel que esté activado (una vez encontrado se rompe el bucle for que hace el barrido con el comando break). Por último, si es una hora válida o tiene una longitud de 1, se almacena el valor en el campo TPWP correspondiente del espacio correspondiente de la Propiedad FlightStrip.

FLSRButtonValueChanged es la función de respuesta asociada al objeto Botón de Estado del Panel de Raíl de Fichas de la Pantalla del Simulador y a las subsiguientes copias de este Botón que se generan cuando se Carga el Escenario en el Simulador. Esta determina primero cuál de sus botones se ha pulsado, mediante la variable event y su campo Source.UserData. Una vez determinado esto se limpia el Panel de Ficha de Progresión de Vuelo y a continuación se desactivan todos los demás Botones del Panel de Raíl de Fichas. Por último se buscan los datos de la Ficha de Progresión de Vuelo del Avión asociado al Botón detonante almacenados en la Propiedad Flighstrip y se cargan en el Panel de Fichas de Progresión de Vuelo.

SalirSimButtonPushed es la función de respuesta asociada al botón Salir de la Pantalla del Simulador. Esta restaura a valores iniciales la variable Sector de la Propiedad SectorSimulador así como los distintos objetos de la Pantalla del Simulador y devuelve a la Pantalla de Selección del Simulador.

DropDownValueChanged es la función de respuesta asociada al Desplegable del Panel de Comunicación del Controlador, el cual se rellena con los identificadores de las Aeronaves al Cargar el Escenario en el Simulador. Esta determina la Aeronave seleccionada y carga los Datos relevantes sobre dicha Aeronave en el campo Data de la Propiedad Comminput. Adicionalmente restaura a valores iniciales el campo msg de la Propiedad Comminput, así como el Área de Texto del Mensaje que manda el Controlador.

AutorizarEntradaButtonPushed, **AutorizarCambioFLButtonPushed**, **NoAutorizarCambioFLButtonPushed**, **AutorizarDirectoButtonPushed**, **NoAutorizarDirectoButtonPushed**, **CambioaRumboAutorizadoButtonPushed** y **CambioFrecuenciaButtonPushed** son las funciones de respuesta asociadas a los respectivos botones del Panel de Comunicaciones del Controlador en la Pantalla del Simulador. Estas montan el Mensaje de Controlador correspondiente al botón pulsado (del 1 al 7 respectivamente), lo completan con los datos del campo Data de la Propiedad Comminput cuando es necesario, lo almacenan en el campo msg de la Propiedad Comminput ya montado, para mandarlo a continuación al Área de Texto donde se muestra el mensaje a enviar por el Controlador. Adicionalmente, para todos los mensajes excepto el 1 (autoriza la entrada) se hace una de dos cosas. Si el mensaje Autoriza una actuación (2 Cambio de Nivel de Vuelo, 4 Directo, 6 Rumbo de Retorno a la Ruta y 7 Frecuencia de Cambio de Sector) se modifica el valor del campo Autoriz.Type de la Propiedad SimData al valor adecuado para cada tipo de Autorización (1, 2, 3 y 4 respectivamente) para la Aeronave a la que se manda el mensaje. Si en cambio, el mensaje Desautoriza una actuación solicitada por la Aeronave (mensajes 3 y 5) se declara como valor lógico verdadero el campo DoneIncident de la Propiedad SimData, de manera que se ignora la Incidencia y se sigue la Ruta establecida.

SendButtonPushed es la función de respuesta asociada al botón Enviar del Panel de Comunicaciones del Controlador en la Pantalla del Simulador. Esta envía el mensaje almacenado en el Área de Texto del Panel de Comunicaciones del Controlador al Área de Texto de la Pantalla del Simulador, que actúa como Pantalla de Comunicaciones. El mensaje se introduce en lo alto del Área, del mismo modo que se hace en la función CommsCheck con los mensajes de los Pilotos. Adicionalmente, si el campo Autoriz.Type de la Propiedad SimData asociado a la Aeronave a la que se manda el mensaje es distinto de 0, el campo Autoriz.Value de la Propiedad SimData asociado a dicha Aeronave se declara como valor lógico verdadero, además de cambiar el valor del campo Comms de la Propiedad SimData asociado a la Aeronave a 4 (esto también se hace si el campo DoneIncident es un valor lógico verdadero y el valor del campo Autoriz.Type es 0). Por último se cambia el valor del Desplegable del Panel de Comunicaciones del Controlador a 0 y se llama a su función de respuesta.

PlaySimButtonValueChanged es la función de respuesta asociada al botón de estado Play, que inicia la Simulación. Esta función solo funciona cuando el botón pasa de estar Desactivado a estar Activado, y lo primero que hace es inhabilitarse a sí mismo y al botón de Salir de la Simulación (de modo que solo se pueda salir del Simulador si la Simulación está en Pausa), y también desactiva el botón de estado de Pausa y lo habilita. Lo siguiente que hace es definir el objeto Timer y almacenarlo en la Propiedad SimTimer. Los parámetros que se definen para el objeto Timer son: Periodo de 10 segundos, BusyMode en modo queue (de este modo ejecuta todas las funciones de respuesta en orden de llamada y sin saltarse ninguna), TaskstoExecute se calcula como el tiempo en segundos que queda de Simulación (diferencia entre TimeSim y la Hora Final) dividida entre 10, ExecutionMode en modo fixedRate (de este modo se llama a las funciones del Timer en los instantes correspondientes a intervalos de 10 segundos, pues ese es el Periodo), StartDelay 10 segundos (De este modo la primera Función del Timer se ejecuta 10 segundos después de que empiece y no al empezar), y para las StartFcn, TimerFcn y StopFcn se definen sendas funciones StartTimer, RunTimer y EndTimer, a las que se les da como argumento adicional a los que toma por el formato de las funciones de un Timer, la variable de la Aplicación (app). Por último, se inicia el Timer con el comando start.

StartTimer(, ,app), **RunTimer(, ,app)** y **EndTimer(, ,app)** son las funciones que ejecuta el objeto Timer. StarTimer y RunTimer son prácticamente iguales, con la diferencia de que StartTimer no añade 10 segundos a la Propiedad TimeSim (lo cual RunTimer sí hace para representar el avance del tiempo en el Reloj de Simulación). Lo que hacen estas dos funciones es: primero actualizar el valor del Campo que representa el Reloj de Simulación con el valor almacenado en la Propiedad TimeSim y a continuación llamar a las funciones que ejecutan la Simulación: primero a SimNav, segundo a ETOCheck y por último a CommCheck. La función EndTimer hace una distinción para reconocer en qué momento de la Simulación se ha parado al Timer, si el valor del botón de estado Pausa está en desactivado, significa que el botón de Pausa no se ha pulsado y la Simulación ha llegado a su fin, por ende, se restauran los objetos y Propiedades correspondientes a los valores que tenían antes de iniciar la Simulación y se llama a la función plotEditorgraph para limpiar

de posibles aviones que se hayan mantenido en la Pantalla de Radar, por último se declara la Propiedad SimTimer que contenía al Timer como matriz vacía, de manera que se elimina el Timer (esto último se hace también si se ha pulsado el botón de Pausa).

PauseSimButtonValueChanged es la función de respuesta asociada al botón de estado Pause, que pausa la Simulación. Al igual que con el botón Play esta función solo funciona cuando el botón pasa de estar Desactivado a estar Activado, y lo que hace es desactivar el botón de Play y habilitarlo junto con el botón Salir al mismo tiempo que se inhabilita a sí mismo. Por último, fuerza la parada del Timer, activando por ende la función EndTimer.

Con esto se han terminado de describir en detalle las Propiedades, Funciones y Funciones de Respuesta o Callbacks del código que se ha desarrollado para la Aplicación ATC Maker.

4 Ejemplo de Simulación

En esta sección se va a ver una descripción del Escenario que se ha creado llamado Tutorial, para probar las funcionalidades del Simulador.

4.1 Escenario Tutorial

Este escenario se ha creado para probar todas las funcionalidades posibles tanto del Editor como del Simulador dentro de los márgenes delimitados por el tipo de Tráfico que se Simula, el cual como se ha mencionado anteriormente es Vuelos en Ruta, con cielos claros y sin viento, con disponibilidad de multiradar y en un espacio con RVSM. Por ello se ha creado el Sector que se muestra en la **Figura 4.1**.

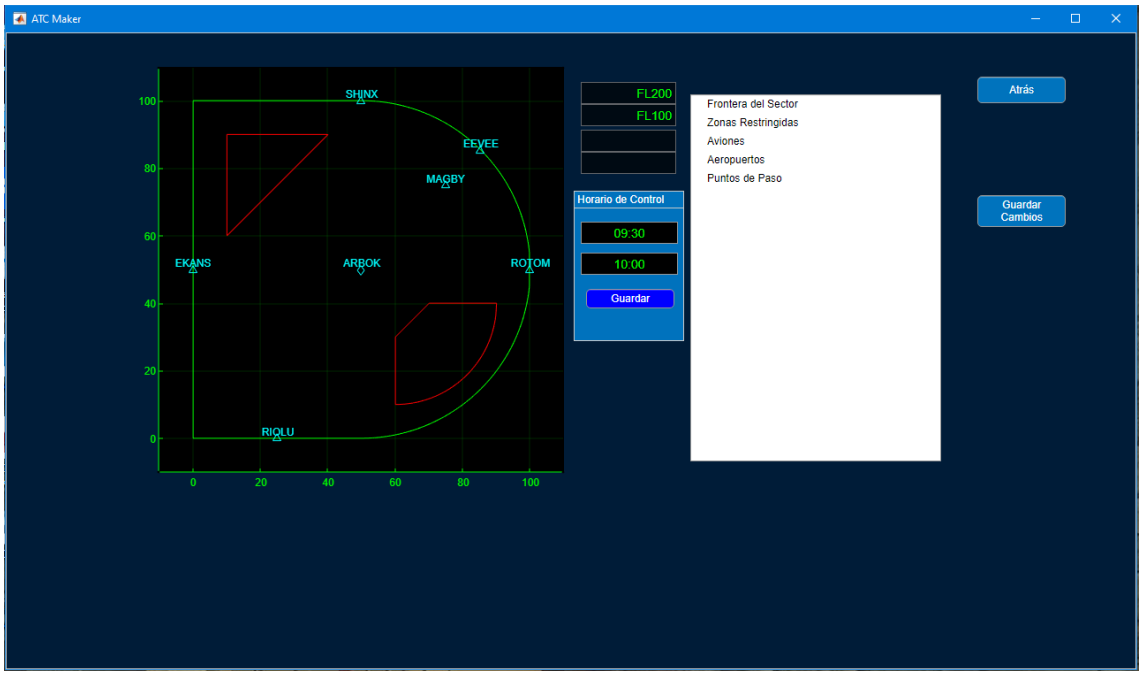

Figura 4.1 Sector en el Editor.

Como se puede ver, la Frontera tiene una Forma de D mayúscula, demostrando así las capacidades de crear una Frontera con líneas rectas y curvas, y con límites verticales entre los Niveles de Vuelo FL100 y FL200. Dentro del Sector se tienen dos Zonas Restringidas, una en la esquina superior izquierda, que tiene forma de Triángulo y encaja con la esquina superior izquierda de la Frontera, cuyos límites verticales están entre los Niveles de Vuelo FL100 y FL150. La segunda Zona Restringida se sitúa en la parte inferior derecha, y tiene una parte de su frontera curva, que se alinea con la curva de la Frontera del Sector, los límites verticales de esta comprenden los Niveles de Vuelo entre FL150 y FL200. Adicionalmente se puede ver que la Simulación de este Escenario empieza a las 9:30 y termina a las 10:00, véase, se tiene media hora de Control en el Sector.

Una vez descritos las Zonas del Sector, se puede hablar de los Puntos de Paso, se pueden ver seis Puntos de Paso (representados por un triangulito de color cian) y un Aeropuerto (representado por un rombo de color cian). El Aeropuerto (ARBOK) se sitúa en el centro del Sector y no se le ha añadido ninguna Pista dado que no sería relevante para lo que se va a Simular. De entre los Puntos de Paso solo hay uno que no es Fronterizo (MAGBY), que sirve como distribuidor entre los tres Puntos Fronterizos que hay en la curva (SHINX, EEVEE y ROTOM) y el Aeropuerto. Luego se tienen dos Puntos Fronterizos más en las partes rectas de la Frontera al otro lado (EKANS y RIOLU). Cabe mencionar que los nombres elegidos para los Puntos son nombres de Pokémon de cinco letras, pues cumplen el criterio establecido en el Apartado de Desarrollo bastante bien (5LNC[17]), y existe una base de nombres bastante extensa, aunque esto no significa que estos nombres deban ser elegidos de esta forma (buscando nombres de Pokémon de cinco letras), sí que se recomienda por el motivo mencionado anteriormente. Adicionalmente, a continuación se muestra en la **Tabla 4.1** las Frecuencias de Cambio de Sector establecidas para cada punto fronterizo.

Tabla 4.1 Frecuencias de Sectores Fronterizos.

Punto	Frecuencia
RIOLU	123.5 MHz
EKANS	125.5 MHz
SHINX	122 MHz
EEVEE	122 MHz
ROTOM	120 MHz

Como se puede ver, los puntos SHINX e EEVEE forman parte de la Frontera con un mismo Sector Adyacente, demostrado porque la Frecuencia de Cambio de Sector de ambos puntos es la misma.

Figura 4.2 Aviones del Escenario.

Una vez determinados los Puntos de Paso, se pueden ver los Aviones que van a Simularse (**Figura 4.2**). Se tienen 4 Vuelos creados, tres de ellos cuentan con Incidencias, una de cada tipo, y uno que no tiene Incidencia. El primero de la lista en la Figura es el EC-BCN, este avión vuela en el Nivel de Vuelo FL100 a 300 kt, entra en el Sector entorno a las 09:31 (un minuto después del inicio de la Simulación, para que dé tiempo a Simular la Entrada), entrando con Rumbo 0 (véase, hacia el Norte), y sigue la Ruta: RIOLU, ARBOK, MAGBY, EEVEE. Este Vuelo no cuenta con ningún Incidente así que en principio seguirá el Plan de Vuelo sin complicaciones.

El Segundo Vuelo en la lista es el EC-MDR, este avión vuela en el Nivel de Vuelo FL170 a 400 kt, entra en el Sector a las 09:39, con Rumbo de Entrada 90 (hacia el Este), y sigue la Ruta: EKANS, ARBOK, MAGBY, SHINX. Debido a esta configuración de Ruta, se ha decidido introducirle a este Vuelo una incidencia Tipo 2 (Solicitud de Directo), para solicitar un Directo desde ARBOK hacia SHINX, y ahorrarse el tramo de Ruta por MAGBY que da un rodeo poco eficiente. El Directo se solicita Antes de llegar a ARBOK.

El Tercer Vuelo es el EC-SEV, este avión vuela en el Nivel de Vuelo FL160 a 450 kt, entra al Sector a las 09:40, con Rumbo 220 y sigue la Ruta: EEVEE, MAGBY, ARBOK, EKANS. Con este Vuelo se ha decidido probar la Incidencia de Tipo 1 (Cambio de Nivel de Vuelo), para solicitar un Cambio de Nivel de Vuelo a FL175 antes de llegar a ARBOK.

El Cuarto y último Vuelo es el EC-JRZ, este avión vuela en el Nivel de Vuelo FL150 a 500 kt, entra al Sector a las 09:45, con Rumbo 200 y sigue la Ruta: ROTOM, MAGBY, ARBOK, RIOLU. Para este Vuelo se ha decidido probar la Incidencia Tipo 3 (Desvío no Programado), para que realice un Cambio de Rumbo a 225, justo después de haber pasado el Punto ROTOM y virado para dirigirse a MAGBY.

En las **Figuras 4.3** y **4.4** se observa la representación de las Rutas Planeadas por cada Vuelo, así como los Paneles de Incidencias de aquellos Vuelos que cuentan con una de estas. En las **Figuras 4.5**, **4.6**, **4.7** y **4.8**,

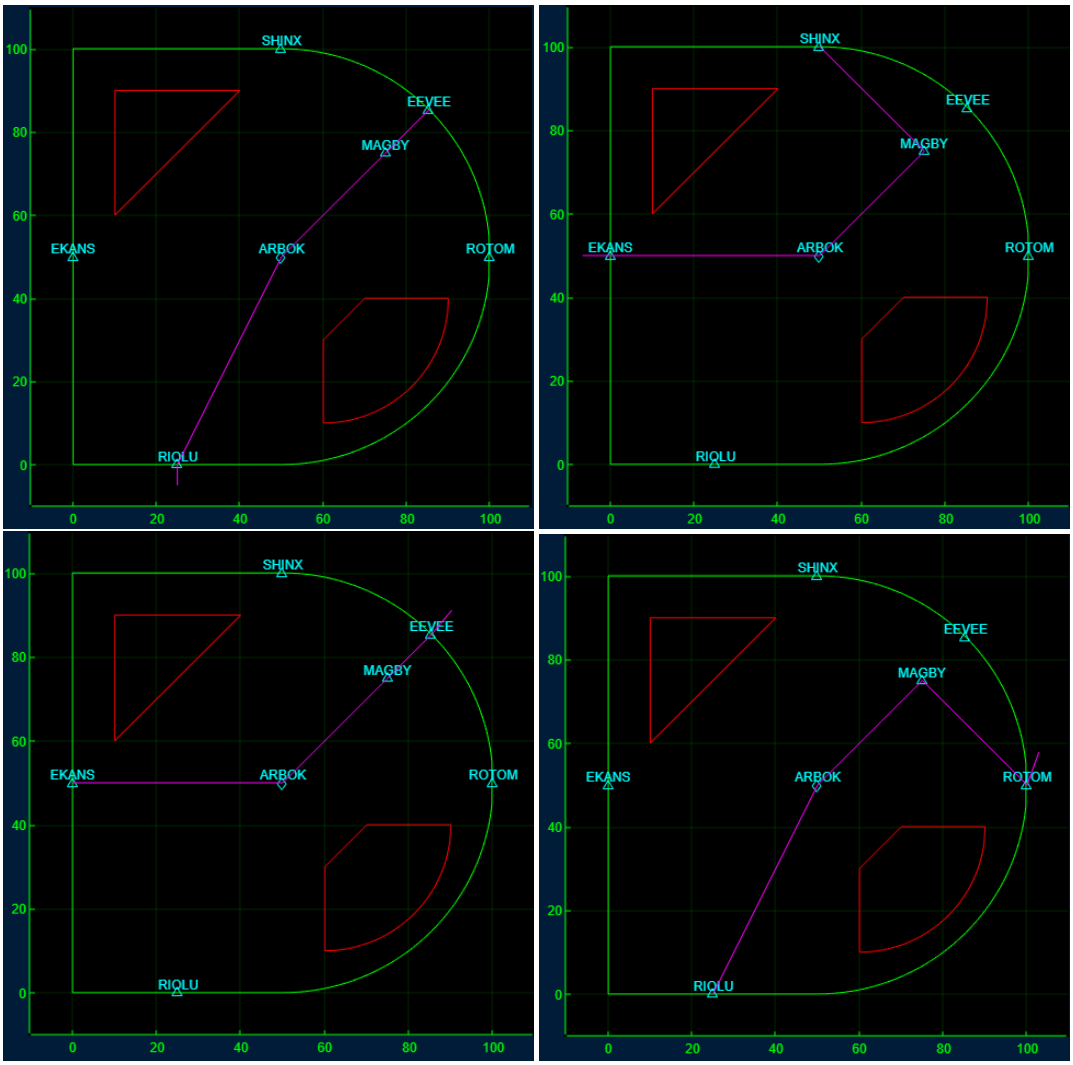

Figura 4.3 Rutas de los Aviones (1) EC-BCN (2) EC-MDR (3) EC-SEV (4) EC-JRZ.

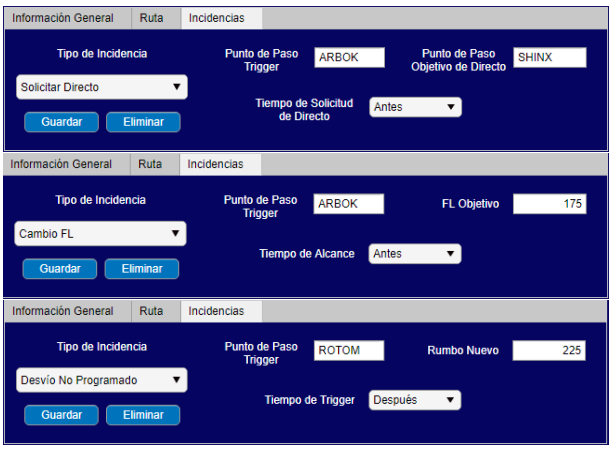

Figura 4.4 Pestañas de Incidencias de los Aviones (1) EC-MDR (2) EC-SEV (3) EC-JRZ.

se observa la pantalla del Simulador tras cargar este escenario Tutorial, Así como unas cuantas Imágenes del dicha Pantalla durante la Simulación que se han considerado relevantes.

Figura 4.5 Escenario en el Simulador.

Figura 4.6 Inicio de Simulación y Entradas de Aviones al Sector.

Figura 4.7 Incidencias durante la Simulación.

Figura 4.8 Salidas de Aviones.

5 Conclusiones

Tras este trabajo se ha demostrado que se ha podido desarrollar una aplicación funcional de Simulación de ATC, así como de Creación de Escenarios de Control para poder Simular. Esta aplicación intenta ser lo más cercana posible a lo que sería la actividad del ATC real, teniendo en cuenta las limitaciones de tiempo y recursos que se esperan de un proyecto de tipo Trabajo de Fin de Grado, además del hecho de que se ha desarrollado por una sola persona y no por un equipo de programadores profesionales. Se puede concluir entonces que se ha cumplido el objetivo, pese a que se deja espacio para ampliar, en cierto sentido se asemeja a la primera versión del Simulador de código abierto estudiado en el Apartado 2, BlueSky ATC, que empezó como un proyecto de un Profesor de la Universidad de Delft.

Cabe destacar que el objetivo de esta Aplicación es poder Introducir a los conceptos básicos del ATC a personas que quizás no tengan tanto conocimiento sobre el campo, de modo que hasta cierto punto las diferencias con respecto a la actividad de ATC real se justifican en este hecho. Sin embargo tampoco se debe olvidar que se ha simplificado el problema y el modelo mucho para ajustarlo a la extensión normal de un Trabajo de Fin de Grado, pero se ha tenido en cuenta este hecho y por ello se han dejado espacios para poder ampliar las funcionalidades de esta aplicación en un desarrollo futuro.

Entre los caminos de ampliación posibles hay unos cuantos obvios. Primero se tendría, modelar el comportamiento de las distintas alertas que se activan al detectar conflictos entre Aeronaves, para lo cual se han dejado vías para introducirse en la Aplicación, pero no se han introducido debido a que cuando fue momento de hacerlo, ATC Maker ya era funcional, e introducirlo solo induciría a una cantidad ingente de fallos que depurar para lo cual no se disponía de mucho más tiempo, de modo que se deja como camino principal para la próxima ampliación. Segundo, introducir un método para poder determinar distintos modelos de aeronave, y encontrar una manera de poder integrar modelos de Simulación que ya existan. Un tercer camino sería el ampliar el abanico de posibilidades para las Incidencias, así como para las Instrucciones que puede dar el Controlador, de modo que se puedan introducir más posibles acciones tanto del Control como de los Pilotos y se acerque todo más a la actividad de ATC real. Una cuarta vía de ampliación sería el introducir la influencia de las condiciones meteorológicas al entorno de Simulación, el cual no se ha tenido en cuenta porque el modelaje de las mismas en MATLAB® sería un TFG en sí mismo. También, y esto se tendría que hacer cada vez que se intentara ampliar este programa, se podría hacer una depuración y optimización del código para que fuera más eficiente, pues siempre existen métodos de hacer algo más eficientemente, así como, por ejemplo, encontrar un método para ampliar los criterios de aceptación de identificadores, en concreto en la creación de Identificadores tanto de Aeronaves, como de Aeropuertos y Radioayudas (incluyendo más tipos de identificadores), o incluso añadir los Identificadores de Radio de ciertas compañías aéreas. Por último, pero no por ello menos importante, se podrían añadir los demás tipos de servicios de ATC a parte del Control de Área, véase, Control de Aproximación y Control de Aeródromo.

Aún con todos estos aspectos en los que se puede mejorar esta aplicación, el resultado al final de este Trabajo es una aplicación funcional y completa a su modo, y que permite tener una experiencia similar a la de un Controlador, salvando las distancias.

A nivel personal este Trabajo me ha aportado muchas cosas, entre ellas la posibilidad de realizar un proyecto de programación de alto nivel. Además de una lección en diligencia y trabajo duro. Por último, me dió la posibilidad de experimentar la satisfacción de un programador cuando ve a alguien usar un Programa o Aplicación creado por él y que este funcione según los parámetros de diseño. Además, aunque faltarían muchas funcionalidades para que se pueda considerar un Simulador Profesional, ese no era el objetivo y éste se ha cumplido, con todas las expectativas.

Apéndice A
Código Completo de ATC Maker

Este es el código completo de la aplicación desarrollada con el Desarrollador de Aplicaciones deMATLAB®

Código A.1 Código Completo de ATC Maker.

```
classdef ATC_Maker < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        ATCMakerApp                  matlab.ui.Figure
        PantalladeInicio             matlab.ui.container.Panel
        EditordeEscenariosButton     matlab.ui.control.Button
        SimuladorButton              matlab.ui.control.Button
        ATCMakerLogo                 matlab.ui.control.Image
        EditordeEscenarios           matlab.ui.container.Panel
        Editorbackbutton             matlab.ui.control.Button
        TreeSector                   matlab.ui.container.Tree
        SectorBorder                 matlab.ui.container.TreeNode
        ZonasRestringidas            matlab.ui.container.TreeNode
        Aviones                      matlab.ui.container.TreeNode
        Aeropuertos                  matlab.ui.container.TreeNode
        Waypoints                    matlab.ui.container.TreeNode
        SectorBorderPanel            matlab.ui.container.Panel
        BorderVertexPanel            matlab.ui.container.Panel
        XnEditFieldLabel             matlab.ui.control.Label
        XnvertEditField              matlab.ui.control.NumericEditField
        YnEditFieldLabel             matlab.ui.control.Label
        YnvertEditField              matlab.ui.control.NumericEditField
        Xn1EditFieldLabel            matlab.ui.control.Label
        Xn1vertEditField             matlab.ui.control.NumericEditField
        Yn1EditFieldLabel            matlab.ui.control.Label
        Yn1vertEditField             matlab.ui.control.NumericEditField
        FronteralneavrticeLabel      matlab.ui.control.Label
        MostrarBorderVButton         matlab.ui.control.Button
        GuardarBorderVButton         matlab.ui.control.Button
        TipodeFronteraButtonGroup    matlab.ui.container.ButtonGroup
        SeleccionaunaButton          matlab.ui.control.ToggleButton
        VerticeButton                matlab.ui.control.ToggleButton
        ArcoButton                   matlab.ui.control.ToggleButton
        AltitudButton                matlab.ui.control.ToggleButton
        BorderTree                   matlab.ui.container.Tree
        EliminarBorderButton         matlab.ui.control.Button
```

```
        BorderArchPanel              matlab.ui.container.Panel
        XnEditFieldLabel_2           matlab.ui.control.Label
        XnarchEditField              matlab.ui.control.NumericEditField
        YnEditFieldLabel_2           matlab.ui.control.Label
        YnarchEditField              matlab.ui.control.NumericEditField
        Xn1EditFieldLabel_2          matlab.ui.control.Label
        Xn1archEditField             matlab.ui.control.NumericEditField
        Yn1EditFieldLabel_2          matlab.ui.control.Label
        Yn1archEditField             matlab.ui.control.NumericEditField
        FronteraarcoLabel            matlab.ui.control.Label
        MostrarBorderAButton         matlab.ui.control.Button
        GuardarBorderAButton         matlab.ui.control.Button
        RadioLabel                   matlab.ui.control.Label
        BARadiusEditField            matlab.ui.control.NumericEditField
        BAConcaveConvexSwitch        matlab.ui.control.Switch
        BACortoLargoSwitch           matlab.ui.control.Switch
        BorderAltitudePanel          matlab.ui.container.Panel
        LmiteInferiorLabel           matlab.ui.control.Label
        ZinfEditField                matlab.ui.control.NumericEditField
        LmiteSuperiorLabel           matlab.ui.control.Label
        ZsupEditField                matlab.ui.control.NumericEditField
        LmitesverticalesLabel        matlab.ui.control.Label
        GuardarBorderAltButton       matlab.ui.control.Button
        FLftinfDropDown              matlab.ui.control.DropDown
        FLftsupDropDown              matlab.ui.control.DropDown
        Editorgraph                  matlab.ui.control.UIAxes
        FLsupEditField               matlab.ui.control.EditField
        FLinfEditField               matlab.ui.control.EditField
        SectorRestricZonePanel       matlab.ui.container.Panel
        RZVertexPanel                matlab.ui.container.Panel
        XnEditFieldLabel_3           matlab.ui.control.Label
        RZXnvertEditField            matlab.ui.control.NumericEditField
        YnEditFieldLabel_3           matlab.ui.control.Label
        RZYnvertEditField            matlab.ui.control.NumericEditField
        Xn1EditFieldLabel_3          matlab.ui.control.Label
        RZXn1vertEditField           matlab.ui.control.NumericEditField
        Yn1EditFieldLabel_3          matlab.ui.control.Label
        RZYn1vertEditField           matlab.ui.control.NumericEditField
        ZonaRestringidalneavrticeLabel matlab.ui.control.Label
        MostrarRZVButton             matlab.ui.control.Button
        GuardarRZVButton             matlab.ui.control.Button
        TipodeFronteraRZButtonGroup  matlab.ui.container.ButtonGroup
        SeleccionaunaButtonRZ        matlab.ui.control.ToggleButton
        VerticeButtonRZ              matlab.ui.control.ToggleButton
        ArcoButtonRZ                 matlab.ui.control.ToggleButton
        AltitudButtonRZ              matlab.ui.control.ToggleButton
        RZTree                       matlab.ui.container.Tree
        EliminarRZButton             matlab.ui.control.Button
        RZArchPanel                  matlab.ui.container.Panel
        XnEditFieldLabel_4           matlab.ui.control.Label
        RZXnarchEditField            matlab.ui.control.NumericEditField
        YnEditFieldLabel_4           matlab.ui.control.Label
        RZYnarchEditField            matlab.ui.control.NumericEditField
        Xn1EditFieldLabel_4          matlab.ui.control.Label
        RZXn1archEditField           matlab.ui.control.NumericEditField
        Yn1EditFieldLabel_4          matlab.ui.control.Label
        RZYn1archEditField           matlab.ui.control.NumericEditField
```

```
95          ZonaRestringidaarcoLabel     matlab.ui.control.Label
96          MostrarRZAButton             matlab.ui.control.Button
97          GuardarRZAButton             matlab.ui.control.Button
98          RadioLabel_2                 matlab.ui.control.Label
99          RZARadiusEditField           matlab.ui.control.NumericEditField
100         RZAConcaveConvexSwitch       matlab.ui.control.Switch
101         RZACortoLargoSwitch          matlab.ui.control.Switch
102         RZAltitudePanel              matlab.ui.container.Panel
103         LmiteInferiorLabel_2         matlab.ui.control.Label
104         RZZinfEditField              matlab.ui.control.NumericEditField
105         LmiteSuperiorLabel_2         matlab.ui.control.Label
106         RZZsupEditField              matlab.ui.control.NumericEditField
107         LmitesverticalesLabelRZ      matlab.ui.control.Label
108         GuardarRZAltButton           matlab.ui.control.Button
109         FLftinfDropDownRZ            matlab.ui.control.DropDown
110         FLftsupDropDownRZ            matlab.ui.control.DropDown
111         NuevaZonaRestringidaButton   matlab.ui.control.Button
112         RZFLsupEditField             matlab.ui.control.EditField
113         RZFLinfEditField             matlab.ui.control.EditField
114         SectorAirportPanel           matlab.ui.container.Panel
115         XLabel                       matlab.ui.control.Label
116         XairportEditField            matlab.ui.control.NumericEditField
117         YLabel                       matlab.ui.control.Label
118         YairportEditField            matlab.ui.control.NumericEditField
119         AeropuertosdelSectorLabel    matlab.ui.control.Label
120         NombreEditFieldLabel         matlab.ui.control.Label
121         NombreAirportEditField       matlab.ui.control.EditField
122         AirportTree                  matlab.ui.container.Tree
123         GuardarAirportButton         matlab.ui.control.Button
124         AddRWYButton                 matlab.ui.control.Button
125         EliminarAirportButton        matlab.ui.control.Button
126         RWYPanel                     matlab.ui.container.Panel
127         PistadeaterrizajeLabel       matlab.ui.control.Label
128         RWYNewEditSwitch             matlab.ui.control.Switch
129         NewRWYPanel                  matlab.ui.container.Panel
130         RWYSpinnerLabel              matlab.ui.control.Label
131         NewRWYSpinner                matlab.ui.control.Spinner
132         GuardarNewRWYButton          matlab.ui.control.Button
133         EditRWYPanel                 matlab.ui.container.Panel
134         RWYDropDownLabel             matlab.ui.control.Label
135         RWYEditDropDown              matlab.ui.control.DropDown
136         RWYEditSpinner               matlab.ui.control.Spinner
137         GuardarRWYEditButton         matlab.ui.control.Button
138         EliminarRWYEditButton        matlab.ui.control.Button
139         SectorWaypointPanel          matlab.ui.container.Panel
140         XLabel_2                     matlab.ui.control.Label
141         XwaypointEditField           matlab.ui.control.NumericEditField
142         YLabel_2                     matlab.ui.control.Label
143         YwaypointEditField           matlab.ui.control.NumericEditField
144         PuntosdePasodelSectorLabel   matlab.ui.control.Label
145         NombreEditField_2Label       matlab.ui.control.Label
146         NombreWPEditField            matlab.ui.control.EditField
147         WaypointTree                 matlab.ui.container.Tree
148         GuardarWaypointButton        matlab.ui.control.Button
149         EliminarWaypointButton       matlab.ui.control.Button
150         WaypointFronterizoButton     matlab.ui.control.Button
151         WaypointFronterizoFreqPanel  matlab.ui.container.Panel
```

```
        FrecuenciadelSectorFronterizoLabel matlab.ui.control.Label
        FreqMHzEditFieldLabel        matlab.ui.control.Label
        FreqMHzEditField             matlab.ui.control.NumericEditField
        GuardarWPFreqButton          matlab.ui.control.Button
        SectorAirplanePanel          matlab.ui.container.Panel
        AirplaneTabGroup             matlab.ui.container.TabGroup
        IGTab                        matlab.ui.container.Tab
        NiveldeVueloEditFieldLabel   matlab.ui.control.Label
        NiveldeVueloEditField        matlab.ui.control.NumericEditField
        MatrculadelaaeronaveEditFieldLabel matlab.ui.control.Label
        MatrculadelaaeronaveEditField matlab.ui.control.EditField
        VelocidaddeVueloktEditFieldLabel matlab.ui.control.Label
        VelocidaddeVueloktEditField  matlab.ui.control.NumericEditField
        ETOEditFieldLabel            matlab.ui.control.Label
        ETOEditField                 matlab.ui.control.EditField
        RumbodeEntradaEditFieldLabel matlab.ui.control.Label
        EHDGEditField                matlab.ui.control.NumericEditField
        RouteTab                     matlab.ui.container.Tab
        RouteTree                    matlab.ui.container.Tree
        PuntodePasodelaRutaEditFieldLabel matlab.ui.control.Label
        RouteWPEditField             matlab.ui.control.EditField
        GuardarRouteWPButton         matlab.ui.control.Button
        EliminarRouteWPButton        matlab.ui.control.Button
        IncidenciasTab               matlab.ui.container.Tab
        TipodeIncidenciaDropDownLabel matlab.ui.control.Label
        TipodeIncidenciaDropDown     matlab.ui.control.DropDown
        CambioFLPanel                matlab.ui.container.Panel
        PuntodePasoTriggerEditFieldLabel matlab.ui.control.Label
        WaypointTriggerCFLEditField  matlab.ui.control.EditField
        FLObjetivoEditFieldLabel     matlab.ui.control.Label
        FLObjetivoEditField          matlab.ui.control.NumericEditField
        TiempodeAlcanceDropDownLabel matlab.ui.control.Label
        CFLDropDown                  matlab.ui.control.DropDown
        GuardarIncidenciaButton      matlab.ui.control.Button
        SolicitarDirectoPanel        matlab.ui.container.Panel
        PuntodePasoTriggerEditField_2Label matlab.ui.control.Label
        WaypointTriggerSDEditField   matlab.ui.control.EditField
        TiempodeSolicituddeDirectoLabel matlab.ui.control.Label
        SDDropDown                   matlab.ui.control.DropDown
        PuntodePasoObjetivodeDirectoEditFieldLabel matlab.ui.control.Label
        WPDirectObjetivoEditField    matlab.ui.control.EditField
        CRNPPanel                    matlab.ui.container.Panel
        PuntodePasoTriggerEditFieldLabel_2 matlab.ui.control.Label
        WaypointTriggerCRNPEditField matlab.ui.control.EditField
        RumboNuevoEditFieldLabel     matlab.ui.control.Label
        RumboNuevoEditField          matlab.ui.control.NumericEditField
        TiempodeTriggerLabel         matlab.ui.control.Label
        CRNPDropDown                 matlab.ui.control.DropDown
        EliminarIncidenciaButton     matlab.ui.control.Button
        AirplaneTree                 matlab.ui.container.Tree
        GuardarAirplaneButton        matlab.ui.control.Button
        EliminarAirplaneButton       matlab.ui.control.Button
        GuardarSectorButton          matlab.ui.control.Button
        HorariodeControlPanel        matlab.ui.container.Panel
        HorainicioEditField          matlab.ui.control.EditField
        HorafinalEditField           matlab.ui.control.EditField
        GuardarHorarioButton         matlab.ui.control.Button
```

```
        Simulador                       matlab.ui.container.Panel
        FlightStripRackPanel            matlab.ui.container.Panel
        FLSRButton                      matlab.ui.control.StateButton
        FLSRCSEF                        matlab.ui.control.EditField
        FLSRFLEF                        matlab.ui.control.EditField
        FLSRSPDEF                       matlab.ui.control.EditField
        FLSRLamp                        matlab.ui.control.Lamp
        FlightStripPanel                matlab.ui.container.Panel
        FlightRoutePanel                matlab.ui.container.Panel
        WPRNEF                          matlab.ui.control.EditField
        TPWPEF                          matlab.ui.control.EditField
        FSCallsignEditField             matlab.ui.control.EditField
        FSFLEditField                   matlab.ui.control.EditField
        FSSPDEditField                  matlab.ui.control.EditField
        FSETOEditField                  matlab.ui.control.EditField
        Simuladorgraph                  matlab.ui.control.UIAxes
        SalirSimButton                  matlab.ui.control.Button
        PlaySimButton                   matlab.ui.control.StateButton
        PauseSimButton                  matlab.ui.control.StateButton
        TimeSimEditField                matlab.ui.control.EditField
        TimeEndEditField                matlab.ui.control.EditField
        CommsTextArea                   matlab.ui.control.TextArea
        CommsInputPanel                 matlab.ui.container.Panel
        InputmsgTextArea                matlab.ui.control.TextArea
        SendButton                      matlab.ui.control.Button
        DropDown                        matlab.ui.control.DropDown
        AutorizarEntradaButton          matlab.ui.control.Button
        AutorizarCambioFLButton         matlab.ui.control.Button
        NoAutorizarCambioFLButton       matlab.ui.control.Button
        AutorizarDirectoButton          matlab.ui.control.Button
        NoAutorizarDirectoButton        matlab.ui.control.Button
        CambioaRumboAutorizadoButton matlab.ui.control.Button
        CambioFrecuenciaButton          matlab.ui.control.Button
        GeneralConflictLamp             matlab.ui.control.Lamp
        Editorseleccion                 matlab.ui.container.Panel
        Editorseleccionbackbutton       matlab.ui.control.Button
        TreeEditor                      matlab.ui.container.Tree
        NuevoEscenario                  matlab.ui.container.Panel
        NombredelescenarioEditFieldLabel matlab.ui.control.Label
        NombredelnuevoescenarioEditField matlab.ui.control.EditField
        CrearButton                     matlab.ui.control.Button
        AtrsButton                      matlab.ui.control.Button
        NuevoEscenarioButton            matlab.ui.control.Button
        RenombrarEscenarioButton        matlab.ui.control.Button
        RenombrarEscenario              matlab.ui.container.Panel
        NombredelescenarioEditFieldLabel_2 matlab.ui.control.Label
        NombrenuevodelnuevoescenarioEditField matlab.ui.control.EditField
        RenombrarButton                 matlab.ui.control.Button
        AtrsButton_2                    matlab.ui.control.Button
        EliminarEscenarioButton         matlab.ui.control.Button
        CargarEscenarioButton           matlab.ui.control.Button
        Simuladorseleccion              matlab.ui.container.Panel
        Simuladorseleccionbackcutton matlab.ui.control.Button
        TreeSimulador                   matlab.ui.container.Tree
        CargarEscenarioSimButton        matlab.ui.control.Button
    end

```

```
266
267     properties (Access = private)
268         SectorEditor;
269         SectorSimulador;
270         S;
271         NombreNuevoEscenario;
272         pmat = uint16(char('.mat'));
273         libapp = uint16(char('libapp\'));
274         SectorEditorSelec;
275         SectorSimuladorSelec;
276         Vert;
277         Arch;
278         Alt;
279         Airport;
280         Waypoint;
281         Airplane;
282         RouteWP;
283         Incident;
284         TimeSector;
285         ATCCommLine = {{{'Recibido '},{', Autorizado'}};...
286             {{'Recibido '},{', Autorizado Cambio de Nivel de Vuelo a '},{' de
                    llegar a '}};...
287             {{'Negativo '},{', Manténgase en el Nivel de Vuelo Autorizado'}};...
288             {{'Recibido '},{', Autorizado Directo hacia '},{' de llegar a '}};...

289             {{'Negativo '},{', Siga con el Plan de Vuelo Establecido'}};...
290             {{' aquí Control, Detectado desvío del rumbo no autorizado, vuelva
                    al Plan de Vuelo Establecido Tomando el Rumbo '},{' para
                    Incorporarse al Radial '},{' de '}};...
291             {{'Recibido '},{', Cambie a Frecuencia '},{' para el Cambio de
                    Sector'}}};
292         AirmenCommLine = {{{'Control aquí '},{' solicitamos Autorización para
                realizar el Plan de Vuelo Establecido en el Sector'}};...
293             {{'Control aqui '},{' solicitamos Autorización para realizar un
                    Cambio de Nivel de Vuelo a '},{' de llegar a '}};...
294             {{'Control aquí '},{' solicitamos Autorización para realizar un
                    Directo hacia '},{' de llegar a '}};...
295             {{'Recibido Control, seguiremos Instrucciones'}};...
296             {{'Control aquí '},{' solicitamos Frecuencia de Radio para el Cambio
                     de Sector'}}};
297         AirplaneSim;
298         TimeSim = duration(0,0,0);
299         FlightStrip;
300         Comminput;
301         SimTimer = [];
302         SimData = [];
303     end
304
305     methods (Access = public)
306
307         function [] = plotEditorgraph(app)
308             if ~isempty(app.SectorEditor.Sector.Border)
309                 Xo = cell2mat(app.SectorEditor.Sector.Border.X);
310                 Yo = cell2mat(app.SectorEditor.Sector.Border.Y);
311                 plot(app.Editorgraph,Xo,Yo,'-g')
312                 hold(app.Editorgraph,"on")
313                 if ~isempty(Xo)
```

```
314                     Xomin = min(Xo);
315                     Xomax = max(Xo);
316                     Yomin = min(Yo);
317                     Yomax = max(Yo);
318                     d = max(0.1*(Xomax-Xomin),0.1*(Yomax-Yomin));
319                     Xb = [Xomin-d, Xomax+d, Xomax+d, Xomin-d];
320                     Yb = [Yomin-d, Yomin-d, Yomax+d, Yomax+d];
321                     plot(app.Editorgraph,Xb,Yb,'.k');
322                 end
323             end
324             if ~isempty(app.SectorEditor.Sector.RestricZones)
325                 for k = 1:size(app.SectorEditor.Sector.RestricZones,2)
326                     Xz = cell2mat(app.SectorEditor.Sector.RestricZones(k).X);
327                     Yz = cell2mat(app.SectorEditor.Sector.RestricZones(k).Y);
328                     plot(app.Editorgraph,Xz,Yz,'-r')
329                     hold(app.Editorgraph,"on")
330                 end
331             end
332             if ~isempty(app.SectorEditor.Sector.Airport)
333                 hold(app.Editorgraph,"on")
334                 for i = 1:size(app.SectorEditor.Sector.Airport,2)
335                     Xa = app.SectorEditor.Sector.Airport(i).X;
336                     Ya = app.SectorEditor.Sector.Airport(i).Y;
337                     Namea = char(app.SectorEditor.Sector.Airport(i).Name);
338                     plot(app.Editorgraph,Xa,Ya,'dc')
339                     text(app.Editorgraph,Xa,Ya,Namea,'Color',[0 1 1],'
                            HorizontalAlignment','center','VerticalAlignment','bottom
                            ')
340                 end
341             end
342             if ~isempty(app.SectorEditor.Sector.Waypoint)
343                 hold(app.Editorgraph,"on")
344                 for i = 1:size(app.SectorEditor.Sector.Waypoint,2)
345                     Xw = app.SectorEditor.Sector.Waypoint(i).X;
346                     Yw = app.SectorEditor.Sector.Waypoint(i).Y;
347                     Namew = app.SectorEditor.Sector.Waypoint(i).Name;
348                     plot(app.Editorgraph,Xw,Yw,'^c')
349                     text(app.Editorgraph,Xw,Yw,Namew,'Color',[0 1 1],'
                            HorizontalAlignment','center','VerticalAlignment','bottom
                            ')
350                 end
351             end
352             app.Editorgraph.DataAspectRatio = [1 1 1];
353         end
354 
355         function [] = resetBorderXYeditfieldvert(app)
356             app.XnvertEditField.Value = 0;
357             app.YnvertEditField.Value = 0;
358             app.Xn1vertEditField.Value = 0;
359             app.Yn1vertEditField.Value = 0;
360         end
361 
362         function [] = updateBorderVert(app)
363             app.Vert.orig(1,1) = app.XnvertEditField.Value;
364             app.Vert.orig(1,2) = app.YnvertEditField.Value;
365             app.Vert.dest(1,1) = app.Xn1vertEditField.Value;
366             app.Vert.dest(1,2) = app.Yn1vertEditField.Value;
```

```
367         end
368
369         function [] = LoadScenario(app)
370             plotEditorgraph(app)
371             hold(app.Editorgraph,"off")
372             resetBorderXYeditfieldvert(app)
373             updateBorderVert(app)
374             clearSelectedNodesEditor(app);
375             if ~isempty(app.SectorEditor.Sector.Border)
376                 Tx = app.SectorEditor.Sector.Border.X;
377                 for i = 1:size(Tx,2)
378                     CharNum = dec2char(app,i);
379                     if size(app.SectorEditor.Sector.Border.X{i},2) > 1
380                         Typenode = char('Arco ');
381                     else
382                         Typenode = char('Vértice ');
383                     end
384                     uitreenode(app.BorderTree,"NodeData",i,"Text",char([Typenode,
                            CharNum]));
385                 end
386                 if ~isempty(app.SectorEditor.Sector.Border.Z)
387                     app.Alt.inf = app.SectorEditor.Sector.Border.Z(2);
388                     app.Alt.sup = app.SectorEditor.Sector.Border.Z(1);
389                     GuardarBorderAltButtonPushed(app);
390                 end
391             end
392             if ~isempty(app.SectorEditor.Sector.RestricZones)
393                 RTx = app.SectorEditor.Sector.RestricZones;
394                 for i = 1:length(RTx)
395                     CharNum = dec2char(app,i);
396                     uitreenode(app.RZTree,'Tag','Node','Text',char(['Zona
                            Restringida ',CharNum]),'NodeData',i);
397                     if ~isempty(app.SectorEditor.Sector.RestricZones(i).X)
398                         for j = 1:length(RTx(i).X)
399                             jchar = dec2char(app,j);
400                             if length(app.SectorEditor.Sector.RestricZones(i).X{j
                                    }) > 1
401                                 Typenode = 'Arco ';
402                             else
403                                 Typenode = 'Vértice ';
404                             end
405                             uitreenode(app.RZTree.Children(i),'NodeData',j,'Text',
                                    char([Typenode,jchar]));
406                         end
407                     end
408                 end
409             end
410             if ~isempty(app.SectorEditor.Sector.Airplane)
411                 for i = 1:length(app.SectorEditor.Sector.Airplane)
412                     ac = app.SectorEditor.Sector.Airplane(i);
413                     uitreenode(app.AirplaneTree,'NodeData',i,'Text',ac.Callsign);
414                 end
415             end
416             if ~isempty(app.SectorEditor.Sector.Airport)
417                 Ap = app.SectorEditor.Sector.Airport;
418                 for i = 1:length(Ap)
419                     Xchar = dec2char(app,Ap(i).X);
```

```
420                     Ychar = dec2char(app,Ap(i).Y);
421                     uitreenode(app.AirportTree,'NodeData',i,'Text',Ap(i).Name);
422                     uitreenode(app.AirportTree.Children(i),'Text',char(['[',Xchar
                            ,',',Ychar,']']));
423                 end
424             end
425             if ~isempty(app.SectorEditor.Sector.Waypoint)
426                 Wp = app.SectorEditor.Sector.Waypoint;
427                 for i = 1:length(Wp)
428                     Xchar = dec2char(app,Wp(i).X);
429                     Ychar = dec2char(app,Wp(i).Y);
430                     uitreenode(app.WaypointTree,'NodeData',i,'Text',Wp(i).Name);
431                     uitreenode(app.WaypointTree.Children(i),'Text',char(['[',
                            Xchar,',',Ychar,']']));
432                 end
433             end
434             if ~isempty(app.SectorEditor.Sector.Time)
435                 H = app.SectorEditor.Sector.Time;
436                 if ~isempty(H.init)
437                     app.HorainicioEditField.Value = char(H.init,'hh:mm');
438                 end
439                 if ~isempty(H.fin)
440                     app.HorafinalEditField.Value = char(H.fin,'hh:mm');
441                 end
442             end
443         end
444
445         function CharNum = dec2char(app,Num)
446             Mil = [];
447             Cent = [];
448             Dec = [];
449             app.Xn1EditFieldLabel;
450             if Num >= 1000
451                 Mil = floor(Num/1000);
452                 Num = Num - Mil*1000;
453             end
454             if Num >= 100
455                 Cent = floor(Num/100);
456                 Num = Num - Cent*100;
457             elseif ~isempty(Mil)
458                 Cent = 0;
459             end
460             if Num >= 10
461                 Dec = floor(Num/10);
462                 Num = Num - Dec*10;
463             elseif ~isempty(Cent)
464                 Dec = 0;
465             end
466             Unit = Num;
467             if ~isempty(Mil)
468                 Mil = Mil + 48;
469             end
470             if ~isempty(Cent)
471                 Cent = Cent + 48;
472             end
473             if ~isempty(Dec)
474                 Dec = Dec + 48;
```

```
475             end
476             Unit = Unit + 48;
477             CharNum = char([Mil,Cent,Dec,Unit]);
478         end
479
480         function [] = clearSector(app)
481             app.SectorEditor.Sector.Name = {};
482             app.SectorEditor.Sector.Time = [];
483             app.SectorEditor.Sector.Border = [];
484             app.SectorEditor.Sector.RestricZones = [];
485             app.SectorEditor.Sector.Airplane = [];
486             app.SectorEditor.Sector.Airport = [];
487             app.SectorEditor.Sector.Waypoint = [];
488         end
489
490         function [] = clearSelectedNodesEditor(app)
491             app.BorderTree.SelectedNodes = [];
492             BorderTreeSelectionChanged(app);
493             app.TreeEditor.SelectedNodes = [];
494             app.RZTree.SelectedNodes = [];
495             RZTreeSelectionChanged(app);
496             app.AirportTree.SelectedNodes = [];
497             AirportTreeSelectionChanged(app);
498             app.WaypointTree.SelectedNodes = [];
499             WaypointTreeSelectionChanged(app);
500             app.AirplaneTree.SelectedNodes = [];
501             AirplaneTreeSelectionChanged(app);
502         end
503
504         function [] = clearTreesEditor(app)
505             delete(app.BorderTree.Children);
506             delete(app.RZTree.Children);
507             delete(app.AirportTree.Children);
508             delete(app.WaypointTree.Children);
509             delete(app.AirplaneTree.Children);
510             delete(app.RouteTree.Children);
511         end
512
513         function [] = clearButtonGroupsEditor(app)
514             app.SeleccionaunaButton.Value = true;
515             app.TipodeFronteraButtonGroupSelectionChanged(app);
516             app.SeleccionaunaButtonRZ.Value = true;
517             app.TipodeFronteraRZButtonGroupSelectionChanged(app);
518         end
519
520         function I = encuentraI(app,Arco,A,B)
521             AB = B - A;
522             U = A + 0.5*AB;
523             pAB = [AB(2),-AB(1)];
524             k = 1;
525             H = zeros(2001,2);
526             for j = -1:0.001:1
527                 H(k,:) = U + j*pAB;
528                 k = k + 1;
529             end
530             s = floor(size(Arco,1)/2);
531             arco = Arco(s-ceil(s*0.01):s+ceil(s*0.01),:);
```

```
            q = 1;
            P = cell(max(size(arco,1),size(H,1)),1);
            for k = 1:size(arco,1)
                d = zeros(size(H,1),1);
                for j = 1:size(H,1)
                    d(j) = norm(arco(k,:)-H(j,:));
                end
                [t,~] = find(abs(d)<0.01);
                if ~isempty(t)
                    P{q} = [d(t), t, k*ones(size(t))];
                    q = q+1;
                end
            end
            T = cell(q-1,1);
            for i = 1:q-1
                T{i} = P{i};
            end
            P = T;
            P = cell2mat(P);
            [~,p] = min(P(:,1));
            I = arco(P(p,3),:);
            app.Arch;
        end

        function Centro = encuentracentro(app,A,B,I,R)
            AB = B - A;
            U = A + 0.5*AB;
            IU = U - I;
            Centro = I + (R/norm(IU))*IU;
            app.Arch;
        end

        function R = CalcRad(app,X,Y)
            A = [X(1),Y(1)];
            B = [X(end),Y(end)];
            U = A + 0.5*(B-A);
            Arco = [X',Y'];
            I = encuentraI(app,Arco,A,B);
            C = norm(I-A);
            CT = norm(I-U);
            calpha = CT/C;
            alpha = acos(calpha);
            gam = 2*pi - 2*alpha;
            cgam = cos(gam);
            f = sqrt(2*(1-cgam));
            R = C/f;
            app.Arch;
        end

        function [Value,len] = ConcaveConvexShortLongcheck(app,X,Y,R)
            Arco = [X',Y'];
            A = Arco(1,:);
            B = Arco(end,:);
            I = encuentraI(app,Arco,A,B);
            U = A + 0.5*(B-A);
            if B(1) < A(1) || B(1) > A(1)
                if U(2) <= I(2)
```

```
                Value = '^';
            elseif U(2) > I(2)
                Value = 'v';
            end
        elseif B(1) == A(1)
            if I(1) >= U(1)
                Value = '^';
            elseif I(1) < U(1)
                Value = 'v';
            end
        end
        if norm(U-I) <= R
            len = 'Corto';
        elseif norm(U-I) > R
            len = 'Largo';
        end
        app.Arch;
    end


    function Centro = Calccentre(app,Arco)
        A = Arco.orig;
        B = Arco.dest;
        R = Arco.rad;
        U = A + 0.5*(B-A);
        AB = A(1)^2 - B(1)^2 + A(2)^2 - B(2)^2;
        ABx = A(1) - B(1);
        ABy = A(2) - B(2);
        if ABy ~= 0
            s = A(1)^2 + A(2)^2 - R^2;
            a = 4*(ABy^2 + ABx^2);
            b = 8*ABy*ABx*A(2) - 8*A(1)*ABy^2 - 4*ABx*AB;
            c = AB^2 + 4*s*ABy^2 - 4*ABy*AB*A(2);
            Cx(1,1) = (-b + sqrt(b^2 - 4*a*c))/(2*a);
            Cx(2,1) = (-b - sqrt(b^2 - 4*a*c))/(2*a);
            Cy = (AB/(2*ABy))*ones(2,1) - (ABx/ABy)*Cx;
            Cpos = [Cx(1) Cy(1)];
            Cneg = [Cx(2) Cy(2)];
            C = [Cpos; Cneg];
            if ABx ~= 0
                [~,nsup] = max(C(:,2));
                [~,ninf] = min(C(:,2));
                if Arco.dir(6) == 'v'
                    if Arco.len(1) == 'L'
                        Centro = C(nsup,:);
                    elseif Arco.len(1) == 'C'
                        Centro = C(ninf,:);
                    end
                elseif Arco.dir(6) == 'x'
                    if Arco.len(1) == 'L'
                        Centro = C(ninf,:);
                    elseif Arco.len(1) == 'C'
                        Centro = C(nsup,:);
                    end
                end
            elseif ABx == 0
                [~,nder] = max(C(:,1));
```

```
646                     [~,nizq] = min(C(:,1));
647                     if Arco.dir(6) == 'v'
648                         if Arco.len(1) == 'L'
649                             Centro = C(nder,:);
650                         elseif Arco.len(1) == 'C'
651                             Centro = C(nizq,:);
652                         end
653                     elseif Arco.dir(6) == 'x'
654                         if Arco.len(1) == 'L'
655                             Centro = C(nder,:);
656                         elseif Arco.len(1) == 'C'
657                             Centro = C(nizq,:);
658                         end
659                     end
660                 end
661             elseif ABy == 0
662                 t = norm(U-A);
663                 gam = acos(t/R);
664                 r = R*sin(gam);
665                 if Arco.dir(6) == 'v'
666                     if Arco.len(1) == 'L'
667                         Centro = U + r*[0,1];
668                     elseif Arco.len(1) == 'C'
669                         Centro = U - r*[0,1];
670                     end
671                 elseif Arco.dir(6) == 'x'
672                     if Arco.len(1) == 'L'
673                         Centro = U - r*[0,1];
674                     elseif Arco.len(1) == 'C'
675                         Centro = U + r*[0,1];
676                     end
677                 end
678             end
679             app.Arch;
680         end
681
682         function [X,Y] = Calcarch(app,Arco,Centro)
683             A = Arco.orig;
684             B = Arco.dest;
685             R = Arco.rad;
686             thetaA = Calctheta(app,A,Centro,R);
687             thetaB = Calctheta(app,B,Centro,R);
688             if  Arco.dir(6) == 'v'
689                 if thetaA > thetaB
690                     if A(1) > B(1)
691                         th = [thetaA:0.0001:2*pi,0:0.0001:thetaB+0.0001];
692                     elseif A(1) <= B(1)
693                         th = thetaA:-0.0001:thetaB-0.0001;
694                     end
695                 elseif thetaA < thetaB
696                     if A(1) > B(1)
697                         th = thetaA:0.0001:thetaB+0.0001;
698                     elseif A(1) <= B(1)
699                         th = [thetaA:-0.0001:0,2*pi:-0.0001:thetaB-0.0001];
700                     end
701                 end
702             elseif Arco.dir(6) == 'x'
```

```
                if thetaA > thetaB
                    if A(1) > B(1)
                        th = thetaA:-0.0001:thetaB-0.0001;
                    elseif A(1) <= B(1)
                        th = [thetaA:0.0001:2*pi,0:0.0001:thetaB+0.0001];
                    end
                elseif thetaA < thetaB
                    if A(1) > B(1)
                        th = [thetaA:-0.0001:0,2*pi:-0.0001:thetaB-0.0001];
                    elseif A(1) <= B(1)
                        th = thetaA:0.0001:thetaB+0.0001;
                    end
                end
            end
            X = Centro(1)*ones(size(th)) + R*cos(th);
            Y = Centro(2)*ones(size(th)) + R*sin(th);
        end

        function theta = Calctheta(~,P,C,R)
            t = acos((P(1)-C(1))/R);
            if P(2) >= C(2)
                theta = t;
            elseif P(2) < C(2)
                theta = 2*pi - t;
            end
        end

        function [] = resetBordereditfieldarch(app)
            app.XnarchEditField.Value = 0;
            app.YnarchEditField.Value = 0;
            app.Xn1archEditField.Value = 0;
            app.Yn1archEditField.Value = 0;
            app.BARadiusEditField.Value = 0;
            app.BAConcaveConvexSwitch.Value = '^';
            app.BACortoLargoSwitch.Value = 'Corto';
        end

        function [] = updateBorderArch(app)
            app.Arch.orig(1) = app.XnarchEditField.Value;
            app.Arch.orig(2) = app.YnarchEditField.Value;
            app.Arch.dest(1) = app.Xn1archEditField.Value;
            app.Arch.dest(2) = app.Yn1archEditField.Value;
            app.Arch.rad = app.BARadiusEditField.Value;
            BAConcaveConvexSwitchValueChanged(app)
            app.Arch.len = app.BACortoLargoSwitch.Value;
        end

        function FLchar = FLChar(app,Altft)
            FL = dec2char(app,Altft/100);
            if Altft >= 10000
                FLchar = char(['FL',FL]);
            else
                FLchar = char(['FL0',FL]);
            end
        end

        function ftchar = ftChar(app,Altft)
```

```
760             ft = dec2char(app,Altft);
761             ftchar = char([ft,'ft']);
762         end
763 
764         function [] = resetBordereditfieldalt(app)
765             app.ZinfEditField.Value = 0;
766             app.ZsupEditField.Value = 0;
767             app.FLftinfDropDown.Value = '1';
768             app.FLftsupDropDown.Value = '1';
769         end
770 
771         function [] = updateBorderAlt(app)
772             ZinfEditFieldValueChanged(app)
773             ZsupEditFieldValueChanged(app)
774         end
775 
776         function [] = resetRZXYeditfieldvert(app)
777             app.RZXnvertEditField.Value = 0;
778             app.RZYnvertEditField.Value = 0;
779             app.RZXn1vertEditField.Value = 0;
780             app.RZYn1vertEditField.Value = 0;
781         end
782 
783         function [] = updateRZVert(app)
784             app.Vert.orig(1,1) = app.RZXnvertEditField.Value;
785             app.Vert.orig(1,2) = app.RZYnvertEditField.Value;
786             app.Vert.dest(1,1) = app.RZXn1vertEditField.Value;
787             app.Vert.dest(1,2) = app.RZYn1vertEditField.Value;
788         end
789 
790         function [] = resetRZeditfieldarch(app)
791             app.RZXnarchEditField.Value = 0;
792             app.RZYnarchEditField.Value = 0;
793             app.RZXn1archEditField.Value = 0;
794             app.RZYn1archEditField.Value = 0;
795             app.RZARadiusEditField.Value = 0;
796             app.RZAConcaveConvexSwitch.Value = '^';
797             app.RZACortoLargoSwitch.Value = 'Corto';
798         end
799 
800         function [] = updateRZArch(app)
801             app.Arch.orig(1) = app.RZXnarchEditField.Value;
802             app.Arch.orig(2) = app.RZYnarchEditField.Value;
803             app.Arch.dest(1) = app.RZXn1archEditField.Value;
804             app.Arch.dest(2) = app.RZYn1archEditField.Value;
805             app.Arch.rad = app.RZARadiusEditField.Value;
806             RZAConcaveConvexSwitchValueChanged(app)
807             app.Arch.len = app.RZACortoLargoSwitch.Value;
808         end
809 
810         function [] = clearFLeditfields(app,param)
811             if param(1) == 'R'
812                 app.RZFLsupEditField.Value = ' ';
813                 app.RZFLinfEditField.Value = ' ';
814             elseif param(1) == 'A'
815                 app.FLsupEditField.Value = ' ';
816                 app.FLinfEditField.Value = ' ';
```

```
817                 app.RZFLsupEditField.Value = ' ';
818                 app.RZFLinfEditField.Value = ' ';
819             end
820         end
821 
822         function [] = resetRZeditfieldalt(app)
823             app.RZZinfEditField.Value = 0;
824             app.RZZsupEditField.Value = 0;
825             app.FLftinfDropDownRZ.Value = '1';
826             app.FLftsupDropDownRZ.Value = '1';
827         end
828 
829         function [] = updateRZAlt(app)
830             RZZinfEditFieldValueChanged(app)
831             RZZsupEditFieldValueChanged(app)
832         end
833 
834         function [] = resetAirporteditfield(app)
835             app.NombreAirportEditField.Value = '';
836             app.XairportEditField.Value = 0;
837             app.YairportEditField.Value = 0;
838         end
839 
840         function [] = updateAirport(app)
841             NombreAirportEditFieldValueChanged(app)
842             XairportEditFieldValueChanged(app)
843             YairportEditFieldValueChanged(app)
844             app.Airport.Runway = [];
845         end
846 
847         function [] = resetRWYPanel(app)
848             selNode = app.AirportTree.SelectedNodes;
849             if ~isempty(selNode)
850                 if selNode.Parent.Tag(1) == 'T'
851                     k = selNode.NodeData;
852                 elseif selNode.Parent.Tag(1) == 'N'
853                     k = selNode.Parent.NodeData;
854                 end
855                 RWY = app.SectorEditor.Sector.Airport(k).Runway;
856                 RWYDDItems = app.RWYEditDropDown.Items;
857                 RWYDDItemsData = app.RWYEditDropDown.ItemsData;
858                 if ~isempty(RWY)
859                     j = 1;
860                     for i = 1:size(RWY,2)
861                         if logical(mod(i,2))
862                             CharNum = dec2char(app,RWY(i));
863                             RWYWchar = dec2char(app,RWY(i+1));
864                             if RWY(i) < 10
865                                 RWYEchar = char(['0',CharNum]);
866                             else
867                                 RWYEchar = CharNum;
868                             end
869                             RWYDDItems{j+1} = char([RWYEchar,'/',RWYWchar]);
870                             RWYDDItemsData(j+1) = j;
871                             j = j+1;
872                         end
873                     end
```

```
874                     app.RWYEditDropDown.Items = RWYDDItems;
875                     app.RWYEditDropDown.ItemsData = RWYDDItemsData;
876                 else
877                     app.RWYEditDropDown.Items = {'--'};
878                     app.RWYEditDropDown.ItemsData = 0;
879                 end
880             else
881                 app.RWYEditDropDown.Items = {'--'};
882                 app.RWYEditDropDown.ItemsData = 0;
883             end
884             clearRWYspinners(app)
885             app.RWYNewEditSwitch.Value = 'Nueva';
886             RWYNewEditSwitchValueChanged(app)
887             app.RWYEditDropDown.Value = 0;
888             RWYEditDropDownValueChanged(app)
889         end
890 
891         function [] = clearRWYspinners(app)
892             app.NewRWYSpinner.Value = 0;
893             app.RWYEditSpinner.Value = 0;
894             NewRWYSpinnerValueChanged(app)
895             RWYEditSpinnerValueChanged(app)
896         end
897 
898         function [] = resetWaypointeditfield(app)
899             app.NombreWPEditField.Value = '';
900             app.XwaypointEditField.Value = 0;
901             app.YwaypointEditField.Value = 0;
902         end
903 
904         function [] = updateWaypoint(app)
905             NombreWPEditFieldValueChanged(app)
906             XwaypointEditFieldValueChanged(app)
907             YwaypointEditFieldValueChanged(app)
908         end
909 
910         function [] = resetWPfreqPanel(app)
911             app.FreqMHzEditField.Value = 0;
912             FreqMHzEditFieldValueChanged(app)
913         end
914 
915         function h = isvalidcallsign(~,value)
916             h = false;
917             a = uint16('A');
918             w = uint16('W');
919             z = uint16('Z');
920             j = [0,0,0];
921             if value(1) == 'E' && value(2) == 'C' && value(3) == '-'
922                 for i = 4:6
923                     l = uint16(value(i));
924                     if i == 4
925                         if l >= a && l <= w
926                             j(i-3) = 1;
927                         end
928                     elseif i > 4
929                         if l >= a && l <= z
930                             j(i-3) = 1;
```

```
                    end
                end
            end
            if isempty(find(~j, 1))
                h = true;
            end
        end
    end

    function h = isvalidETO(~,value)
        h = false;
        zero = uint16('0');
        two = uint16('2');
        three = uint16('3');
        five = uint16('5');
        nine = uint16('9');
        j = [0 0 0 0 0];
        if value(3) == ':'
            j(3) = 1;
            dh = uint16(value(1));
            uh = uint16(value(2));
            dm = uint16(value(4));
            um = uint16(value(5));
            if dh >= zero && dh <= two
                j(1) = 1;
            end
            if dh == two
                if uh >= zero && uh <= three
                    j(2) = 1;
                end
            else
                if uh >= zero && uh <= nine
                    j(2) = 1;
                end
            end
            if dm >= zero && dm <= five
                j(4) = 1;
            end
            if um >= zero && um <= nine
                j(5) = 1;
            end
        end
        if isempty(find(~j, 1))
            h = true;
        end
    end

    function Num = char2dec(~,char)
        zero = uint16('0');
        d = zeros(size(char));
        p = ones(size(char));
        for i = 1:length(char)
            d(i) = double(uint16(char(i))-zero);
            p(end-i+1) = p(end-i+1)*10^(i-1);
        end
        Num = p*d';
    end
```

```
988
989         function h = iswp(app,wp)
990             allmayus = true;
991             h = false;
992             for i = 1:length(wp)
993                 if uint16(wp(i)) >= 65 && uint16(wp(i))<=90
994                 else
995                     allmayus = false;
996                 end
997             end
998             if length(wp) == 5 && allmayus
999                 for j = 1:length(app.WaypointTree.Children)
1000                    k = find(wp==app.WaypointTree.Children(j).Text);
1001                    if length(k) == 5
1002                        h = true;
1003                        break
1004                    end
1005                end
1006                if ~h
1007                    for j = 1:length(app.AirportTree.Children)
1008                        k = find(wp==app.AirportTree.Children(j).Text);
1009                        if length(k) == 5
1010                            h = true;
1011                            break
1012                        end
1013                    end
1014                end
1015            end
1016        end
1017
1018        function pos = findwp(app,wp)
1019            allmayus = true;
1020            h = false;
1021            for i = 1:length(wp)
1022                if uint16(wp(i)) >= uint16('A') && uint16(wp(i))<=uint16('Z')
1023                else
1024                    allmayus = false;
1025                end
1026            end
1027            if length(wp) == 5 && allmayus
1028                for j = 1:length(app.WaypointTree.Children)
1029                    k = find(wp==app.WaypointTree.Children(j).Text);
1030                    if length(k) == 5
1031                        h = true;
1032                        pos = [app.SectorEditor.Sector.Waypoint(j).X,app.
                                SectorEditor.Sector.Waypoint(j).Y];
1033                        break
1034                    end
1035                end
1036                if ~h
1037                    for j = 1:length(app.AirportTree.Children)
1038                        k = find(wp==app.AirportTree.Children(j).Text);
1039                        if length(k) == 5
1040                            pos = [app.SectorEditor.Sector.Airport(j).X,app.
                                    SectorEditor.Sector.Airport(j).Y];
1041                            break
1042                        end
```

```
1043                    end
1044                end
1045            end
1046        end
1047
1048        function [] = clearIncident(app)
1049            app.Incident.Type = '0';
1050            app.Incident.Trigger = '';
1051            app.Incident.Target = [];
1052            app.Incident.Time = '1';
1053        end
1054
1055        function [] = updateAirplaneTab(app)
1056            app.NiveldeVueloEditField.Value = app.Airplane.FL;
1057            app.MatrculadelaaeronaveEditField.Value = app.Airplane.Callsign;
1058            app.VelocidaddeVueloktEditField.Value = app.Airplane.FSPD;
1059            if ~isempty(app.Airplane.ETO)
1060                app.ETOEditField.Value = char(app.Airplane.ETO,'hh:mm');
1061            else
1062                app.ETOEditField.Value = '';
1063            end
1064            app.EHDGEditField.Value = app.Airplane.HDG;
1065            delete(app.RouteTree.Children);
1066            app.RouteTree.Children = [];
1067            loadRouteTree(app);
1068            app.RouteWPEditField.Value = '';
1069            RouteWPEditFieldValueChanged(app);
1070            app.Incident = app.Airplane.Incident;
1071            updateIncidentTab(app);
1072        end
1073
1074
1075        function [] = loadRouteTree(app)
1076            for j = 1:length(app.Airplane.Route)
1077                if length(app.Airplane.Route{j}) == 5
1078                    uitreenode(app.RouteTree,'Text',app.Airplane.Route{j},'
                            NodeData',j);
1079                end
1080            end
1081        end
1082
1083        function [] = updateIncidentTab(app)
1084            app.TipodeIncidenciaDropDown.Value = app.Incident.Type;
1085            value = app.Incident.Type;
1086            if value == '0'
1087                app.CambioFLPanel.Visible = 'off';
1088                app.SolicitarDirectoPanel.Visible = 'off';
1089                app.CRNPPanel.Visible = 'off';
1090                app.Incident.Type = value;
1091            elseif value == '1'
1092                app.CambioFLPanel.Visible = 'on';
1093                app.SolicitarDirectoPanel.Visible = 'off';
1094                app.CRNPPanel.Visible = 'off';
1095                app.Incident.Type = value;
1096            elseif value == '2'
1097                app.CambioFLPanel.Visible = 'off';
1098                app.SolicitarDirectoPanel.Visible = 'on';
```

```
1099                app.CRNPPanel.Visible = 'off';
1100                app.Incident.Type = value;
1101            elseif value == '3'
1102                app.CambioFLPanel.Visible = 'off';
1103                app.SolicitarDirectoPanel.Visible = 'off';
1104                app.CRNPPanel.Visible = 'on';
1105                app.Incident.Type = value;
1106            end
1107            if app.Incident.Type == '1'
1108                app.WaypointTriggerCFLEditField.Value = app.Incident.Trigger;
1109                app.FLObjetivoEditField.Value = app.Incident.Target;
1110                app.CFLDropDown.Value = app.Incident.Time;
1111            elseif app.Incident.Type == '2'
1112                app.WaypointTriggerSDEditField.Value = app.Incident.Trigger;
1113                app.WPDirectObjetivoEditField.Value = app.Incident.Target;
1114                app.SDDropDown.Value = app.Incident.Time;
1115            elseif app.Incident.Type == '3'
1116                app.WaypointTriggerCRNPEditField.Value = app.Incident.Trigger;
1117                app.RumboNuevoEditField.Value = app.Incident.Target;
1118                app.CRNPDropDown.Value = app.Incident.Time;
1119            end
1120        end
1121
1122        function [] = plotRoute(app,Route,hdg)
1123            pos = findwp(app,Route{1});
1124            spdkt = app.Airplane.FSPD;
1125            spdnmmin = spdkt/60;
1126            PreRoute = pos - spdnmmin*[sind(hdg),cosd(hdg)];
1127            X = zeros(1,length(Route)+1);
1128            Y = zeros(1,length(Route)+1);
1129            X(1) = PreRoute(1);
1130            Y(1) = PreRoute(2);
1131            for i = 1:length(Route)
1132                pos = findwp(app,Route{i});
1133                X(i+1) = pos(1);
1134                Y(i+1) = pos(2);
1135            end
1136            plot(app.Editorgraph,X,Y,'-m');
1137        end
1138
1139        function [] = clearAirplaneTab(app)
1140            clearAirplane(app);
1141            updateAirplaneTab(app);
1142        end
1143
1144        function [] = clearAirplane(app)
1145            app.Airplane.Callsign = '';
1146            app.Airplane.FL = 0;
1147            app.Airplane.FSPD = 0;
1148            app.Airplane.HDG = 0;
1149            app.Airplane.ETO = duration(0,0,0);
1150            app.Airplane.Route = {''};
1151            clearIncident(app);
1152            app.Airplane.Incident = app.Incident;
1153        end
1154
1155        function [] = plotSimuladorgraph(app)
```

```
1156            if ~isempty(app.SectorSimulador.Sector.Border)
1157                Xo = cell2mat(app.SectorSimulador.Sector.Border.X);
1158                Yo = cell2mat(app.SectorSimulador.Sector.Border.Y);
1159                plot(app.Simuladorgraph,Xo,Yo,'-g')
1160                hold(app.Simuladorgraph,"on")
1161                if ~isempty(Xo)
1162                    Xomin = min(Xo);
1163                    Xomax = max(Xo);
1164                    Yomin = min(Yo);
1165                    Yomax = max(Yo);
1166                    d = max(0.1*(Xomax-Xomin),0.1*(Yomax-Yomin));
1167                    Xb = [Xomin-d, Xomax+d, Xomax+d, Xomin-d];
1168                    Yb = [Yomin-d, Yomin-d, Yomax+d, Yomax+d];
1169                    plot(app.Simuladorgraph,Xb,Yb,'.k');
1170                end
1171            end
1172            if ~isempty(app.SectorSimulador.Sector.RestricZones)
1173                for k = 1:size(app.SectorSimulador.Sector.RestricZones,2)
1174                    Xz = cell2mat(app.SectorSimulador.Sector.RestricZones(k).X);
1175                    Yz = cell2mat(app.SectorSimulador.Sector.RestricZones(k).Y);
1176                    plot(app.Simuladorgraph,Xz,Yz,'-r')
1177                    hold(app.Simuladorgraph,"on")
1178                end
1179            end
1180            if ~isempty(app.SectorSimulador.Sector.Airport)
1181                hold(app.Simuladorgraph,"on")
1182                for i = 1:size(app.SectorSimulador.Sector.Airport,2)
1183                    Xa = app.SectorSimulador.Sector.Airport(i).X;
1184                    Ya = app.SectorSimulador.Sector.Airport(i).Y;
1185                    Namea = char(app.SectorSimulador.Sector.Airport(i).Name);
1186                    plot(app.Simuladorgraph,Xa,Ya,'dc')
1187                    text(app.Simuladorgraph,Xa,Ya,Namea,'Color',[0 1 1],'
                            HorizontalAlignment','center','VerticalAlignment','bottom
                            ')
1188                end
1189            end
1190            if ~isempty(app.SectorSimulador.Sector.Waypoint)
1191                hold(app.Simuladorgraph,"on")
1192                for i = 1:size(app.SectorSimulador.Sector.Waypoint,2)
1193                    Xw = app.SectorSimulador.Sector.Waypoint(i).X;
1194                    Yw = app.SectorSimulador.Sector.Waypoint(i).Y;
1195                    Namew = app.SectorSimulador.Sector.Waypoint(i).Name;
1196                    plot(app.Simuladorgraph,Xw,Yw,'^c')
1197                    text(app.Simuladorgraph,Xw,Yw,Namew,'Color',[0 1 1],'
                            HorizontalAlignment','center','VerticalAlignment','bottom
                            ')
1198                end
1199            end
1200            app.Simuladorgraph.DataAspectRatio = [1 1 1];
1201        end
1202
1203        function [] = LoadScenarioSim(app)
1204            plotSimuladorgraph(app)
1205            hold(app.Simuladorgraph,'off')
1206            app.AirplaneSim = app.SectorSimulador.Sector.Airplane;
1207            if ~isempty(app.AirplaneSim)
1208                OrdenaAirplaneSim(app);
```

```
1209                app.FLSRCSEF(1).Value = app.AirplaneSim(1).Callsign;
1210                FLchar = dec2char(app,app.AirplaneSim(1).FL);
1211                SPDchar = dec2char(app,app.AirplaneSim(1).FSPD);
1212                app.FLSRFLEF(1).Value = char(['FL',FLchar]);
1213                app.FLSRSPDEF(1).Value = char([SPDchar,'kt']);
1214                app.DropDown.Items{2} = app.AirplaneSim(1).Callsign;
1215                app.DropDown.ItemsData{2} = '1';
1216                app.FlightStrip(1).Callsign = app.AirplaneSim(1).Callsign;
1217                app.FlightStrip(1).FL = app.AirplaneSim(1).FL;
1218                app.FlightStrip(1).FSPD = app.AirplaneSim(1).FSPD;
1219                app.FlightStrip(1).ETO = app.AirplaneSim(1).ETO;
1220                app.FlightStrip(1).Route = cell(size(app.AirplaneSim(1).Route));
1221                app.FlightStrip(1).TPWP = cell(size(app.AirplaneSim(1).Route));
1222                app.AirplaneSim(1).X = [];
1223                app.AirplaneSim(1).Y = [];
1224                app.AirplaneSim(1).Z = 100*app.AirplaneSim(1).FL;
1225                for j = 1:length(app.AirplaneSim(1).Route)
1226                    app.FlightStrip(1).Route{j} = app.AirplaneSim(1).Route{j};
1227                    app.FlightStrip(1).TPWP{j} = {' '};
1228                end
1229                l = length(app.AirplaneSim);
1230                for i = 2:l
1231                    y = 20 + (i-1)*105;
1232                    app.FLSRButton(i) = uibutton(app.FlightStripRackPanel,'state'
                        ,'Text','','Position',[20,y,225,100],'UserData',i);
1233                    app.FLSRButton(i).ValueChangedFcn = createCallbackFcn(app,
                        @FLSRButtonValueChanged, true);
1234                    FLchar = dec2char(app,app.AirplaneSim(i).FL);
1235                    SPDchar = dec2char(app,app.AirplaneSim(i).FSPD);
1236                    app.FLSRCSEF(i) = uieditfield(app.FlightStripRackPanel,'text'
                        ,'Value',app.AirplaneSim(i).Callsign,'Position',[40,y
                        +69,100,22],'Editable','off');
1237                    app.FLSRFLEF(i) = uieditfield(app.FlightStripRackPanel,'Text'
                        ,'Value',char(['FL',FLchar]),'Position',[40,y+39,70,22],'
                        Editable','off');
1238                    app.FLSRSPDEF(i) = uieditfield(app.FlightStripRackPanel,'Text
                        ','Value',char([SPDchar,'kt']),'Position',[40,y+9,70,22],
                        'Editable','off');
1239                    app.FLSRLamp(i) = uilamp(app.FlightStripRackPanel,'Color'
                        ,[1,0,0],'Position',[205,y+40,20,20],'UserData',i,'Enable
                        ','off');
1240                    app.DropDown.Items{i+1} = app.AirplaneSim(i).Callsign;
1241                    app.DropDown.ItemsData{i+1} = dec2char(app,i);
1242                    app.FlightStrip(i).Callsign = app.AirplaneSim(i).Callsign;
1243                    app.FlightStrip(i).FL = app.AirplaneSim(i).FL;
1244                    app.FlightStrip(i).FSPD = app.AirplaneSim(i).FSPD;
1245                    app.FlightStrip(i).ETO = app.AirplaneSim(i).ETO;
1246                    app.FlightStrip(i).Route = cell(size(app.AirplaneSim(i).Route
                        ));
1247                    app.FlightStrip(i).TPWP = cell(size(app.AirplaneSim(1).Route)
                        );
1248                    for j = 1:length(app.AirplaneSim(i).Route)
1249                        app.FlightStrip(i).Route{j} = app.AirplaneSim(i).Route{j
                            };
1250                        app.FlightStrip(i).TPWP{j} = {' '};
1251                    end
1252                    app.AirplaneSim(i).X = [];
```

```
1253                    app.AirplaneSim(i).Y = [];
1254                    app.AirplaneSim(i).Z = 100*app.AirplaneSim(i).FL;
1255                end
1256                app.SimData.Autoriz = struct('Type',{zeros(1,l)},'Value',{false
                        (1,l)});
1257                app.SimData.ETOCheck = zeros(1,l);
1258                app.SimData.Alert = false(1,l);
1259                app.SimData.Comms = zeros(1,l);
1260                app.SimData.Tracks = struct('X',NaN(l,5),'Y',NaN(l,5));
1261                app.SimData.RouteTrack = ones(1,l);
1262                app.SimData.Turning = zeros(1,l);
1263                n = length(app.SectorEditor.Sector.RestricZones);
1264                app.SimData.Conflicts = false(l+n);
1265                app.SimData.Climbing = zeros(1,l);
1266                app.SimData.DoneIncident = false(1,l);
1267            end
1268            if ~isempty(app.SectorSimulador.Sector.Time)
1269                app.TimeSim = app.SectorSimulador.Sector.Time.init;
1270                app.TimeSimEditField.Value = char(app.TimeSim,'hh:mm:ss');
1271                app.TimeEndEditField.Value = char(app.SectorSimulador.Sector.
                        Time.fin,'hh:mm:ss');
1272            end
1273            app.InputmsgTextArea.Value = ' ';
1274            app.CommsTextArea.Value = ' ';
1275        end
1276
1277
1278        function [] = OrdenaAirplaneSim(app)
1279            FL = zeros(1,length(app.AirplaneSim));
1280            for k = 1:length(app.AirplaneSim)
1281                FL(k) = app.AirplaneSim(k).FL;
1282            end
1283            i = zeros(size(FL));
1284            for j = 1:length(i)
1285                [~,i(j)] = min(FL,[],'omitnan');
1286                FL(i(j)) = NaN;
1287            end
1288            app.AirplaneSim = app.AirplaneSim(i);
1289        end
1290
1291        function [] = clearFlightStrip(app)
1292            for i = 2:length(app.WPRNEF)
1293                delete(app.WPRNEF(i));
1294                delete(app.TPWPEF(i));
1295            end
1296            app.WPRNEF(1).Value = ' ';
1297            app.TPWPEF(1).Value = ' ';
1298        end
1299
1300        function [] = clearSectorSim(app)
1301            app.SectorSimulador.Sector.Name = {};
1302            app.SectorSimulador.Sector.Time = [];
1303            app.SectorSimulador.Sector.Border = [];
1304            app.SectorSimulador.Sector.RestricZones = [];
1305            app.SectorSimulador.Sector.Airplane = [];
1306            app.SectorSimulador.Sector.Airport = [];
1307            app.SectorSimulador.Sector.Waypoint = [];
```

```
1308        end
1309
1310        function [] = clearlamps(app)
1311            app.GeneralConflictLamp.Enable = 'off';
1312            app.FLSRLamp(1).Enable = 'off';
1313        end
1314
1315        function [] = clearTimers(app)
1316            t0 = duration(0,0,0);
1317            app.TimeSimEditField.Value = char(t0,'hh:mm:ss');
1318            app.TimeEndEditField.Value = char(t0,'hh:mm:ss');
1319            app.TimeSim = t0;
1320        end
1321
1322        function [] = clearFLSR(app)
1323            for i = 2:length(app.FLSRButton)
1324                delete(app.FLSRButton(i))
1325                delete(app.FLSRCSEF(i))
1326                delete(app.FLSRFLEF(i))
1327                delete(app.FLSRSPDEF(i))
1328                delete(app.FLSRLamp(i))
1329            end
1330            app.FLSRButton(1).Value = false;
1331            app.FLSRCSEF(1).Value = ' ';
1332            app.FLSRFLEF(1).Value = ' ';
1333            app.FLSRSPDEF(1).Value = ' ';
1334            app.FLSRLamp(1).Enable = 'off';
1335        end
1336
1337        function Rad = encuentraRadial(~,postrig,postarg)
1338            C = postarg(2)-postrig(2);
1339            Si = postarg(1)-postrig(1);
1340            H = norm(postarg-postrig);
1341            alpha = acosd(C/H);
1342            if Si >= 0
1343                Rad = alpha;
1344            elseif Si < 0
1345                Rad = 360-alpha;
1346            end
1347        end
1348
1349        function wpi = locatewp(app,wp)
1350            for j = 1:length(app.SectorSimulador.Sector.Waypoint)
1351                L = find(wp==app.SectorSimulador.Sector.Waypoint(j).Name);
1352                if length(L) == 5
1353                    wpi = j;
1354                    break
1355                end
1356            end
1357        end
1358
1359        function [] = ETOCheck(app)
1360            for i = 1:length(app.AirplaneSim)
1361                if app.TimeSim >= app.AirplaneSim(i).ETO - duration(0,1,0) &&
                        app.SimData.ETOCheck(i) == 0
1362                    app.SimData.ETOCheck(i) = 1;
1363                    pos = findwpsim(app,app.AirplaneSim(i).Route{1});
```

```
1364                    spdmin = app.AirplaneSim(i).FSPD/60;
1365                    hdg = app.AirplaneSim(i).HDG;
1366                    PreRoute = pos - spdmin*[sind(hdg) cosd(hdg)];
1367                    app.AirplaneSim(i).X = PreRoute(1);
1368                    app.AirplaneSim(i).Y = PreRoute(2);
1369                    app.SimData.Comms(i) = 1;
1370                end
1371            end
1372            plotAirplanes(app);
1373        end
1374
1375        function [] = SimNav(app)
1376            for i = 1: length(app.AirplaneSim)
1377                if app.SimData.ETOCheck(i) == 1
1378                    acpos = [app.AirplaneSim(i).X,app.AirplaneSim(i).Y];
1379                    acspdsec = app.AirplaneSim(i).FSPD/3600;
1380                    achdg = app.AirplaneSim(i).HDG;
1381                    RT = app.SimData.RouteTrack(i);
1382                    if RT < length(app.AirplaneSim(i).Route) || app.SimData.
                            Turning(i) == 2
1383                        turning = app.SimData.Turning(i);
1384                        if turning == 0
1385                            nxtpos = acpos + 10*acspdsec*[sind(achdg) cosd(achdg)
                                    ];
1386                            WPtarg = app.AirplaneSim(i).Route{RT};
1387                            WPtarg1 = app.AirplaneSim(i).Route{RT+1};
1388                            wptpos = findwpsim(app,WPtarg);
1389                            wpt1pos = findwpsim(app,WPtarg1);
1390                            hdgtarg = encuentraRadial(app,wptpos,wpt1pos);
1391                            turn = (hdgtarg - achdg)/abs(hdgtarg - achdg);
1392                            if abs(hdgtarg-achdg) > 180
1393                                hdgtarg = hdgtarg - turn*360;
1394                            end
1395                            nturns = abs(hdgtarg-achdg)/30;
1396                            if nturns >= 1
1397                                if mod(nturns,floor(nturns)) == 0
1398                                    nturns = nturns - 1;
1399                                else
1400                                    nturns = floor(nturns);
1401                                end
1402                                cturn = 0;
1403                                sturn = 0;
1404                                turn = (hdgtarg - achdg)/abs(hdgtarg - achdg);
1405                                for n = 1:nturns
1406                                    cturn = cturn + cosd(achdg + turn*n*30);
1407                                    sturn = sturn + sind(achdg + turn*n*30);
1408                                end
1409                                xturn = 10*acspdsec*sturn + nxtpos(1);
1410                                yturn = 10*acspdsec*cturn + nxtpos(2);
1411                                facpos = [xturn yturn];
1412                                inawy = inawycheck(app,hdgtarg,wptpos,facpos);
1413                                if inawy
1414                                    turning = 1;
1415                                end
1416                            elseif nturns ~= 0
1417                                nxthdg = achdg + hdgtarg-achdg;
1418                                Tturn = abs(hdgtarg-achdg)/3;
```

```
                            fnxtpos = nxtpos + Tturn*acspdsec*[sind(achdg)
                                cosd(achdg)] + (10-Tturn)*acspdsec*[sind(
                                nxthdg) cosd(nxthdg)];
                            inawy = inawycheck(app,hdgtarg,wptpos,fnxtpos);
                            if inawy || norm(wptpos - nxtpos) <= 15*acspdsec
                                turning = 1;
                            end
                        elseif nturns == 0
                            if norm(wptpos - nxtpos) <= 10*acspdsec
                                app.SimData.RouteTrack(i) = RT+1;
                                app.SimData.Comms(i) = 0;
                            end
                        end
                        app.AirplaneSim(i).X = nxtpos(1);
                        app.AirplaneSim(i).Y = nxtpos(2);
                        app.AirplaneSim(i).HDG = achdg;
                        app.SimData.Turning(i) = turning;
                    elseif turning == 1
                        WPtarg = app.AirplaneSim(i).Route{RT};
                        WPtarg1 = app.AirplaneSim(i).Route{RT+1};
                        wptpos = findwpsim(app,WPtarg);
                        wpt1pos = findwpsim(app,WPtarg1);
                        hdgtarg = encuentraRadial(app,wptpos,wpt1pos);
                        turn = (hdgtarg - achdg)/abs(hdgtarg - achdg);
                        if abs(hdgtarg-achdg) > 180
                            hdgtarg = hdgtarg - turn*360;
                        end
                        if hdgtarg == achdg
                            turn = 0;
                        else
                            turn = (hdgtarg - achdg)/abs(hdgtarg - achdg);
                        end
                        if abs(hdgtarg-achdg) < 30
                            nxthdg = achdg + hdgtarg-achdg;
                            Tturn = abs(hdgtarg-achdg)/3;
                            turning = 0;
                            app.SimData.RouteTrack(i) = RT+1;
                            app.SimData.Comms(i) = 0;
                        else
                            nxthdg = achdg + turn*30;
                            Tturn = 10;
                        end
                        inawy = inawycheck(app,hdgtarg,wptpos,acpos);
                        if inawy && turning ~= 0 && achdg == hdgtarg
                            turning = 0;
                            app.SimData.RouteTrack(i) = RT+1;
                            app.SimData.Comms(i) = 0;
                        end
                        nxtpos = acpos + Tturn*acspdsec*[sind(achdg) cosd(
                            achdg)] + (10-Tturn)*acspdsec*[sind(nxthdg) cosd(
                            nxthdg)];
                        app.AirplaneSim(i).X = nxtpos(1);
                        app.AirplaneSim(i).Y = nxtpos(2);
                        app.AirplaneSim(i).HDG = nxthdg;
                        app.SimData.Turning(i) = turning;
                    elseif turning == 2
                        hdgtarg = app.AirplaneSim(i).Incident.Target;
```

```
1472                        turn = (hdgtarg - achdg)/abs(hdgtarg - achdg);
1473                        if abs(hdgtarg-achdg) > 180
1474                            hdgtarg = hdgtarg - turn*360;
1475                        end
1476                        if hdgtarg == achdg
1477                            turn = 0;
1478                        else
1479                            turn = (hdgtarg - achdg)/abs(hdgtarg - achdg);
1480                        end
1481                        if abs(hdgtarg-achdg) < 30
1482                            nxthdg = achdg + hdgtarg-achdg;
1483                            Tturn = abs(hdgtarg-achdg)/3;
1484                        else
1485                            nxthdg = achdg + turn*30;
1486                            Tturn = 10;
1487                        end
1488                        if app.SimData.Autoriz.Type(i) == 3 && app.SimData.
                                Autoriz.Value(i)
1489                            if hdgtarg == achdg
1490                                turning = 0;
1491                            elseif app.SimData.Comms(i) ~=6
1492                                app.SimData.Comms(i) = 4;
1493                            end
1494                        end
1495                        nxtpos = acpos + Tturn*acspdsec*[sind(achdg) cosd(
                                achdg)] + (10-Tturn)*acspdsec*[sind(nxthdg) cosd(
                                nxthdg)];
1496                        app.AirplaneSim(i).X = nxtpos(1);
1497                        app.AirplaneSim(i).Y = nxtpos(2);
1498                        app.AirplaneSim(i).HDG = nxthdg;
1499                        app.SimData.Turning(i) = turning;
1500                    end
1501                elseif RT == length(app.AirplaneSim(i).Route)
1502                    nxtpos = acpos + 10*acspdsec*[sind(achdg) cosd(achdg)];
1503                    wptpos = findwpsim(app,app.AirplaneSim(i).Route{RT});
1504                    if norm(wptpos - nxtpos) <= 90*acspdsec
1505                        if app.SimData.Comms(i) == 6 && app.SimData.Autoriz.
                                Value(i) == true && app.SimData.Autoriz.Type(i) ==
                                 4
1506                            app.SimData.ETOCheck(i) = 2;
1507                        elseif app.SimData.Comms(i) ~= 4 && app.SimData.Comms
                                (i) ~= 6
1508                            app.SimData.Comms(i) = 5;
1509                        end
1510                    end
1511                    app.AirplaneSim(i).X = nxtpos(1);
1512                    app.AirplaneSim(i).Y = nxtpos(2);
1513                end
1514                hdg = app.AirplaneSim(i).HDG;
1515                if hdg < 0
1516                    app.AirplaneSim(i).HDG = hdg + 360;
1517                elseif hdg > 359
1518                    app.AirplaneSim(i).HDG = hdg - 360;
1519                end
1520                app.SimData.Tracks.X(i,:) = [app.SimData.Tracks.X(i,2:5)
                        acpos(1)];
```

```
1521                    app.SimData.Tracks.Y(i,:) = [app.SimData.Tracks.Y(i,2:5)
                            acpos(2)];
1522                    climbing = app.SimData.Climbing(i);
1523                    if climbing ~= 0
1524                        acalt = app.AirplaneSim(i).Z;
1525                        alttarg = app.AirplaneSim(i).Incident.Target*100;
1526                        nxtalt = acalt + 1000*climbing/6;
1527                        if abs(alttarg-nxtalt) < 99
1528                            climbing = 0;
1529                            app.SimData.Autoriz.Type(i) = 0;
1530                            app.SimData.Autoriz.Value(i) = false;
1531                            app.SimData.Comms(i) = 0;
1532                        end
1533                        app.AirplaneSim(i).Z = nxtalt;
1534                        app.AirplaneSim(i).FL = floor(nxtalt/100);
1535                        if climbing == 1
1536                            arr = '^';
1537                        elseif climbing == -1
1538                            arr = 'v';
1539                        else
1540                            arr = '';
1541                        end
1542                        app.SimData.Climbing(i) = climbing;
1543                        FLchar = FLChar(app,nxtalt);
1544                        app.FLSRFLEF(i).Value = char([FLchar,arr]);
1545                        app.FlightStrip(i).FL = floor(nxtalt/100);
1546                        if app.FLSRButton(i).Value
1547                            app.FSFLEditField.Value = FLchar;
1548                        end
1549                    end
1550                    if app.AirplaneSim(i).Incident.Type ~= '0' && ~app.SimData.
                            DoneIncident(i)
1551                        In = app.AirplaneSim(i).Incident;
1552                        if In.Time == '1'
1553                            rtwp = app.AirplaneSim(i).Route{RT};
1554                            L = find(In.Trigger==rtwp);
1555                            if length(L) == 5
1556                                trigpos = findwpsim(app,In.Trigger);
1557                                if In.Type == '1'
1558                                   acalt = app.AirplaneSim(i).Z;
1559                                   alttarg = In.Target*100;
1560                                   climb = alttarg - acalt;
1561                                   tclimb = 60*abs(climb)/1000;
1562                                   sclimb = climb/abs(climb);
1563                                   d = (tclimb+60)*acspdsec;
1564                                   if norm(trigpos-nxtpos) <= d
1565                                       if app.SimData.Comms(i) == 6 && app.SimData.
                                               Autoriz.Value(i) == true && app.SimData.
                                               Autoriz.Type(i) == 1
1566                                           app.SimData.Climbing(i) = sclimb;
1567                                           app.SimData.DoneIncident(i) = true;
1568                                       elseif app.SimData.Comms(i) ~= 4 && app.
                                               SimData.Comms(i) ~= 6
1569                                           app.SimData.Comms(i) = 2;
1570                                       end
1571                                   end
1572                                elseif In.Type == '2'
```

```
1573                                    if norm(trigpos-nxtpos) <= 90*acspdsec
1574                                        if app.SimData.Comms(i) == 6 && app.SimData
                                                .Autoriz.Value(i) == true && app.
                                                SimData.Autoriz.Type(i) == 2
1575                                          for p = 1:length(app.AirplaneSim(i).
                                                  Route)
1576                                              rtwp = app.AirplaneSim(i).Route{p};
1577                                              L1 = find(In.Trigger==rtwp);
1578                                              L2 = find(In.Target==rtwp);
1579                                              if length(L1) == 5
1580                                                  wp1i = p;
1581                                              elseif length(L2) == 5
1582                                                  wp2i = p;
1583                                                  break
1584                                              end
1585                                          end
1586                                          rtlong = length(app.AirplaneSim(i).Route
                                                );
1587                                          Direct = cell(rtlong-wp2i+wp1i+1,1);
1588                                          r = 1;
1589                                          for p = [1:wp1i,wp2i:rtlong]
1590                                              Direct{r} = app.AirplaneSim(i).Route{
                                                    p};
1591                                              r = r+1;
1592                                          end
1593                                          app.AirplaneSim(i).Route = Direct;
1594                                          app.SimData.DoneIncident(i) = true;
1595                                       elseif app.SimData.Comms(i) ~= 4 && app.
                                              SimData.Comms(i) ~= 6
1596                                          app.SimData.Comms(i) = 3;
1597                                       end
1598                                    end
1599                                elseif In.Type == '3'
1600                                    if norm(trigpos-nxtpos) <= 90*acspdsec
1601                                        app.SimData.Turning(i) = 2;
1602                                        app.SimData.DoneIncident(i) = true;
1603                                    end
1604                                end
1605                            end
1606                        elseif In.Time == '2'
1607                            if RT > 1
1608                                rtwp1 = app.AirplaneSim(i).Route{RT};
1609                                T1 = find(In.Trigger==rtwp1);
1610                                rtwp2 = app.AirplaneSim(i).Route{RT-1};
1611                                T2 = find(In.Trigger==rtwp2);
1612                                trigpos = findwpsim(app,In.Trigger);
1613                                if length(T1) == 5
1614                                    if norm(trigpos-nxtpos) <= 90*acspdcsec
1615                                        if In.Type == '1'
1616                                            if app.SimData.Comms(i) ~= 4 && app.
                                                    SimData.Comms(i) ~= 6 && app.
                                                    SimData.Autoriz.Type == 0
1617                                                app.SimData.Comms(i) = 2;
1618                                            end
1619                                        elseif In.Type == '2'
```

```
1620                                    if app.SimData.Comms(i) ~= 4 && app.
                                            SimData.Comms(i) ~= 6 && app.
                                            SimData.Autoriz.Type == 0
1621                                        app.SimData.Comms(i) = 2;
1622                                    end
1623                                end
1624                            end
1625                        elseif length(T2) == 5
1626                            if In.Type == '1'
1627                                if app.SimData.Comms(i) == 6 && app.SimData
                                        .Autoriz.Value(i) == true && app.
                                        SimData.Autoriz.Type(i) == 1
1628                                   app.SimData.Climbing(i) = sclimb;
1629                                   app.SimData.DoneIncident(i) = true;
1630                                end
1631                            elseif In.Type == '2'
1632                                if app.SimData.Comms(i) == 6 && app.SimData
                                        .Autoriz.Value(i) == true && app.
                                        SimData.Autoriz.Type(i) == 2
1633                                   for p = 1:length(app.AirplaneSim(i).
                                           Route)
1634                                       rtwp = app.AirplaneSim(i).Route{p};
1635                                       L1 = find(In.Trigger==rtwp);
1636                                       L2 = find(In.Target==rtwp);
1637                                       if length(L1) == 5
1638                                           wp1i = p;
1639                                       elseif length(L2) == 5
1640                                           wp2i = p;
1641                                           break
1642                                       end
1643                                   end
1644                                   rtlong = length(app.AirplaneSim(i).Route
                                           );
1645                                   Direct = cell(rtlong-wp2i+wp1i+1,1);
1646                                   r = 1;
1647                                   for p = [1:wp1i,wp2i:rtlong]
1648                                       Direct{r} = app.AirplaneSim(i).Route{
                                               p};
1649                                       r = r+1;
1650                                   end
1651                                   app.AirplaneSim(i).Route = Direct;
1652                                   app.SimData.DoneIncident(i) = true;
1653                                end
1654                            elseif In.Type == '3'
1655                                app.SimData.Turning(i) = 2;
1656                                app.SimData.RouteTrack(i) = RT-1;
1657                                app.SimData.DoneIncident(i) = true;
1658                            end
1659                        end
1660                    end
1661                end
1662            end
1663        end
1664    end
1665    plotAirplanes(app);
1666 end
1667
```

```
1668        function [] = STCACheck(app)
1669            %app.SimData.Conflicts;
1670        end
1671
1672        function [] = CommCheck(app)
1673            for i = 1:length(app.AirplaneSim)
1674                if app.SimData.Comms(i) ~= 0 && app.SimData.Comms(i) ~= 6
1675                    k = app.SimData.Comms(i);
1676                    if k == 1 || k == 5
1677                        msg = cell(1,3);
1678                        msg{1} = app.AirmenCommLine{k}{1}{1};
1679                        msg{2} = app.AirplaneSim(i).Callsign;
1680                        msg{3} = app.AirmenCommLine{k}{2}{1};
1681                        txt = char([msg{1},msg{2},msg{3}]);
1682                    elseif k == 2 || k == 3
1683                        msg = cell(1,7);
1684                        msg{1} = app.AirmenCommLine{k}{1}{1};
1685                        msg{2} = app.AirplaneSim(i).Callsign;
1686                        msg{3} = app.AirmenCommLine{k}{2}{1};
1687                        if ~ischar(app.AirplaneSim(i).Incident.Target)
1688                            FLchar = dec2char(app,app.AirplaneSim(i).Incident.
                                Target);
1689                            msg{4} = char(['FL',FLchar]);
1690                        else
1691                            msg{4} = app.AirplaneSim(i).Incident.Target;
1692                        end
1693                        if app.AirplaneSim(i).Incident.Time == '1'
1694                            msg{5} = ' antes';
1695                        elseif app.AirplaneSim(i).Incident.Time == '2'
1696                            msg{5} = ' después';
1697                        end
1698                        msg{6} = app.AirmenCommLine{k}{3}{1};
1699                        msg{7} = app.AirplaneSim(i).Incident.Trigger;
1700                        txt = char([msg{1},msg{2},msg{3},msg{4},msg{5},msg{6},msg
                            {7}]);
1701                    elseif k == 4
1702                        msg = cell(1);
1703                        msg{1} = app.AirmenCommLine{4}{1}{1};
1704                        txt = char(msg);
1705                    end
1706                    Text = sprintf('%s: %s\n\n',app.AirplaneSim(i).Callsign,txt);
1707                    Comm = cell(length(app.CommsTextArea.Value)+1,1);
1708                    for t = 1:length(app.CommsTextArea.Value) + 1
1709                        if t == 1
1710                            Comm{1} = Text;
1711                        else
1712                            Comm{t} = app.CommsTextArea.Value{t-1};
1713                        end
1714                    end
1715                    app.CommsTextArea.Value = Comm;
1716                    app.SimData.Comms(i) = 6;
1717                end
1718            end
1719        end
1720
1721        function [] = plotAirplanes(app)
1722            plotSimuladorgraph(app);
```

```
1723            hold(app.Simuladorgraph,'on');
1724            for i = 1:length(app.AirplaneSim)
1725                if app.SimData.ETOCheck(i) == 1
1726                    acpos = [app.AirplaneSim(i).X app.AirplaneSim(i).Y];
1727                    achdg = app.AirplaneSim(i).HDG;
1728                    acspdmin = app.AirplaneSim(i).FSPD/60;
1729                    uhdg = [sind(achdg) cosd(achdg)];
1730                    nuhdg = [cosd(achdg) -sind(achdg)];
1731                    triac = [acpos;acpos-0.5*uhdg+0.2*nuhdg;acpos-0.5*uhdg-0.2*
                            nuhdg;acpos];
1732                    Xa = triac(:,1)';
1733                    Ya = triac(:,2)';
1734                    xpos = acpos + acspdmin*uhdg;
1735                    Xx = xpos(1);
1736                    Yx = xpos(2);
1737                    plot(app.Simuladorgraph,Xa,Ya,'-g',Xx,Yx,'xw',[acpos(1) Xx],[
                            acpos(2) Yx],'-m',app.SimData.Tracks.X(i,:),app.SimData.
                            Tracks.Y(i,:),':w');
1738                    FLchar = FLChar(app,app.AirplaneSim(i).Z);
1739                    SPDchar = dec2char(app,app.AirplaneSim(i).FSPD);
1740                    cs = app.AirplaneSim(i).Callsign;
1741                    Data = sprintf('%s\n%s\n%s kt',cs,FLchar,SPDchar);
1742                    if achdg >= 90 && achdg <= 270
1743                        text(app.Simuladorgraph,acpos(1),acpos(2),Data,'Color',[0
                                1 0],'VerticalAlignment','bottom');
1744                    else
1745                        text(app.Simuladorgraph,acpos(1),acpos(2),Data,'Color',[0
                                1 0],'VerticalAlignment','top');
1746                    end
1747                end
1748            end
1749            hold(app.Simuladorgraph,'off');
1750        end
1751
1752        function pos = findwpsim(app,wp)
1753            allmayus = true;
1754            h = false;
1755            for i = 1:length(wp)
1756                if uint16(wp(i)) >= uint16('A') && uint16(wp(i))<=uint16('Z')
1757                else
1758                    allmayus = false;
1759                end
1760            end
1761            if length(wp) == 5 && allmayus
1762                for j = 1:length(app.SectorSimulador.Sector.Waypoint)
1763                    k = find(wp==app.SectorSimulador.Sector.Waypoint(j).Name);
1764                    if length(k) == 5
1765                        h = true;
1766                        pos = [app.SectorSimulador.Sector.Waypoint(j).X,app.
                                SectorSimulador.Sector.Waypoint(j).Y];
1767                        break
1768                    end
1769                end
1770                if ~h
1771                    for j = 1:length(app.SectorSimulador.Sector.Airport)
1772                        k = find(wp==app.SectorSimulador.Sector.Airport(j).Name);
1773                        if length(k) == 5
```

```
                        pos = [app.SectorSimulador.Sector.Airport(j).X,app.
                            SectorSimulador.Sector.Airport(j).Y];
                        break
                    end
                end
            end
        end
    end

    function inawy = inawycheck(~,hdgtarg,wptpos,acpos)
        inawy = false;
        if hdgtarg ~= 0 && hdgtarg ~=180
            nuhdg = [cosd(hdgtarg) -sind(hdgtarg)];
            awylimpts = [wptpos + 5*nuhdg; wptpos - 5*nuhdg];
            [~,imax] = max(awylimpts(:,2));
            [~,imin] = min(awylimpts(:,2));
            yrefmax = awylimpts(imax,2) + (1/tand(hdgtarg))*(acpos(1)-
                awylimpts(imax,1));
            yrefmin = awylimpts(imin,2) + (1/tand(hdgtarg))*(acpos(1)-
                awylimpts(imin,1));
            if acpos(2) >= yrefmin && acpos(2) <= yrefmax
                inawy = true;
            end
        else
            if acpos(1) >= wptpos(1) - 5 && acpos(2) <= wptpos(1) + 5
                inawy = true;
            end
        end
    end
end

% Callbacks that handle component events
methods (Access = private)

    % Code that executes after component creation
    function startupFcn(app)
        movegui(app.ATCMakerApp,'center');
        app.S = load('libapp\escenarioslist.mat');
        for j = 1:size(app.S.F,2)
            Nombreescenario = char(app.S.F(j));
            uitreenode(app.TreeEditor,'Text',Nombreescenario);
            uitreenode(app.TreeSimulador,'Text',Nombreescenario);
        end
        clearSector(app);
        app.Vert.orig = [0, 0];
        app.Vert.dest = [0, 0];
        app.Arch.orig = [0, 0];
        app.Arch.dest = [0, 0];
        app.Arch.rad = 0;
        app.Arch.dir = 'Concavo';
        app.Arch.len = 'Corto';
        app.Alt.inf = 0;
        app.Alt.sup = 0;
        app.RZTree.Tag = 'Tree';
        app.AirportTree.Tag = 'Tree';
        app.Airport.Name = '';
        app.Airport.X = 0;
```

```
            app.Airport.Y = 0;
            app.Airport.Runway = [];
            app.WaypointTree.Tag = 'Tree';
            app.Waypoint.Name = '';
            app.Waypoint.X = 0;
            app.Waypoint.Y = 0;
            app.Waypoint.Freq = [];
            clearAirplane(app);
            app.RouteWP = '';
            app.TimeSector.init = [];
            app.TimeSector.fin = [];
            app.FLSRButton.UserData = 1;
            app.FLSRLamp.UserData = 1;
            app.TPWPEF.UserData = 1;
        end

        % Button pushed function: EditordeEscenariosButton
        function EditordeEscenariosButtonPushed(app, event)
            app.PantalladeInicio.Visible = 'off';
            app.Editorseleccion.Visible = 'on';
        end

        % Button pushed function: SimuladorButton
        function SimuladorButtonPushed(app, event)
            app.PantalladeInicio.Visible = 'off';
            app.Simuladorseleccion.Visible = 'on';
        end

        % Button pushed function: Editorseleccionbackbutton
        function EditorseleccionbackbuttonButtonPushed(app, event)
            app.Editorseleccion.Visible = 'off';
            app.PantalladeInicio.Visible = 'on';
        end

        % Button pushed function: Simuladorseleccionbackcutton
        function SimuladorseleccionbackcuttonButtonPushed(app, event)
            app.Simuladorseleccion.Visible = 'off';
            app.PantalladeInicio.Visible = 'on';
        end

        % Selection changed function: TreeEditor
        function TreeEditorSelectionChanged(app, event)
            selectedNode = app.TreeEditor.SelectedNodes;
            app.SectorEditorSelec = selectedNode.Text;
        end

        % Button pushed function: NuevoEscenarioButton
        function NuevoEscenarioButtonPushed(app, event)
            app.NuevoEscenario.Visible = 'on';
            app.NombredelnuevoescenarioEditField.Value = '';
            app.TreeEditor.Enable = 'off';
            app.Editorseleccionbackbutton.Enable = 'off';
            app.NuevoEscenarioButton.Enable = 'off';
            app.RenombrarEscenarioButton.Enable = 'off';
            app.EliminarEscenarioButton.Enable = 'off';
            app.CargarEscenarioButton.Enable = 'off';
        end
```

```
1885
1886        % Value changed function: NombredelnuevoescenarioEditField
1887        function NombredelnuevoescenarioEditFieldValueChanged(app, event)
1888            app.NombreNuevoEscenario = app.NombredelnuevoescenarioEditField.
                    Value;
1889
1890        end
1891
1892        % Button pushed function: AtrsButton
1893        function AtrsButtonPushed(app, event)
1894            app.NuevoEscenario.Visible = 'off';
1895            app.TreeEditor.Enable = 'on';
1896            app.NuevoEscenarioButton.Enable = 'on';
1897            app.Editorseleccionbackbutton.Enable = 'on';
1898            app.RenombrarEscenarioButton.Enable = 'on';
1899            app.EliminarEscenarioButton.Enable = 'on';
1900            app.CargarEscenarioButton.Enable = 'on';
1901        end
1902
1903        % Button pushed function: CrearButton
1904        function CrearButtonPushed(app, event)
1905            app.S.F(size(app.S.F,2)+1) = {app.NombreNuevoEscenario};
1906            F = app.S.F;
1907            save 'libapp\escenarioslist.mat' F;
1908            uitreenode(app.TreeEditor,'Text',app.NombreNuevoEscenario);
1909            uitreenode(app.TreeSimulador,'Text',app.NombreNuevoEscenario);
1910            app.NuevoEscenario.Visible = 'off';
1911            app.TreeEditor.Enable = 'on';
1912            app.NuevoEscenarioButton.Enable = 'on';
1913            app.Editorseleccionbackbutton.Enable = 'on';
1914            app.RenombrarEscenarioButton.Enable = 'on';
1915            app.EliminarEscenarioButton.Enable = 'on';
1916            app.CargarEscenarioButton.Enable = 'on';
1917            clearSector(app)
1918            app.SectorEditor.Sector.Name = {app.NombreNuevoEscenario};
1919            sect = uint16(char(app.NombreNuevoEscenario));
1920            archivo = char([app.libapp,sect,app.pmat]);
1921            Sector = app.SectorEditor.Sector;
1922            save(archivo,'Sector');
1923            app.NombreNuevoEscenario = [];
1924            app.Editorseleccion.Visible = 'off';
1925            app.EditordeEscenarios.Visible = 'on';
1926        end
1927
1928        % Button pushed function: RenombrarEscenarioButton
1929        function RenombrarEscenarioButtonPushed(app, event)
1930            selectednode = app.TreeEditor.SelectedNodes;
1931            if ~isempty(selectednode)
1932                app.RenombrarEscenario.Visible = 'on';
1933                app.TreeEditor.Enable = 'off';
1934                app.Editorseleccionbackbutton.Enable = 'off';
1935                app.NuevoEscenarioButton.Enable = 'off';
1936                app.RenombrarEscenarioButton.Enable = 'off';
1937                app.EliminarEscenarioButton.Enable = 'off';
1938                app.CargarEscenarioButton.Enable = 'off';
1939                app.NombrenuevodelnuevoescenarioEditField.Value = selectednode.
                        Text;
```

```
            end

        end

        % Value changed function:
        % NombrenuevodelnuevoescenarioEditField
        function NombrenuevodelnuevoescenarioEditFieldValueChanged(app, event)
            app.NombreNuevoEscenario = app.NombrenuevodelnuevoescenarioEditField.
                Value;

        end

        % Button pushed function: RenombrarButton
        function RenombrarButtonPushed(app, event)
            selectednode = app.TreeEditor.SelectedNodes;
            Node = selectednode.Text;
            sect = uint16(char(Node));
            archivo = char([app.libapp,sect,app.pmat]);
            L = load(archivo);
            L.Sector.Name = {app.NombreNuevoEscenario};
            sect = uint16(char(app.NombreNuevoEscenario));
            archivo = char([app.libapp,sect,app.pmat]);
            Sector = L.Sector;
            save(archivo,'Sector');
            sect = uint16(char(Node));
            archivo = char([app.libapp,sect,app.pmat]);
            delete(archivo);
            u = strfind(app.S.F,Node);
            for j = 1:length(app.S.F)
                if isempty(u{:,j})
                else
                    idnode = j;
                end
            end
            app.S.F(idnode) = {app.NombreNuevoEscenario};
            F = app.S.F;
            save 'libapp\escenarioslist.mat' F;
            app.TreeEditor.SelectedNodes.Text = char(app.NombreNuevoEscenario);
            app.TreeSimulador.Children(idnode).Text = char(app.
                NombreNuevoEscenario);
            app.RenombrarEscenario.Visible = 'off';
            app.TreeEditor.Enable = 'on';
            app.NuevoEscenarioButton.Enable = 'on';
            app.Editorseleccionbackbutton.Enable = 'on';
            app.RenombrarEscenarioButton.Enable = 'on';
            app.NombreNuevoEscenario = [];
        end

        % Button pushed function: AtrsButton_2
        function AtrsButton_2Pushed(app, event)
            app.RenombrarEscenario.Visible = 'off';
            app.TreeEditor.Enable = 'on';
            app.NuevoEscenarioButton.Enable = 'on';
            app.Editorseleccionbackbutton.Enable = 'on';
            app.RenombrarEscenarioButton.Enable = 'on';
            app.EliminarEscenarioButton.Enable = 'on';
            app.CargarEscenarioButton.Enable = 'on';
```

```
1995        end
1996
1997        % Button pushed function: Editorbackbutton
1998        function EditorbackbuttonButtonPushed(app, event)
1999            app.EditordeEscenarios.Visible = 'off';
2000            clearSector(app);
2001            app.SectorEditor.Sector.Border.X = {0};
2002            app.SectorEditor.Sector.Border.Y = {0};
2003            plotEditorgraph(app);
2004            hold(app.Editorgraph,"off")
2005            clearSector(app);
2006            app.TreeSector.SelectedNodes = [];
2007            app.TreeSectorSelectionChanged(app);
2008            clearSelectedNodesEditor(app);
2009            clearTreesEditor(app);
2010            clearButtonGroupsEditor(app);
2011            app.Editorseleccion.Visible = 'on';
2012        end
2013
2014        % Button pushed function: EliminarEscenarioButton
2015        function EliminarEscenarioButtonPushed(app, event)
2016            selectednode = app.TreeEditor.SelectedNodes;
2017            Node = selectednode.Text;
2018            delete(app.TreeEditor.SelectedNodes);
2019            u = strfind(app.S.F,Node);
2020            for j = 1:length(app.S.F)
2021                if isempty(u{:,j})
2022                else
2023                    idnode = j;
2024                end
2025            end
2026            delete(app.TreeSimulador.Children(idnode));
2027            app.S.F(idnode) = [];
2028            F = app.S.F;
2029            save 'libapp\escenarioslist.mat' F;
2030            sect = uint16(char(Node));
2031            archivo = char([app.libapp,sect,app.pmat]);
2032            delete(archivo);
2033        end
2034
2035        % Button pushed function: CargarEscenarioButton
2036        function CargarEscenarioButtonPushed(app, event)
2037            selectednode = app.TreeEditor.SelectedNodes;
2038            if ~isempty(selectednode)
2039                sect = uint16(char(selectednode.Text));
2040                archivo = char([app.libapp,sect,app.pmat]);
2041                app.SectorEditor = load(archivo);
2042                LoadScenario(app);
2043                app.Editorseleccion.Visible = 'off';
2044                app.EditordeEscenarios.Visible = 'on';
2045                app.TreeEditor.SelectedNodes = [];
2046            end
2047        end
2048
2049        % Selection changed function: TreeSector
2050        function TreeSectorSelectionChanged(app, event)
2051            selectedNodes = app.TreeSector.SelectedNodes;
```

```
            if isempty(selectedNodes)
                app.SectorBorderPanel.Visible = 'off';
                app.SectorRestricZonePanel.Visible = 'off';
                app.SectorAirplanePanel.Visible = 'off';
                app.SectorAirportPanel.Visible = 'off';
                app.SectorWaypointPanel.Visible = 'off';
            else
                if selectedNodes.NodeData == 1
                    app.SectorBorderPanel.Visible = 'on';
                    app.SectorRestricZonePanel.Visible = 'off';
                    app.SectorAirplanePanel.Visible = 'off';
                    app.SectorAirportPanel.Visible = 'off';
                    app.SectorWaypointPanel.Visible = 'off';
                elseif selectedNodes.NodeData == 2
                    app.SectorBorderPanel.Visible = 'off';
                    app.SectorRestricZonePanel.Visible = 'on';
                    app.SectorAirplanePanel.Visible = 'off';
                    app.SectorAirportPanel.Visible = 'off';
                    app.SectorWaypointPanel.Visible = 'off';
                elseif selectedNodes.NodeData == 3
                    app.SectorBorderPanel.Visible = 'off';
                    app.SectorRestricZonePanel.Visible = 'off';
                    app.SectorAirplanePanel.Visible = 'on';
                    app.SectorAirportPanel.Visible = 'off';
                    app.SectorWaypointPanel.Visible = 'off';
                elseif selectedNodes.NodeData == 4
                    app.SectorBorderPanel.Visible = 'off';
                    app.SectorRestricZonePanel.Visible = 'off';
                    app.SectorAirplanePanel.Visible = 'off';
                    app.SectorAirportPanel.Visible = 'on';
                    app.SectorWaypointPanel.Visible = 'off';
                elseif selectedNodes.NodeData == 5
                    app.SectorBorderPanel.Visible = 'off';
                    app.SectorRestricZonePanel.Visible = 'off';
                    app.SectorAirplanePanel.Visible = 'off';
                    app.SectorAirportPanel.Visible = 'off';
                    app.SectorWaypointPanel.Visible = 'on';
                end
            end
            clearSelectedNodesEditor(app)
        end

        % Selection changed function: TipodeFronteraButtonGroup
        function TipodeFronteraButtonGroupSelectionChanged(app, event)
            if app.SeleccionaunaButton.Value
                app.BorderVertexPanel.Visible = 'off';
                app.BorderArchPanel.Visible = 'off';
                app.BorderAltitudePanel.Visible = 'off';
            elseif app.VerticeButton.Value
                app.BorderVertexPanel.Visible = 'on';
                app.BorderArchPanel.Visible = 'off';
                app.BorderAltitudePanel.Visible = 'off';
            elseif app.ArcoButton.Value
                app.BorderVertexPanel.Visible = 'off';
                app.BorderArchPanel.Visible = 'on';
                app.BorderAltitudePanel.Visible = 'off';
            elseif app.AltitudButton.Value
```

```
2109                app.BorderVertexPanel.Visible = 'off';
2110                app.BorderArchPanel.Visible = 'off';
2111                app.BorderAltitudePanel.Visible = 'on';
2112            end
2113            BorderTreeSelectionChanged(app)
2114        end
2115
2116        % Value changed function: XnvertEditField
2117        function XnvertEditFieldValueChanged(app, event)
2118            app.Vert.orig(1,1) = app.XnvertEditField.Value;
2119        end
2120
2121        % Value changed function: YnvertEditField
2122        function YnvertEditFieldValueChanged(app, event)
2123            app.Vert.orig(1,2) = app.YnvertEditField.Value;
2124        end
2125
2126        % Value changed function: Xn1vertEditField
2127        function Xn1vertEditFieldValueChanged(app, event)
2128            app.Vert.dest(1,1) = app.Xn1vertEditField.Value;
2129        end
2130
2131        % Value changed function: Yn1vertEditField
2132        function Yn1vertEditFieldValueChanged(app, event)
2133            app.Vert.dest(1,2) = app.Yn1vertEditField.Value;
2134        end
2135
2136        % Button pushed function: MostrarBorderVButton
2137        function MostrarBorderVButtonPushed(app, event)
2138            plotEditorgraph(app)
2139            if ~isempty(app.Vert)
2140                X = [app.Vert.orig(1,1), app.Vert.dest(1,1)];
2141                Y = [app.Vert.orig(1,2), app.Vert.dest(1,2)];
2142                plot(app.Editorgraph,X,Y,'-g')
2143            end
2144            hold(app.Editorgraph,"off")
2145        end
2146
2147        % Button pushed function: GuardarBorderVButton
2148        function GuardarBorderVButtonPushed(app, event)
2149            if ~isempty(app.Vert)
2150                if isempty(app.SectorEditor.Sector.Border)
2151                    app.SectorEditor.Sector.Border.X{1} = app.Vert.orig(1,1);
2152                    app.SectorEditor.Sector.Border.X{2} = app.Vert.dest(1,1);
2153                    app.SectorEditor.Sector.Border.Y{1} = app.Vert.orig(1,2);
2154                    app.SectorEditor.Sector.Border.Y{2} = app.Vert.dest(1,2);
2155                    uitreenode(app.BorderTree,"NodeData",1,"Text",'Vertice 1');
2156                    uitreenode(app.BorderTree,"NodeData",2,"Text",'Vertice 2');
2157                elseif ~isempty(app.BorderTree.SelectedNodes)
2158                    t = size(app.SectorEditor.Sector.Border.X,2);
2159                    d = app.BorderTree.SelectedNodes.NodeData + 1;
2160                    for i = t+1:-1:d
2161                        if i ~= d
2162                            app.SectorEditor.Sector.Border.X{i} = app.SectorEditor
                                    .Sector.Border.X{i-1};
2163                            app.SectorEditor.Sector.Border.Y{i} = app.SectorEditor
                                    .Sector.Border.Y{i-1};
```

```
2164                        T = app.BorderTree.Children(i-1).Text(1);
2165                        CharNum = dec2char(app,i);
2166                        if T == 'V'
2167                            app.BorderTree.Children(i-1).Text = char(['Vértice
                                    ',CharNum]);
2168                            app.BorderTree.Children(i-1).NodeData = i;
2169                        elseif T == 'A'
2170                            app.BorderTree.Children(i-1).Text = char(['Arco ',
                                   CharNum]);
2171                            app.BorderTree.Children(i-1).NodeData = i;
2172                        end
2173                    elseif i == d
2174                        app.SectorEditor.Sector.Border.X{i} = app.Vert.dest
                               (1,1);
2175                        app.SectorEditor.Sector.Border.Y{i} = app.Vert.dest
                               (1,2);
2176                        CharNum = dec2char(app,d);
2177                        Node = uitreenode(app.BorderTree,'NodeData',d,'Text',
                               char(['Vértice ',CharNum]));
2178                        move(Node,app.BorderTree.Children(d-1));
2179                    end
2180                end
2181            else
2182                t = size(app.SectorEditor.Sector.Border.X,2);
2183                app.SectorEditor.Sector.Border.X{t+1} = app.Vert.dest(1,1);
2184                app.SectorEditor.Sector.Border.Y{t+1} = app.Vert.dest(1,2);
2185                CharNum = dec2char(app,t+1);
2186                uitreenode(app.BorderTree,"NodeData",t+1,"Text",char(['
                       Vertice ',CharNum]));
2187            end
2188            resetBorderXYeditfieldvert(app)
2189            updateBorderVert(app)
2190            plotEditorgraph(app)
2191            hold(app.Editorgraph,"off")
2192            app.BorderTree.SelectedNodes = [];
2193            BorderTreeSelectionChanged(app)
2194        end
2195    end
2196
2197    % Selection changed function: BorderTree
2198    function BorderTreeSelectionChanged(app, event)
2199        selectedNodes = app.BorderTree.SelectedNodes;
2200        if ~isempty(selectedNodes)
2201            if selectedNodes.Text(1) == 'V'
2202                if app.VerticeButton.Value
2203                    if selectedNodes.NodeData < size(app.SectorEditor.Sector.
                           Border.X,2)
2204                        app.XnvertEditField.Value = app.SectorEditor.Sector.
                               Border.X{selectedNodes.NodeData}(1);
2205                        app.YnvertEditField.Value = app.SectorEditor.Sector.
                               Border.Y{selectedNodes.NodeData}(1);
2206                        app.Xn1vertEditField.Value = app.SectorEditor.Sector.
                               Border.X{selectedNodes.NodeData+1}(1);
2207                        app.Yn1vertEditField.Value = app.SectorEditor.Sector.
                               Border.Y{selectedNodes.NodeData+1}(1);
2208                    elseif selectedNodes.NodeData == size(app.SectorEditor.
                           Sector.Border.X,2)
```

```
2209                            resetBorderXYeditfieldvert(app)
2210                            app.XnvertEditField.Value = app.SectorEditor.Sector.
                                    Border.X{selectedNodes.NodeData};
2211                            app.YnvertEditField.Value = app.SectorEditor.Sector.
                                    Border.Y{selectedNodes.NodeData};
2212                        end
2213                        updateBorderVert(app)
2214                    elseif app.ArcoButton.Value
2215                        resetBordereditfieldarch(app)
2216                        if selectedNodes.NodeData < size(app.SectorEditor.Sector.
                                Border.X,2)
2217                            app.XnarchEditField.Value = app.SectorEditor.Sector.
                                    Border.X{selectedNodes.NodeData}(1);
2218                            app.YnarchEditField.Value = app.SectorEditor.Sector.
                                    Border.Y{selectedNodes.NodeData}(1);
2219                            app.Xn1archEditField.Value = app.SectorEditor.Sector.
                                    Border.X{selectedNodes.NodeData+1}(1);
2220                            app.Yn1archEditField.Value = app.SectorEditor.Sector.
                                    Border.Y{selectedNodes.NodeData+1}(1);
2221                        elseif selectedNodes.NodeData == size(app.SectorEditor.
                                Sector.Border.X,2)
2222                            app.XnarchEditField.Value = app.SectorEditor.Sector.
                                    Border.X{selectedNodes.NodeData}(1);
2223                            app.YnarchEditField.Value = app.SectorEditor.Sector.
                                    Border.Y{selectedNodes.NodeData}(1);
2224                        end
2225                        updateBorderArch(app)
2226                    end
2227                elseif selectedNodes.Text(1) == 'A'
2228                    if app.ArcoButton.Value
2229                        app.XnarchEditField.Value = app.SectorEditor.Sector.
                                Border.X{selectedNodes.NodeData}(1);
2230                        app.YnarchEditField.Value = app.SectorEditor.Sector.
                                Border.Y{selectedNodes.NodeData}(1);
2231                        app.Xn1archEditField.Value = app.SectorEditor.Sector.
                                Border.X{selectedNodes.NodeData}(end);
2232                        app.Yn1archEditField.Value = app.SectorEditor.Sector.
                                Border.Y{selectedNodes.NodeData}(end);
2233                        R = CalcRad(app,app.SectorEditor.Sector.Border.X{
                                selectedNodes.NodeData},app.SectorEditor.Sector.
                                Border.Y{selectedNodes.NodeData});
2234                        app.BARadiusEditField.Value = R;
2235                        [Value,len] = ConcaveConvexShortLongcheck(app,app.
                                SectorEditor.Sector.Border.X{selectedNodes.NodeData},
                                app.SectorEditor.Sector.Border.Y{selectedNodes.
                                NodeData},R);
2236                        app.BAConcaveConvexSwitch.Value = Value;
2237                        app.BACortoLargoSwitch.Value = len;
2238                        updateBorderArch(app)
2239                    elseif app.VerticeButton.Value
2240                        resetBorderXYeditfieldvert(app)
2241                        app.XnvertEditField.Value = app.SectorEditor.Sector.
                                Border.X{selectedNodes.NodeData}(end);
2242                        app.YnvertEditField.Value = app.SectorEditor.Sector.
                                Border.Y{selectedNodes.NodeData}(end);
2243                        updateBorderVert(app)
2244                    end
```

```
2245                end
2246                plotEditorgraph(app)
2247                if selectedNodes.Text(1) == 'V'
2248                    Xb = app.SectorEditor.Sector.Border.X{selectedNodes.NodeData
                        };
2249                    Yb = app.SectorEditor.Sector.Border.Y{selectedNodes.NodeData
                        };
2250                elseif selectedNodes.Text(1) == 'A'
2251                    X = app.SectorEditor.Sector.Border.X{selectedNodes.NodeData};
2252                    Y = app.SectorEditor.Sector.Border.Y{selectedNodes.NodeData};
2253                    A = [X(1),Y(1)];
2254                    B = [X(end), Y(end)];
2255                    Arco = [X',Y'];
2256                    I = encuentraI(app,Arco,A,B);
2257                    Xb = [A(1), I(1), B(1)];
2258                    Yb = [A(2), I(2), B(2)];
2259                end
2260                plot(app.Editorgraph,Xb,Yb,'og')
2261                hold(app.Editorgraph,"off")
2262            elseif isempty(selectedNodes) && ~isempty(app.BorderTree.Children)
2263                if app.VerticeButton.Value
2264                    resetBorderXYeditfieldvert(app)
2265                    app.XnvertEditField.Value = app.SectorEditor.Sector.Border.X{
                        end}(end);
2266                    app.YnvertEditField.Value = app.SectorEditor.Sector.Border.Y{
                        end}(end);
2267                    updateBorderVert(app)
2268                elseif app.ArcoButton.Value
2269                    resetBordereditfieldarch(app)
2270                    app.XnarchEditField.Value = app.SectorEditor.Sector.Border.X{
                        end}(end);
2271                    app.YnarchEditField.Value = app.SectorEditor.Sector.Border.Y{
                        end}(end);
2272                    updateBorderArch(app)
2273                end
2274            end
2275        end
2276
2277        % Button pushed function: EliminarBorderButton
2278        function EliminarBorderButtonPushed(app, event)
2279            if ~isempty(app.BorderTree.SelectedNodes)
2280                t = app.BorderTree.SelectedNodes.NodeData;
2281                if size(app.BorderTree.Children,1) > 1
2282                    for i = 1:size(app.BorderTree.Children,1)
2283                        if i > t
2284                            app.SectorEditor.Sector.Border.X{i-1} = app.
                                SectorEditor.Sector.Border.X{i};
2285                            app.SectorEditor.Sector.Border.Y{i-1} = app.
                                SectorEditor.Sector.Border.Y{i};
2286                            T = app.BorderTree.Children(i).Text;
2287                            CharNum = dec2char(app,i-1);
2288                            if T(1) == 'V'
2289                                app.BorderTree.Children(i-1).Text = char(['Vértice
                                    ',CharNum]);
2290                            elseif T(1) == 'A'
2291                                app.BorderTree.Children(i-1).Text = char(['Arco ',
                                    CharNum]);
```

```
                        end
                    end
                end
                X = cell([size(app.SectorEditor.Sector.Border.X,1),size(app.
                    SectorEditor.Sector.Border.X,2)-1]);
                Y = cell([size(app.SectorEditor.Sector.Border.Y,1),size(app.
                    SectorEditor.Sector.Border.Y,2)-1]);
                for j = 1:size(app.SectorEditor.Sector.Border.X,2)-1
                    X{j} = app.SectorEditor.Sector.Border.X{j};
                    Y{j} = app.SectorEditor.Sector.Border.Y{j};
                end
                app.SectorEditor.Sector.Border.X = X;
                app.SectorEditor.Sector.Border.Y = Y;
                delete(app.BorderTree.Children(end))
            elseif size(app.BorderTree.Children) == 1
                app.SectorEditor.Sector.Border.X = {};
                app.SectorEditor.Sector.Border.Y = {};
                delete(app.BorderTree.Children(1))
            end
            resetBorderXYeditfieldvert(app)
            updateBorderVert(app)
            app.BorderTree.SelectedNodes = [];
            BorderTreeSelectionChanged(app)
            plotEditorgraph(app)
            hold(app.Editorgraph,"off")
        end
    end

    % Value changed function: XnarchEditField
    function XnarchEditFieldValueChanged(app, event)
        app.Arch.orig(1,1) = app.XnarchEditField.Value;
    end

    % Value changed function: YnarchEditField
    function YnarchEditFieldValueChanged(app, event)
        app.Arch.orig(1,2) = app.YnarchEditField.Value;
    end

    % Value changed function: Xn1archEditField
    function Xn1archEditFieldValueChanged(app, event)
        app.Arch.dest(1,1) = app.Xn1archEditField.Value;
    end

    % Value changed function: Yn1archEditField
    function Yn1archEditFieldValueChanged(app, event)
        app.Arch.dest(1,2) = app.Yn1archEditField.Value;
    end

    % Value changed function: BARadiusEditField
    function BARadiusEditFieldValueChanged(app, event)
        app.Arch.rad = app.BARadiusEditField.Value;
    end

    % Value changed function: BAConcaveConvexSwitch
    function BAConcaveConvexSwitchValueChanged(app, event)
        dir = app.BAConcaveConvexSwitch.Value;
        if dir == '^'
```

```
2347                app.Arch.dir = 'Concavo';
2348            elseif dir == 'v'
2349                app.Arch.dir = 'Convexo';
2350            end
2351        end
2352
2353        % Button pushed function: MostrarBorderAButton
2354        function MostrarBorderAButtonPushed(app, event)
2355            plotEditorgraph(app)
2356            if ~isempty(app.Arch)
2357                if app.Arch.orig(1) ~= app.Arch.dest(1) || app.Arch.orig(2) ~=
                        app.Arch.dest(2)
2358                    if app.Arch.rad >= 0.5*norm(app.Arch.dest-app.Arch.orig)
2359                        Centro = Calccentre(app,app.Arch);
2360                        [X,Y] = Calcarch(app,app.Arch,Centro);
2361                    elseif app.Arch.rad >= 0
2362                        X = [app.Arch.orig(1), app.Arch.dest(1)];
2363                        Y = [app.Arch.orig(2), app.Arch.dest(2)];
2364                    end
2365                    plot(app.Editorgraph,X,Y,'-g')
2366                end
2367            end
2368            hold(app.Editorgraph,"off")
2369        end
2370
2371        % Value changed function: BACortoLargoSwitch
2372        function BACortoLargoSwitchValueChanged(app, event)
2373            app.Arch.len = app.BACortoLargoSwitch.Value;
2374        end
2375
2376        % Button pushed function: GuardarBorderAButton
2377        function GuardarBorderAButtonPushed(app, event)
2378            if ~isempty(app.Arch)
2379                if app.Arch.orig(1) ~= app.Arch.dest(1) || app.Arch.orig(2) ~=
                        app.Arch.dest(2)
2380                    if app.Arch.rad >= 0.5*norm(app.Arch.dest-app.Arch.orig)
2381                        Centro = Calccentre(app,app.Arch);
2382                        [X,Y] = Calcarch(app,app.Arch,Centro);
2383                    elseif app.Arch.rad >= 0
2384                        X = [app.Arch.orig(1), app.Arch.dest(1)];
2385                        Y = [app.Arch.orig(2), app.Arch.dest(2)];
2386                    end
2387                elseif app.Arch.orig(1) == app.Arch.dest(1) && app.Arch.orig(2)
                        == app.Arch.dest(2)
2388                    X = [];
2389                    Y = [];
2390                end
2391                if ~isempty(X)
2392                    if isempty(app.SectorEditor.Sector.Border)
2393                        app.SectorEditor.Sector.Border.X{1} = X;
2394                        app.SectorEditor.Sector.Border.Y{1} = Y;
2395                        uitreenode(app.BorderTree,"NodeData",1,"Text",'Arco 1');
2396                    elseif ~isempty(app.BorderTree.SelectedNodes)
2397                        t = size(app.SectorEditor.Sector.Border.X,2);
2398                        d = app.BorderTree.SelectedNodes.NodeData + 1;
2399                        for i = t+1:-1:d
2400                            if i ~= d
```

```
2401                            app.SectorEditor.Sector.Border.X{i} = app.
                                    SectorEditor.Sector.Border.X{i-1};
2402                            app.SectorEditor.Sector.Border.Y{i} = app.
                                    SectorEditor.Sector.Border.Y{i-1};
2403                            T = app.BorderTree.Children(i-1).Text(1);
2404                            CharNum = dec2char(app,i);
2405                            if T == 'V'
2406                                app.BorderTree.Children(i-1).Text = char(['Vé
                                        rtice ',CharNum]);
2407                                app.BorderTree.Children(i-1).NodeData = i;
2408                            elseif T == 'A'
2409                                app.BorderTree.Children(i-1).Text = char(['
                                        Arco ',CharNum]);
2410                                app.BorderTree.Children(i-1).NodeData = i;
2411                            end
2412                        elseif i == d
2413                            app.SectorEditor.Sector.Border.X{i} = X;
2414                            app.SectorEditor.Sector.Border.Y{i} = Y;
2415                            CharNum = dec2char(app,d);
2416                            Node = uitreenode(app.BorderTree,'NodeData',d,'
                                    Text',char(['Arco ',CharNum]));
2417                            move(Node,app.BorderTree.Children(d-1));
2418                        end
2419                    end
2420                else
2421                    t = size(app.SectorEditor.Sector.Border.X,2);
2422                    app.SectorEditor.Sector.Border.X{t+1} = X;
2423                    app.SectorEditor.Sector.Border.Y{t+1} = Y;
2424                    CharNum = dec2char(app,t+1);
2425                    uitreenode(app.BorderTree,"NodeData",t+1,"Text",char(['
                            Arco ',CharNum]));
2426                end
2427                resetBordereditfieldarch(app)
2428                updateBorderArch(app)
2429                plotEditorgraph(app)
2430                hold(app.Editorgraph,"off")
2431            end
2432            app.BorderTree.SelectedNodes = [];
2433            BorderTreeSelectionChanged(app)
2434        end
2435    end
2436
2437    % Value changed function: ZinfEditField
2438    function ZinfEditFieldValueChanged(app, event)
2439        if app.FLftinfDropDown.Value == '1'
2440            app.Alt.inf = app.ZinfEditField.Value * 100;
2441        elseif app.FLftinfDropDown.Value == '2'
2442            app.Alt.inf = app.ZinfEditField.Value;
2443        end
2444    end
2445
2446    % Value changed function: ZsupEditField
2447    function ZsupEditFieldValueChanged(app, event)
2448        if app.FLftsupDropDown.Value == '1'
2449            app.Alt.sup = app.ZsupEditField.Value * 100;
2450        elseif app.FLftsupDropDown.Value == '2'
2451            app.Alt.sup = app.ZsupEditField.Value;
```

```
2452            end
2453        end
2454
2455        % Button pushed function: GuardarBorderAltButton
2456        function GuardarBorderAltButtonPushed(app, event)
2457            if app.Alt.inf >= 0 && app.Alt.sup > app.Alt.inf
2458                app.SectorEditor.Sector.Border.Z(1) = app.Alt.sup;
2459                app.SectorEditor.Sector.Border.Z(2) = app.Alt.inf;
2460                if app.Alt.inf >= 1000
2461                    FLcharsup = FLChar(app,app.Alt.sup);
2462                    FLcharinf = FLChar(app,app.Alt.inf);
2463                    app.FLsupEditField.Value = FLcharsup;
2464                    app.FLinfEditField.Value = FLcharinf;
2465                elseif app.Alt.inf < 1000 && app.Alt.sup >= 1000
2466                    FLcharsup = FLChar(app,app.Alt.sup);
2467                    ftcharinf = ftChar(app,app.Alt.inf);
2468                    app.FLsupEditField.Value = FLcharsup;
2469                    app.FLinfEditField.Value = ftcharinf;
2470                elseif app.Alt.sup < 1000
2471                    ftcharsup = ftChar(app,app.Alt.sup);
2472                    ftcharinf = ftChar(app,app.Alt.inf);
2473                    app.FLsupEditField.Value = ftcharsup;
2474                    app.FLinfEditField.Value = ftcharinf;
2475                end
2476                resetBordereditfieldalt(app)
2477                updateBorderAlt(app)
2478            end
2479        end
2480
2481        % Selection changed function: TipodeFronteraRZButtonGroup
2482        function TipodeFronteraRZButtonGroupSelectionChanged(app, event)
2483            if app.SeleccionaunaButtonRZ.Value
2484                app.RZVertexPanel.Visible = 'off';
2485                app.RZArchPanel.Visible = 'off';
2486                app.RZAltitudePanel.Visible = 'off';
2487            elseif app.VerticeButtonRZ.Value
2488                app.RZVertexPanel.Visible = 'on';
2489                app.RZArchPanel.Visible = 'off';
2490                app.RZAltitudePanel.Visible = 'off';
2491            elseif app.ArcoButtonRZ.Value
2492                app.RZVertexPanel.Visible = 'off';
2493                app.RZArchPanel.Visible = 'on';
2494                app.RZAltitudePanel.Visible = 'off';
2495            elseif app.AltitudButtonRZ.Value
2496                app.RZVertexPanel.Visible = 'off';
2497                app.RZArchPanel.Visible = 'off';
2498                app.RZAltitudePanel.Visible = 'on';
2499            end
2500            %RZTreeSelectionChanged(app)
2501        end
2502
2503        % Value changed function: RZXnvertEditField
2504        function RZXnvertEditFieldValueChanged(app, event)
2505            app.Vert.orig(1,1) = app.RZXnvertEditField.Value;
2506        end
2507
2508        % Value changed function: RZYnvertEditField
```

```
2509        function RZYnvertEditFieldValueChanged(app, event)
2510            app.Vert.orig(1,2) = app.RZYnvertEditField.Value;
2511        end
2512
2513        % Value changed function: RZXn1vertEditField
2514        function RZXn1vertEditFieldValueChanged(app, event)
2515            app.Vert.dest(1,1) = app.RZXn1vertEditField.Value;
2516        end
2517
2518        % Value changed function: RZYn1vertEditField
2519        function RZYn1vertEditFieldValueChanged(app, event)
2520            app.Vert.dest(1,2) = app.RZYn1vertEditField.Value;
2521        end
2522
2523        % Button pushed function: MostrarRZVButton
2524        function MostrarRZVButtonPushed(app, event)
2525            plotEditorgraph(app)
2526            if ~isempty(app.Vert)
2527                X = [app.Vert.orig(1,1), app.Vert.dest(1,1)];
2528                Y = [app.Vert.orig(1,2), app.Vert.dest(1,2)];
2529                plot(app.Editorgraph,X,Y,'-r')
2530            end
2531            hold(app.Editorgraph,"off")
2532        end
2533
2534        % Button pushed function: GuardarRZVButton
2535        function GuardarRZVButtonPushed(app, event)
2536            if ~isempty(app.Vert)
2537                selNode = app.RZTree.SelectedNodes;
2538                if ~isempty(selNode)
2539                    if selNode.Parent.Tag(1) == 'T'
2540                        k = selNode.NodeData;
2541                        if isempty(selNode.Children)
2542                           d = 1;
2543                        else
2544                           d = selNode.Children(end).NodeData + 1;
2545                        end
2546                    elseif selNode.Parent.Tag(1) == 'N'
2547                        k = selNode.Parent.NodeData;
2548                        d = selNode.NodeData + 1;
2549                    end
2550                elseif ~isempty(app.RZTree.Children)
2551                    k = size(app.RZTree.Children,1);
2552                    d = size(app.RZTree.Children(k).Children,1) + 1;
2553                else
2554                    k = 0;
2555                end
2556                if k > 0
2557                    if isempty(app.SectorEditor.Sector.RestricZones(k).X)
2558                        app.SectorEditor.Sector.RestricZones(k).X{1} = app.Vert.
                                orig(1,1);
2559                        app.SectorEditor.Sector.RestricZones(k).X{2} = app.Vert.
                                dest(1,1);
2560                        app.SectorEditor.Sector.RestricZones(k).Y{1} = app.Vert.
                                orig(1,2);
2561                        app.SectorEditor.Sector.RestricZones(k).Y{2} = app.Vert.
                                dest(1,2);
```

```
2562                        uitreenode(app.RZTree.Children(k),"NodeData",1,"Text",'
                                Vertice 1');
2563                        uitreenode(app.RZTree.Children(k),"NodeData",2,"Text",'
                                Vertice 2');
2564                    elseif ~isempty(selNode)
2565                        t = size(app.SectorEditor.Sector.RestricZones(k).X,2);
2566                        for i = t+1:-1:d
2567                            if i ~= d
2568                                app.SectorEditor.Sector.RestricZones(k).X{i} = app.
                                        SectorEditor.Sector.RestricZones(k).X{i-1};
2569                                app.SectorEditor.Sector.RestricZones(k).Y{i} = app.
                                        SectorEditor.Sector.RestricZones(k).Y{i-1};
2570                                T = app.RZTree.Children(k).Children(i-1).Text(1);
2571                                CharNum = dec2char(app,i);
2572                                if T == 'V'
2573                                    app.RZTree.Children(k).Children(i-1).Text =
                                            char(['Vértice ',CharNum]);
2574                                    app.RZTree.Children(k).Children(i-1).NodeData
                                            = i;
2575                                elseif T == 'A'
2576                                    app.RZTree.Children(k).Children(i-1).Text =
                                            char(['Arco ',CharNum]);
2577                                    app.RZTree.Children(k).Children(i-1).NodeData
                                            = i;
2578                                end
2579                            elseif i == d
2580                                app.SectorEditor.Sector.RestricZones(k).X{i} = app.
                                        Vert.dest(1,1);
2581                                app.SectorEditor.Sector.RestricZones(k).Y{i} = app.
                                        Vert.dest(1,2);
2582                                CharNum = dec2char(app,d);
2583                                Node = uitreenode(app.RZTree.Children(k),'NodeData
                                        ',d,'Text',char(['Vértice ',CharNum]));
2584                                move(Node,app.RZTree.Children(k).Children(d-1));
2585                            end
2586                        end
2587                    else
2588                        t = size(app.SectorEditor.Sector.RestricZones(k).X,2);
2589                        app.SectorEditor.Sector.RestricZones(k).X{t+1} = app.Vert.
                                dest(1,1);
2590                        app.SectorEditor.Sector.RestricZones(k).Y{t+1} = app.Vert.
                                dest(1,2);
2591                        CharNum = dec2char(app,t+1);
2592                        uitreenode(app.RZTree.Children(k),"NodeData",t+1,"Text",
                                char(['Vertice ',CharNum]));
2593                    end
2594                end
2595                resetRZXYeditfieldvert(app)
2596                updateRZVert(app)
2597                plotEditorgraph(app)
2598                hold(app.Editorgraph,"off")
2599                app.RZTree.SelectedNodes = [];
2600                RZTreeSelectionChanged(app)
2601            end
2602        end
2603
2604        % Button pushed function: NuevaZonaRestringidaButton
```

```
2605        function NuevaZonaRestringidaButtonPushed(app, event)
2606            if isempty(app.RZTree.Children)
2607                uitreenode(app.RZTree,'Text','Zona Restringida 1',"NodeData",1,'
                        Tag','Node');
2608                app.SectorEditor.Sector.RestricZones(1).X = {};
2609                app.SectorEditor.Sector.RestricZones(1).Y = {};
2610                app.SectorEditor.Sector.RestricZones(1).Z = [];
2611            else
2612                k = size(app.RZTree.Children,1) + 1;
2613                CharNum = dec2char(app,k);
2614                uitreenode(app.RZTree,'Text',char(['Zona Restringida ',CharNum])
                        ,"NodeData",k,'Tag','Node');
2615                app.SectorEditor.Sector.RestricZones(k).X = {};
2616                app.SectorEditor.Sector.RestricZones(k).Y = {};
2617                app.SectorEditor.Sector.RestricZones(k).Z = [];
2618            end
2619        end
2620
2621        % Selection changed function: RZTree
2622        function RZTreeSelectionChanged(app, event)
2623            selNode = app.RZTree.SelectedNodes;
2624            if ~isempty(selNode)
2625                if selNode.Parent.Tag(1) == 'T'
2626                    k = selNode.NodeData;
2627                    if isempty(selNode.Children)
2628                       d = 1;
2629                    else
2630                       d = selNode.Children(end).NodeData;
2631                    end
2632                elseif selNode.Parent.Tag(1) == 'N'
2633                    k = selNode.Parent.NodeData;
2634                    d = selNode.NodeData;
2635                end
2636            elseif ~isempty(app.RZTree.Children)
2637                k = size(app.RZTree.Children,1);
2638                d = size(app.RZTree.Children(k).Children,1);
2639            else
2640                k = 1;
2641                d = 1;
2642            end
2643            if ~isempty(selNode) && ~isempty(app.RZTree.Children(k).Children)
2644                if app.RZTree.Children(k).Children(d).Text(1) == 'V'
2645                    if app.VerticeButtonRZ.Value
2646                        if d < size(app.SectorEditor.Sector.RestricZones(k).X,2)
2647                            app.RZXnvertEditField.Value = app.SectorEditor.Sector.
                                    RestricZones(k).X{d}(end);
2648                            app.RZYnvertEditField.Value = app.SectorEditor.Sector.
                                    RestricZones(k).Y{d}(end);
2649                            app.RZXn1vertEditField.Value = app.SectorEditor.Sector
                                    .RestricZones(k).X{d+1}(1);
2650                            app.RZYn1vertEditField.Value = app.SectorEditor.Sector
                                    .RestricZones(k).Y{d+1}(1);
2651                        elseif d == size(app.SectorEditor.Sector.RestricZones(k).
                                X,2)
2652                            resetRZXYeditfieldvert(app)
2653                            app.RZXnvertEditField.Value = app.SectorEditor.Sector.
                                    RestricZones(k).X{d};
```

```
2654                        app.RZYnvertEditField.Value = app.SectorEditor.Sector.
                                RestricZones(k).Y{d};
2655                    end
2656                    updateRZVert(app)
2657                elseif app.ArcoButtonRZ.Value
2658                    resetRZeditfieldarch(app)
2659                    if d < size(app.SectorEditor.Sector.RestricZones(k).X,2)
2660                        app.RZXnarchEditField.Value = app.SectorEditor.Sector.
                                RestricZones(k).X{d}(end);
2661                        app.RZYnarchEditField.Value = app.SectorEditor.Sector.
                                RestricZones(k).Y{d}(end);
2662                        app.RZXn1archEditField.Value = app.SectorEditor.Sector
                                .RestricZones(k).X{d+1}(1);
2663                        app.RZYn1archEditField.Value = app.SectorEditor.Sector
                                .RestricZones(k).Y{d+1}(1);
2664                    elseif d == size(app.SectorEditor.Sector.RestricZones(k).
                            X,2)
2665                        app.RZXnarchEditField.Value = app.SectorEditor.Sector.
                                RestricZones(k).X{d}(1);
2666                        app.RZYnarchEditField.Value = app.SectorEditor.Sector.
                                RestricZones(k).Y{d}(1);
2667                    end
2668                    updateRZArch(app)
2669                end
2670            elseif app.RZTree.Children(k).Children(d).Text(1) == 'A'
2671                if app.ArcoButtonRZ.Value
2672                    app.RZXnarchEditField.Value = app.SectorEditor.Sector.
                            RestricZones(k).X{d}(1);
2673                    app.RZYnarchEditField.Value = app.SectorEditor.Sector.
                            RestricZones(k).Y{d}(1);
2674                    app.RZXn1archEditField.Value = app.SectorEditor.Sector.
                            RestricZones(k).X{d}(end);
2675                    app.RZYn1archEditField.Value = app.SectorEditor.Sector.
                            RestricZones(k).Y{d}(end);
2676                    R = CalcRad(app,app.SectorEditor.Sector.RestricZones(k).X
                            {d},app.SectorEditor.Sector.RestricZones(k).Y{d});
2677                    app.RZARadiusEditField.Value = R;
2678                    [Value,len] = ConcaveConvexShortLongcheck(app,app.
                            SectorEditor.Sector.RestricZones(k).X{d},app.
                            SectorEditor.Sector.RestricZones(k).Y{d},R);
2679                    app.RZAConcaveConvexSwitch.Value = Value;
2680                    app.RZACortoLargoSwitch.Value = len;
2681                    updateRZArch(app)
2682                elseif app.VerticeButtonRZ.Value
2683                    resetRZXYeditfieldvert(app)
2684                    app.RZXnvertEditField.Value = app.SectorEditor.Sector.
                            RestricZones(k).X{d}(end);
2685                    app.RZYnvertEditField.Value = app.SectorEditor.Sector.
                            RestricZones(k).Y{d}(end);
2686                    updateRZVert(app)
2687                end
2688            end
2689        elseif ~isempty(selNode) && isempty(app.RZTree.Children(k).Children)
2690            if app.VerticeButtonRZ.Value
2691                resetRZXYeditfieldvert(app)
2692                updateRZVert(app)
2693            elseif app.ArcoButtonRZ.Value
```

```
2694                    resetRZeditfieldarch(app)
2695                    updateRZArch(app)
2696                end
2697            elseif isempty(selNode) && ~isempty(app.RZTree.Children)
2698                if app.VerticeButtonRZ.Value
2699                    resetRZXYeditfieldvert(app)
2700                    app.RZXnvertEditField.Value = app.SectorEditor.Sector.
                            RestricZones(k).X{end}(end);
2701                    app.RZYnvertEditField.Value = app.SectorEditor.Sector.
                            RestricZones(k).Y{end}(end);
2702                    updateBorderVert(app)
2703                elseif app.ArcoButtonRZ.Value
2704                    resetRZeditfieldarch(app)
2705                    app.RZXnarchEditField.Value = app.SectorEditor.Sector.
                            RestricZones(k).X{end}(end);
2706                    app.RZYnarchEditField.Value = app.SectorEditor.Sector.
                            RestricZones(k).Y{end}(end);
2707                    updateRZArch(app)
2708                end
2709            end
2710            if ~isempty(selNode) && ~isempty(app.SectorEditor.Sector.
                    RestricZones(k).Z)
2711                if app.SectorEditor.Sector.RestricZones(k).Z(2) >= 1000
2712                    FLcharsup = FLChar(app,app.SectorEditor.Sector.RestricZones(k
                            ).Z(1));
2713                    FLcharinf = FLChar(app,app.SectorEditor.Sector.RestricZones(k
                            ).Z(2));
2714                    app.RZFLsupEditField.Value = FLcharsup;
2715                    app.RZFLinfEditField.Value = FLcharinf;
2716                elseif app.SectorEditor.Sector.RestricZones(k).Z(2) < 1000 &&
                        app.SectorEditor.Sector.RestricZones(k).Z(1) >= 1000
2717                    FLcharsup = FLChar(app,app.SectorEditor.Sector.RestricZones(k
                            ).Z(1));
2718                    ftcharinf = ftChar(app,app.SectorEditor.Sector.RestricZones(k
                            ).Z(2));
2719                    app.RZFLsupEditField.Value = FLcharsup;
2720                    app.RZFLinfEditField.Value = ftcharinf;
2721                elseif app.SectorEditor.Sector.RestricZones(k).Z(1) < 1000
2722                    ftcharsup = ftChar(app,app.SectorEditor.Sector.RestricZones(k
                            ).Z(1));
2723                    ftcharinf = ftChar(app,app.SectorEditor.Sector.RestricZones(k
                            ).Z(2));
2724                    app.RZFLsupEditField.Value = ftcharsup;
2725                    app.RZFLinfEditField.Value = ftcharinf;
2726                end
2727                updateRZAlt(app)
2728            else
2729                clearFLeditfields(app,'RZ')
2730                resetRZeditfieldalt(app)
2731                updateRZAlt(app)
2732            end
2733            if ~isempty(selNode) && ~isempty(app.RZTree.Children(k).Children)
2734                plotEditorgraph(app)
2735                if selNode.Parent.Tag(1) == 'T'
2736                    Xrc = cell(size(app.SectorEditor.Sector.RestricZones(k).X));
2737                    Yrc = cell(size(app.SectorEditor.Sector.RestricZones(k).Y));
2738                    for j = 1:size(app.SectorEditor.Sector.RestricZones(k).X,2)
```

```
                    if app.RZTree.Children(k).Children(j).Text(1) == 'V'
                        Xrc{j} = app.SectorEditor.Sector.RestricZones(k).X{j};
                        Yrc{j} = app.SectorEditor.Sector.RestricZones(k).Y{j};
                    elseif app.RZTree.Children(k).Children(j).Text(1) == 'A'
                        Xa = app.SectorEditor.Sector.RestricZones(k).X{j};
                        Ya = app.SectorEditor.Sector.RestricZones(k).Y{j};
                        A = [Xa(1),Ya(1)];
                        B = [Xa(end),Ya(end)];
                        Arco = [Xa',Ya'];
                        I = encuentraI(app,Arco,A,B);
                        Xrc{j} = [A(1),I(1),B(1)];
                        Yrc{j} = [A(2),I(2),B(2)];
                    end
                end
                Xr = cell2mat(Xrc);
                Yr = cell2mat(Yrc);
            elseif selNode.Parent.Tag(1) == 'N'
                if selNode.Text(1) == 'V'
                    Xr = app.SectorEditor.Sector.RestricZones(k).X{d};
                    Yr = app.SectorEditor.Sector.RestricZones(k).Y{d};
                elseif selNode.Text(1) == 'A'
                    X = app.SectorEditor.Sector.RestricZones(k).X{d};
                    Y = app.SectorEditor.Sector.RestricZones(k).Y{d};
                    A = [X(1),Y(1)];
                    B = [X(end), Y(end)];
                    Arco = [X',Y'];
                    I = encuentraI(app,Arco,A,B);
                    Xr = [A(1), I(1), B(1)];
                    Yr = [A(2), I(2), B(2)];
                end
            end
            plot(app.Editorgraph,Xr,Yr,'or')
            hold(app.Editorgraph,"off")
        end
    end

    % Value changed function: RZXnarchEditField
    function RZXnarchEditFieldValueChanged(app, event)
        app.Arch.orig(1,1) = app.RZXnarchEditField.Value;
    end

    % Value changed function: RZYnarchEditField
    function RZYnarchEditFieldValueChanged(app, event)
        app.Arch.orig(1,2) = app.RZYnarchEditField.Value;
    end

    % Value changed function: RZXn1archEditField
    function RZXn1archEditFieldValueChanged(app, event)
        app.Arch.dest(1,1) = app.RZXn1archEditField.Value;
    end

    % Value changed function: RZYn1archEditField
    function RZYn1archEditFieldValueChanged(app, event)
        app.Arch.dest(1,2) = app.RZYn1archEditField.Value;
    end
```

```
2794
2795        % Value changed function: RZARadiusEditField
2796        function RZARadiusEditFieldValueChanged(app, event)
2797            app.Arch.rad = app.RZARadiusEditField.Value;
2798        end
2799
2800        % Value changed function: RZAConcaveConvexSwitch
2801        function RZAConcaveConvexSwitchValueChanged(app, event)
2802            dir = app.RZAConcaveConvexSwitch.Value;
2803            if dir == '^'
2804                app.Arch.dir = 'Concavo';
2805            elseif dir == 'v'
2806                app.Arch.dir = 'Convexo';
2807            end
2808        end
2809
2810        % Value changed function: RZACortoLargoSwitch
2811        function RZACortoLargoSwitchValueChanged(app, event)
2812            app.Arch.len = app.RZACortoLargoSwitch.Value;
2813        end
2814
2815        % Value changed function: RZZinfEditField
2816        function RZZinfEditFieldValueChanged(app, event)
2817            if app.FLftinfDropDownRZ.Value == '1'
2818                app.Alt.inf = app.RZZinfEditField.Value * 100;
2819            elseif app.FLftinfDropDownRZ.Value == '2'
2820                app.Alt.inf = app.RZZinfEditField.Value;
2821            end
2822        end
2823
2824        % Button pushed function: EliminarRZButton
2825        function EliminarRZButtonPushed(app, event)
2826            selNode = app.RZTree.SelectedNodes;
2827            if ~isempty(selNode)
2828                if selNode.Parent.Tag(1) == 'T'
2829                    k = selNode.NodeData;
2830                    t = size(app.SectorEditor.Sector.RestricZones,2);
2831                    if t == 1
2832                        app.SectorEditor.Sector.RestricZones = [];
2833                    elseif t > 1
2834                        if k < t && k > 1
2835                            i = [1:k-1,k+1:t];
2836                        elseif k == 1
2837                            i = k+1:t;
2838                        elseif k == t
2839                            i = 1:k-1;
2840                        end
2841                        if k < t
2842                            for j = k+1:t
2843                                CharNum = dec2char(app,j-1);
2844                                app.RZTRee.Children(j).Text = char(['Zona
                                        Restringida ',CharNum]);
2845                                app.RZTRee.Children(j).NodeData = j-1;
2846                            end
2847                        end
2848                        RZ = app.SectorEditor.Sector.RestricZones(i);
2849                        app.SectorEditor.Sector.RestricZones = RZ;
```

```
2850                    end
2851                    delete(app.RZTree.Children(k));
2852                elseif selNode.Parent.Tag(1) == 'N'
2853                    k = selNode.Parent.NodeData;
2854                    d = selNode.NodeData + 1;
2855                    t = size(app.RZTree.Children(k).Children,1);
2856                    if t > 1
2857                        for i = 1:t
2858                            if i > d
2859                                app.SectorEditor.Sector.RestricZones(k).X{i-1} =
                                        app.SectorEditor.Sector.RestricZones(k).X{i};
2860                                app.SectorEditor.Sector.RestricZones(k).Y{i-1} =
                                        app.SectorEditor.Sector.RestricZones(k).Y{i};
2861                                T = app.RZTree.Children(k).Children(i).Text;
2862                                CharNum = dec2char(app,i-1);
2863                                if T(1) == 'V'
2864                                    app.RZTree.Children(k).Children(i-1).Text =
                                            char(['Vértice ',CharNum]);
2865                                elseif T(1) == 'A'
2866                                    app.RZTree.Children(k).Children(i-1).Text =
                                            char(['Arco ',CharNum]);
2867                                end
2868                            end
2869                        end
2870                        X = cell([size(app.SectorEditor.Sector.RestricZones(k).X
                                ,1),size(app.SectorEditor.Sector.RestricZones(k).X,2)
                                -1]);
2871                        Y = cell([size(app.SectorEditor.Sector.RestricZones(k).Y
                                ,1),size(app.SectorEditor.Sector.RestricZones(k).Y,2)
                                -1]);
2872                        for j = 1:size(app.SectorEditor.Sector.RestricZones(k).X
                                ,2)-1
2873                            X{j} = app.SectorEditor.Sector.RestricZones(k).X{j};
2874                            Y{j} = app.SectorEditor.Sector.RestricZones(k).Y{j};
2875                        end
2876                        app.SectorEditor.Sector.RestricZones(k).X = X;
2877                        app.SectorEditor.Sector.RestricZones(k).Y = Y;
2878                        delete(app.RZTree.Children(k).Children(end))
2879                    elseif t == 1
2880                        app.SectorEditor.Sector.RestricZones(k).X = {};
2881                        app.SectorEditor.Sector.RestricZones(k).Y = {};
2882                        delete(app.RZTree.Children(k).Children(1))
2883                    end
2884                end
2885                app.RZTree.SelectedNodes = [];
2886                RZTreeSelectionChanged(app)
2887                plotEditorgraph(app)
2888                hold(app.Editorgraph,"off")
2889            end
2890        end
2891
2892        % Button pushed function: MostrarRZAButton
2893        function MostrarRZAButtonPushed(app, event)
2894            plotEditorgraph(app)
2895            if ~isempty(app.Arch)
2896                if app.Arch.orig(1) ~= app.Arch.dest(1) || app.Arch.orig(2) ~=
                        app.Arch.dest(2)
```

```
                    if app.Arch.rad >= 0.5*norm(app.Arch.dest-app.Arch.orig)
                        Centro = Calccentre(app,app.Arch);
                        [X,Y] = Calcarch(app,app.Arch,Centro);
                    elseif app.Arch.rad >= 0
                        X = [app.Arch.orig(1), app.Arch.dest(1)];
                        Y = [app.Arch.orig(2), app.Arch.dest(2)];
                    end
                    plot(app.Editorgraph,X,Y,'-r')
                end
            end
            hold(app.Editorgraph,"off")
        end

        % Button pushed function: GuardarRZAButton
        function GuardarRZAButtonPushed(app, event)
            if ~isempty(app.Arch)
                selNode = app.RZTree.SelectedNodes;
                if ~isempty(selNode)
                    if selNode.Parent.Tag(1) == 'T'
                        k = selNode.NodeData;
                        if isempty(selNode.Children)
                           d = 1;
                        else
                           d = selNode.Children(end).NodeData + 1;
                        end
                    elseif selNode.Parent.Tag(1) == 'N'
                        k = selNode.Parent.NodeData;
                        d = selNode.NodeData + 1;
                    end
                else
                    k = size(app.RZTree.Children,1);
                    d = size(app.RZTree.Children(k).Children,1) + 1;
                end
                if app.Arch.orig(1) ~= app.Arch.dest(1) || app.Arch.orig(2) ~=
                    app.Arch.dest(2)
                    if app.Arch.rad >= 0.5*norm(app.Arch.dest-app.Arch.orig)
                        Centro = Calccentre(app,app.Arch);
                        [X,Y] = Calcarch(app,app.Arch,Centro);
                    elseif app.Arch.rad >= 0
                        X = [app.Arch.orig(1), app.Arch.dest(1)];
                        Y = [app.Arch.orig(2), app.Arch.dest(2)];
                    end
                elseif app.Arch.orig(1) == app.Arch.dest(1) && app.Arch.orig(2)
                    == app.Arch.dest(2)
                    X = [];
                    Y = [];
                end
                if ~isempty(X)
                    if isempty(app.SectorEditor.Sector.RestricZones(k).X)
                        app.SectorEditor.Sector.RestricZones(k).X{1} = X;
                        app.SectorEditor.Sector.RestricZones(k).Y{1} = Y;
                        uitreenode(app.RZTree.Children(k),"NodeData",1,"Text",'
                            Arco 1');
                    elseif ~isempty(selNode)
                        t = size(app.SectorEditor.Sector.RestricZones(k).X,2);
                        for i = t+1:-1:d
                            if i ~= d
```

```
2951                            app.SectorEditor.Sector.RestricZones(k).X{i} = app.
                                    SectorEditor.Sector.RestricZones(k).X{i-1};
2952                            app.SectorEditor.Sector.RestricZones(k).Y{i} = app.
                                    SectorEditor.Sector.RestricZones(k).Y{i-1};
2953                            T = app.RZTree.Children(k).Children(i-1).Text(1);
2954                            CharNum = dec2char(app,i);
2955                            if T == 'V'
2956                                app.RZTree.Children(k).Children(i-1).Text =
                                        char(['Vértice ',CharNum]);
2957                                app.RZTree.Children(k).Children(i-1).NodeData
                                        = i;
2958                            elseif T == 'A'
2959                                app.RZTree.Children(k).Children(i-1).Text =
                                        char(['Arco ',CharNum]);
2960                                app.RZTree.Children(k).Children(i-1).NodeData
                                        = i;
2961                            end
2962                        elseif i == d
2963                            app.SectorEditor.Sector.RestricZones(k).X{i} = X;
2964                            app.SectorEditor.Sector.RestricZones(k).Y{i} = Y;
2965                            CharNum = dec2char(app,d);
2966                            Node = uitreenode(app.RZTree.Children(k),'NodeData
                                    ',d,'Text',char(['Arco ',CharNum]));
2967                            move(Node,app.RZTree.Children(k).Children(d-1));
2968                        end
2969                    end
2970                else
2971                    t = size(app.SectorEditor.Sector.RestricZones(k).X,2);
2972                    app.SectorEditor.Sector.RestricZones(k).X{t+1} = X;
2973                    app.SectorEditor.Sector.RestricZones(k).Y{t+1} = Y;
2974                    CharNum = dec2char(app,t+1);
2975                    uitreenode(app.RZTree.Children(k),"NodeData",t+1,"Text",
                            char(['Arco ',CharNum]));
2976                end
2977                resetRZeditfieldarch(app)
2978                updateRZArch(app)
2979                plotEditorgraph(app)
2980                hold(app.Editorgraph,"off")
2981            end
2982            app.RZTree.SelectedNodes = [];
2983            RZTreeSelectionChanged(app)
2984        end
2985    end
2986
2987    % Value changed function: RZZsupEditField
2988    function RZZsupEditFieldValueChanged(app, event)
2989        if app.FLftsupDropDownRZ.Value == '1'
2990            app.Alt.sup = app.RZZsupEditField.Value * 100;
2991        elseif app.FLftsupDropDownRZ.Value == '2'
2992            app.Alt.sup = app.RZZsupEditField.Value;
2993        end
2994    end
2995
2996    % Button pushed function: GuardarRZAltButton
2997    function GuardarRZAltButtonPushed(app, event)
2998        if app.Alt.inf >= 0 && app.Alt.sup > app.Alt.inf
2999            selNode = app.RZTree.SelectedNodes;
```

```
3000                if ~isempty(selNode)
3001                    if selNode.Parent.Tag(1) == 'T'
3002                        k = selNode.NodeData;
3003                    elseif selNode.Parent.Tag(1) == 'N'
3004                        k = selNode.Parent.NodeData;
3005                    end
3006                else
3007                    k = size(app.RZTree.Children,1);
3008                end
3009                app.SectorEditor.Sector.RestricZones(k).Z(1) = app.Alt.sup;
3010                app.SectorEditor.Sector.RestricZones(k).Z(2) = app.Alt.inf;
3011                app.RZFLsupEditField.Editable = 'on';
3012                app.RZFLinfEditField.Editable = 'on';
3013                if app.Alt.inf >= 1000
3014                    FLcharsup = FLChar(app,app.Alt.sup);
3015                    FLcharinf = FLChar(app,app.Alt.inf);
3016                    app.RZFLsupEditField.Value = FLcharsup;
3017                    app.RZFLinfEditField.Value = FLcharinf;
3018                elseif app.Alt.inf < 1000 && app.Alt.sup >= 1000
3019                    FLcharsup = FLChar(app,app.Alt.sup);
3020                    ftcharinf = ftChar(app,app.Alt.inf);
3021                    app.RZFLsupEditField.Value = FLcharsup;
3022                    app.RZFLinfEditField.Value = ftcharinf;
3023                elseif app.Alt.sup < 1000
3024                    ftcharsup = ftChar(app,app.Alt.sup);
3025                    ftcharinf = ftChar(app,app.Alt.inf);
3026                    app.RZFLsupEditField.Value = ftcharsup;
3027                    app.RZFLinfEditField.Value = ftcharinf;
3028                end
3029                app.RZFLsupEditField.Editable = 'off';
3030                app.RZFLinfEditField.Editable = 'off';
3031                resetRZeditfieldalt(app)
3032                updateRZAlt(app)
3033            end
3034        end
3035
3036        % Value changed function: NombreAirportEditField
3037        function NombreAirportEditFieldValueChanged(app, event)
3038            app.NombreAirportEditField.BackgroundColor = [1 1 1];
3039            Name = app.NombreAirportEditField.Value;
3040            allmayus = 1;
3041            for i = 1:length(Name)
3042                if uint16(Name(i)) >= 65 && uint16(Name(i))<=90
3043                else
3044                    allmayus = 0;
3045                end
3046            end
3047            if length(Name) == 5 && allmayus == 1
3048                app.Airport.Name = Name;
3049            else
3050                app.Airport.Name = 'FAIL';
3051            end
3052        end
3053
3054        % Value changed function: XairportEditField
3055        function XairportEditFieldValueChanged(app, event)
3056            app.Airport.X = app.XairportEditField.Value;
```

```
3057        end
3058
3059        % Value changed function: YairportEditField
3060        function YairportEditFieldValueChanged(app, event)
3061            app.Airport.Y = app.YairportEditField.Value;
3062        end
3063
3064        % Button pushed function: GuardarAirportButton
3065        function GuardarAirportButtonPushed(app, event)
3066            if ~isempty(app.Airport)
3067                selNode = app.AirportTree.SelectedNodes;
3068                if ~isempty(selNode)
3069                    if selNode.Parent.Tag(1) == 'T'
3070                        k = selNode.NodeData+1;
3071                    elseif selNoe.Parent.Tag(1) == 'N'
3072                        k = selNode.Parent.NodeData+1;
3073                    end
3074                elseif ~isempty(app.AirportTree.Children)
3075                    k = app.AirportTree.Children(end).NodeData+1;
3076                else
3077                    k = 1;
3078                end
3079                if length(app.Airport.Name) == 4
3080                    app.NombreAirportEditField.Value = 'ERROR';
3081                    app.NombreAirportEditField.BackgroundColor = [1 0 0];
3082                elseif length(app.Airport.Name) == 5
3083                    app.SectorEditor.Sector.Airport(k).Name = app.Airport.Name;
3084                    app.SectorEditor.Sector.Airport(k).X = app.Airport.X;
3085                    app.SectorEditor.Sector.Airport(k).Y = app.Airport.Y;
3086                    if isempty(selNode)
3087                        app.SectorEditor.Sector.Airport(k).Runway = [];
3088                        Xchar = dec2char(app,app.Airport.X);
3089                        Ychar = dec2char(app,app.Airport.Y);
3090                        Node = uitreenode(app.AirportTree,'NodeData',k,'Text',app.
                                Airport.Name,'Tag','Node');
3091                        uitreenode(Node,'Text',char(['[',Xchar,',',Ychar,']']))
3092                    else
3093                        app.AirportTree.Children(k).Text = app.Airport.Name;
3094                        Xchar = dec2char(app,app.Airport.X);
3095                        Ychar = dec2char(app,app.Airport.Y);
3096                        app.AirportTree.Children(k).Children(1).Text = char(['[',
                                Xchar,',',Ychar,']']);
3097                    end
3098                end
3099                plotEditorgraph(app)
3100                app.AirportTree.SelectedNodes = [];
3101                AirportTreeSelectionChanged(app)
3102            end
3103        end
3104
3105        % Selection changed function: AirportTree
3106        function AirportTreeSelectionChanged(app, event)
3107            selNode = app.AirportTree.SelectedNodes;
3108            if ~isempty(selNode)
3109                if selNode.Parent.Tag(1) == 'T'
3110                    k = selNode.NodeData;
3111                elseif selNode.Parent.Tag(1) == 'N'
```

```
3112                    k = selNode.Parent.NodeData;
3113                end
3114                app.NombreAirportEditField.Value = app.SectorEditor.Sector.
                        Airport(k).Name;
3115                app.XairportEditField.Value = app.SectorEditor.Sector.Airport(k).
                        X;
3116                app.YairportEditField.Value = app.SectorEditor.Sector.Airport(k).
                        X;
3117                if ~isempty(app.SectorEditor.Sector.Airport(k).Runway)
3118                    app.Airport.Runway = app.SectorEditor.Sector.Airport(k).
                            Runway;
3119                end
3120                updateAirport(app)
3121                plotEditorgraph(app)
3122                hold(app.Editorgraph,'on')
3123                plot(app.Editorgraph,app.Airport.X,app.Airport.Y,'sc')
3124                text(app.Editorgraph,app.Airport.X,app.Airport.Y,app.Airport.
                        Name,'Color',[0 1 1],'HorizontalAlignment','center','
                        VerticalAlignment','bottom','EdgeColor',[0 1 1]);
3125                hold(app.Editorgraph,'off')
3126            else
3127                resetAirporteditfield(app)
3128                updateAirport(app)
3129            end
3130            app.RWYPanel.Visible = 'off';
3131            resetRWYPanel(app)
3132        end
3133
3134        % Button pushed function: EliminarAirportButton
3135        function EliminarAirportButtonPushed(app, event)
3136            selNode = app.AirportTree.SelectedNodes;
3137            if ~isempty(selNode)
3138                if selNode.Parent.Tag(1) == 'T'
3139                    k = selNode.NodeData;
3140                elseif selNode.Parent.Tag(1) == 'N'
3141                    k = selNode.Parent.NodeData;
3142                end
3143                app.SectorEditor.Sector.Airport(k) = [];
3144                for i = 1:length(app.AirportTree.Children)
3145                    if i > k
3146                        app.AirportTree.Children(i).NodeData = app.AirportTree.
                                Children(i-1).NodeData;
3147                    end
3148                end
3149                delete(app.AirportTree.Children(k))
3150                app.AirportTree.SelectedNodes = [];
3151                AirportTreeSelectionChanged(app)
3152                plotEditorgraph(app)
3153                hold(app.Editorgraph,'off')
3154            end
3155        end
3156
3157        % Button pushed function: AddRWYButton
3158        function AddRWYButtonPushed(app, event)
3159            resetRWYPanel(app)
3160            app.RWYPanel.Visible = 'on';
3161        end
```

```
3162
3163        % Value changed function: RWYNewEditSwitch
3164        function RWYNewEditSwitchValueChanged(app, event)
3165            value = app.RWYNewEditSwitch.Value;
3166            if value(1) == 'N'
3167                app.NewRWYPanel.Visible = 'on';
3168                app.EditRWYPanel.Visible = 'off';
3169            elseif value(1) == 'E'
3170                app.NewRWYPanel.Visible = 'off';
3171                app.EditRWYPanel.Visible = 'on';
3172            end
3173        end
3174
3175        % Value changed function: NewRWYSpinner
3176        function NewRWYSpinnerValueChanged(app, event)
3177            app.Airport.Runway = app.NewRWYSpinner.Value;
3178        end
3179
3180        % Value changed function: RWYEditSpinner
3181        function RWYEditSpinnerValueChanged(app, event)
3182            app.Airport.Runway = app.RWYEditSpinner.Value;
3183        end
3184
3185        % Value changed function: RWYEditDropDown
3186        function RWYEditDropDownValueChanged(app, event)
3187            RWYpair = app.RWYEditDropDown.Value;
3188            clearRWYspinners(app)
3189            selNode = app.AirportTree.SelectedNodes;
3190            if ~isempty(selNode)
3191                if selNode.Parent.Tag(1) == 'T'
3192                    k = selNode.NodeData;
3193                elseif selNode.Parent.Tag(1) == 'N'
3194                    k = selNode.Parent.NodeData;
3195                end
3196                if RWYpair > 0
3197                    RWYE = app.SectorEditor.Sector.Airport(k).Runway(2*(RWYpair
                            -1) + 1);
3198                    app.RWYEditSpinner.Value = RWYE;
3199                    RWYEditSpinnerValueChanged(app)
3200                end
3201            end
3202        end
3203
3204        % Button pushed function: GuardarNewRWYButton
3205        function GuardarNewRWYButtonPushed(app, event)
3206            selNode = app.AirportTree.SelectedNodes;
3207            if selNode.Parent.Tag(1) == 'T'
3208                k = selNode.NodeData;
3209            elseif selNode.Parent.Tag(1) == 'N'
3210                k = selNode.Parent.NodeData;
3211            end
3212            RWY = app.SectorEditor.Sector.Airport(k).Runway;
3213            if ~isempty(RWY)
3214                t = size(RWY,2);
3215            else
3216                t = 0;
3217            end
```

```
3218            if app.Airport.Runway > 0
3219                if app.Airport.Runway <= 18
3220                    RWYE = app.Airport.Runway;
3221                    RWYW = RWYE + 18;
3222                else
3223                    RWYW = app.Airport.Runway;
3224                    RWYE = RWYW - 18;
3225                end
3226                app.SectorEditor.Sector.Airport(k).Runway(t+1:t+2) = [RWYE,RWYW];

3227            end
3228            resetRWYPanel(app)
3229        end
3230
3231        % Button pushed function: GuardarRWYEditButton
3232        function GuardarRWYEditButtonPushed(app, event)
3233            selNode = app.AirportTree.SelectedNodes;
3234            if selNode.Parent.Tag(1) == 'T'
3235                k = selNode.NodeData;
3236            elseif selNode.Parent.Tag(1) == 'N'
3237                k = selNode.Parent.NodeData;
3238            end
3239            RWYpair = app.RWYEditDropDown.Value;
3240            t = 2*(RWYpair-1);
3241            if app.Airport.Runway > 0 && t >= 0
3242                if app.Airport.Runway <= 18
3243                    RWYE = app.Airport.Runway;
3244                    RWYW = RWYE + 18;
3245                else
3246                    RWYW = app.Airport.Runway;
3247                    RWYE = RWYW - 18;
3248                end
3249                app.SectorEditor.Sector.Airport(k).Runway(t+1:t+2) = [RWYE,RWYW];

3250            end
3251            resetRWYPanel(app)
3252        end
3253
3254        % Button pushed function: EliminarRWYEditButton
3255        function EliminarRWYEditButtonPushed(app, event)
3256            selNode = app.AirportTree.SelectedNodes;
3257            if selNode.Parent.Tag(1) == 'T'
3258                k = selNode.NodeData;
3259            elseif selNode.Parent.Tag(1) == 'N'
3260                k = selNode.Parent.NodeData;
3261            end
3262            RWYpair = app.RWYEditDropDown.Value;
3263            t = 2*(RWYpair-1);
3264            if t >= 0
3265                app.SectorEditor.Sector.Airport(k).Runway(t+1:t+2) = [];
3266            end
3267            resetRWYPanel(app)
3268        end
3269
3270        % Value changed function: XwaypointEditField
3271        function XwaypointEditFieldValueChanged(app, event)
3272            app.Waypoint.X = app.XwaypointEditField.Value;
```

```
3273        end
3274
3275        % Value changed function: YwaypointEditField
3276        function YwaypointEditFieldValueChanged(app, event)
3277            app.Waypoint.Y = app.YwaypointEditField.Value;
3278        end
3279
3280        % Value changed function: NombreWPEditField
3281        function NombreWPEditFieldValueChanged(app, event)
3282            app.NombreWPEditField.BackgroundColor = [1 1 1];
3283            Name = app.NombreWPEditField.Value;
3284            allmayus = 1;
3285            for i = 1:length(Name)
3286                if uint16(Name(i)) >= 65 && uint16(Name(i))<=90
3287                else
3288                    allmayus = 0;
3289                end
3290            end
3291            if length(Name) == 5 && allmayus == 1
3292                app.Waypoint.Name = Name;
3293            else
3294                app.Waypoint.Name = 'FAIL';
3295            end
3296        end
3297
3298        % Button pushed function: GuardarWaypointButton
3299        function GuardarWaypointButtonPushed(app, event)
3300            if ~isempty(app.Waypoint)
3301                selNode = app.WaypointTree.SelectedNodes;
3302                if ~isempty(selNode)
3303                    if selNode.Parent.Tag(1) == 'T'
3304                        k = selNode.NodeData;
3305                    elseif selNode.Parent.Tag(1) == 'N'
3306                        k = selNode.Parent.NodeData;
3307                    end
3308                elseif ~isempty(app.WaypointTree.Children)
3309                    k = app.WaypointTree.Children(end).NodeData+1;
3310                else
3311                    k = 1;
3312                end
3313                if length(app.Waypoint.Name) == 4
3314                    app.NombreWPEditField.Value = 'ERROR';
3315                    app.NombreWPEditField.BackgroundColor = [1 0 0];
3316                elseif length(app.Waypoint.Name) == 5
3317                    app.SectorEditor.Sector.Waypoint(k).Name = app.Waypoint.Name;
3318                    app.SectorEditor.Sector.Waypoint(k).X = app.Waypoint.X;
3319                    app.SectorEditor.Sector.Waypoint(k).Y = app.Waypoint.Y;
3320                    app.SectorEditor.Sector.Waypoint(k).Freq = app.Waypoint.Freq;
3321                    if isempty(selNode)
3322                        Xchar = dec2char(app,app.Waypoint.X);
3323                        Ychar = dec2char(app,app.Waypoint.Y);
3324                        Node = uitreenode(app.WaypointTree,'NodeData',k,'Text',
                                app.Waypoint.Name,'Tag','Node');
3325                        uitreenode(Node,'Text',char(['[',Xchar,',',Ychar,']']))
3326                    else
3327                        app.WaypointTree.Children(k).Text = app.Waypoint.Name;
3328                        Xchar = dec2char(app,app.Waypoint.X);
```

```
3329                        Ychar = dec2char(app,app.Waypoint.Y);
3330                        app.WaypointTree.Children(k).Children(1).Text = char(['[',
                                Xchar,',',Ychar,']']);
3331                    end
3332                end
3333                plotEditorgraph(app)
3334                app.WaypointTree.SelectedNodes = [];
3335                WaypointTreeSelectionChanged(app)
3336            end
3337        end
3338
3339        % Selection changed function: WaypointTree
3340        function WaypointTreeSelectionChanged(app, event)
3341            selNode = app.WaypointTree.SelectedNodes;
3342            if ~isempty(selNode)
3343                if selNode.Parent.Tag(1) == 'T'
3344                    k = selNode.NodeData;
3345                elseif selNode.Parent.Tag(1) == 'N'
3346                    k = selNode.Parent.NodeData;
3347                end
3348                app.NombreWPEditField.Value = app.SectorEditor.Sector.Waypoint(k
                        ).Name;
3349                app.XwaypointEditField.Value = app.SectorEditor.Sector.Waypoint(
                        k).X;
3350                app.YwaypointEditField.Value = app.SectorEditor.Sector.Waypoint(
                        k).Y;
3351                updateWaypoint(app)
3352                if ~isempty(app.SectorEditor.Sector.Waypoint(k).Freq)
3353                    app.Waypoint.Freq = app.SectorEditor.Sector.Waypoint(k).Freq;
3354                    app.FreqMHzEditField.Value = app.Waypoint.Freq;
3355                    app.WaypointFronterizoFreqPanel.Visible = 'on';
3356                else
3357                    app.WaypointFronterizoFreqPanel.Visible = 'off';
3358                end
3359                plotEditorgraph(app)
3360                hold(app.Editorgraph,'on')
3361                plot(app.Editorgraph,app.Waypoint.X,app.Waypoint.Y,'sc')
3362                text(app.Editorgraph,app.Waypoint.X,app.Waypoint.Y,app.Waypoint.
                        Name,'Color',[0 1 1],'HorizontalAlignment','center','
                        VerticalAlignment','bottom','EdgeColor',[0 1 1]);
3363                hold(app.Editorgraph,'off')
3364            else
3365                resetWaypointeditfield(app)
3366                updateWaypoint(app)
3367                app.WaypointFronterizoFreqPanel.Visible = 'off';
3368            end
3369        end
3370
3371        % Button pushed function: EliminarWaypointButton
3372        function EliminarWaypointButtonPushed(app, event)
3373            selNode = app.WaypointTree.SelectedNodes;
3374            if ~isempty(selNode)
3375                if selNode.Parent.Tag(1) == 'T'
3376                    k = selNode.NodeData;
3377                elseif selNode.Parent.Tag(1) == 'N'
3378                    k = selNode.Parent.NodeData;
3379                end
```

```
3380                app.SectorEditor.Sector.Waypoint(k) = [];
3381                for i = 1:length(app.WaypointTree.Children)
3382                    if i > k
3383                        app.WaypointTree.Children(i).NodeData = app.WaypointTree.
                               Children(i-1).NodeData;
3384                    end
3385                end
3386                delete(app.WaypointTree.Children(k))
3387                app.WaypointTree.SelectedNodes = [];
3388                WaypointTreeSelectionChanged(app)
3389                plotEditorgraph(app)
3390                hold(app.Editorgraph,'off')
3391            end
3392        end
3393
3394        % Button pushed function: WaypointFronterizoButton
3395        function WaypointFronterizoButtonPushed(app, event)
3396            if ~isempty(app.Waypoint)
3397                WP = [app.Waypoint.X, app.Waypoint.Y];
3398                Bcell = [app.SectorEditor.Sector.Border.X; app.SectorEditor.
                           Sector.Border.Y];
3399                j = 1;
3400                dminmatrix = zeros(size(Bcell,2),3);
3401                while j <= size(Bcell,2)
3402                    if size(Bcell{1,j},2)> 1
3403                        Bmat = [Bcell{1,j};Bcell{2,j}];
3404                    elseif j < size(Bcell,2) && size(Bcell{1,j},2) == 1
3405                        v = [Bcell{1,j+1}(1);Bcell{2,j+1}(1)] - [Bcell{1,j};Bcell
                               {2,j}];
3406                        Bmat = zeros(2,1001);
3407                        for h = 0:0.001:1
3408                            Bmat(:,round(1000*h+1)) = [Bcell{1,j};Bcell{2,j}] + h*
                                   v;
3409                        end
3410                    end
3411                    D = zeros(size(Bmat,2),1);
3412                    for i = 1:length(D)
3413                        D(i) = norm(Bmat(:,i)'-WP);
3414                    end
3415                    [dmin,imin] = min(D);
3416                    dminmatrix(j,:) = [dmin,Bmat(:,imin)'];
3417                    j = j+1;
3418                end
3419                [~,jmin] = min(dminmatrix(:,1));
3420                WP = dminmatrix(jmin,2:3);
3421                app.XwaypointEditField.Value = WP(1);
3422                app.YwaypointEditField.Value = WP(2);
3423                updateWaypoint(app)
3424                resetWPfreqPanel(app)
3425                app.WaypointFronterizoFreqPanel.Visible = 'on';
3426            end
3427        end
3428
3429        % Value changed function: FreqMHzEditField
3430        function FreqMHzEditFieldValueChanged(app, event)
3431            app.Waypoint.Freq = app.FreqMHzEditField.Value;
3432        end
```

```
3433
3434        % Button pushed function: GuardarWPFreqButton
3435        function GuardarWPFreqButtonPushed(app, event)
3436            selNode = app.WaypointTree.SelectedNodes;
3437            if ~isempty(selNode)
3438                if selNode.Parent.Tag(1) == 'T'
3439                    k = selNode.NodeData;
3440                elseif selNode.Parent.Tag(1) == 'N'
3441                    k = selNode.Parent.NodeData;
3442                end
3443                if ~isempty(app.Waypoint.Freq)
3444                    if app.Waypoint.Freq > 0
3445                        app.SectorEditor.Sector.Waypoint(k).Freq = app.Waypoint.
                                Freq;
3446                    end
3447                end
3448            end
3449            app.Waypoint.Freq = [];
3450            resetWPfreqPanel(app)
3451            resetWaypointeditfield(app)
3452            updateWaypoint(app)
3453            app.WaypointFronterizoFreqPanel.Visible = 'off';
3454            app.WaypointTree.SelectedNodes = [];
3455            WaypointTreeSelectionChanged(app)
3456        end
3457
3458        % Value changed function: NiveldeVueloEditField
3459        function NiveldeVueloEditFieldValueChanged(app, event)
3460            app.Airplane.FL = app.NiveldeVueloEditField.Value;
3461        end
3462
3463        % Value changed function: MatrculadelaaeronaveEditField
3464        function MatrculadelaaeronaveEditFieldValueChanged(app, event)
3465            value = app.MatrculadelaaeronaveEditField.Value;
3466            if length(value) == 6
3467                h = isvalidcallsign(app,value);
3468                if h
3469                    app.Airplane.Callsign = value;
3470                else
3471                    app.Airplane.Callsign = '';
3472                end
3473            end
3474        end
3475
3476        % Value changed function: VelocidaddeVueloktEditField
3477        function VelocidaddeVueloktEditFieldValueChanged(app, event)
3478            app.Airplane.FSPD = app.VelocidaddeVueloktEditField.Value;
3479        end
3480
3481        % Value changed function: ETOEditField
3482        function ETOEditFieldValueChanged(app, event)
3483            value = app.ETOEditField.Value;
3484            eto = [0 0 0 0];
3485            if length(value) == 5
3486                h = isvalidETO(app,value);
3487                if h
3488                    j = 1;
```

```
3489                    for i = [1:2,4:5]
3490                        eto(j) = char2dec(app,value(i));
3491                        j = j + 1;
3492                    end
3493                    H = 10*eto(1) + eto(2);
3494                    Min = 10*eto(3) + eto(4);
3495                    Sec = 0;
3496                    app.Airplane.ETO = duration(H,Min,Sec);
3497                else
3498                    app.ETOEditField.Value = '00:00';
3499                    app.Airplane.ETO = [];
3500                end
3501            else
3502                app.ETOEditField.Value = '00:00';
3503                app.Airplane.ETO = [];
3504            end
3505        end
3506
3507        % Value changed function: EHDGEditField
3508        function EHDGEditFieldValueChanged(app, event)
3509            app.Airplane.HDG = app.EHDGEditField.Value;
3510        end
3511
3512        % Value changed function: RouteWPEditField
3513        function RouteWPEditFieldValueChanged(app, event)
3514            app.RouteWP = app.RouteWPEditField.Value;
3515        end
3516
3517        % Button pushed function: GuardarRouteWPButton
3518        function GuardarRouteWPButtonPushed(app, event)
3519            selNode = app.RouteTree.SelectedNodes;
3520            if isempty(selNode)
3521                if isempty(app.RouteTree.Children)
3522                    k = 1;
3523                else
3524                    k = length(app.RouteTree.Children) + 1;
3525                end
3526            else
3527                k = selNode.NodeData + 1;
3528            end
3529            h = iswp(app,app.RouteWP);
3530            if h
3531                if k <= length(app.Airplane.Route)
3532                    for j = length(app.Airplane.Route):-1:k
3533                        app.Airplane.Route{j+1} = app.Airplane.Route{j};
3534                        app.RouteTree.Children(j).NodeData = j+1;
3535                    end
3536                end
3537                app.Airplane.Route{k} = app.RouteWP;
3538                Node = uitreenode(app.RouteTree,'NodeData',k,'Text',app.RouteWP);

3539                move(Node,app.RouteTree.Children(k-1));
3540            end
3541            app.RouteTree.SelectedNodes = [];
3542            RouteTreeSelectionChanged(app);
3543        end
3544
```

```
3545        % Button pushed function: EliminarRouteWPButton
3546        function EliminarRouteWPButtonPushed(app, event)
3547            selNode = app.RouteTree.SelectedNodes;
3548            if ~isempty(selNode)
3549                k = selNode.NodeData;
3550                delete(app.RouteTree.Children(k));
3551                R = cell([1,length(app.Airplane.Route)-1]);
3552                for i = 1:length(app.Airplane.Route)
3553                    if i < k
3554                        R{i} = app.Airplane.Route{1};
3555                    elseif i >= k && i < length(app.Airplane.Route)
3556                        R{i} = app.Airplane.Route{i+1};
3557                        app.RouteTree.Children(i).NodeData = i;
3558                    end
3559                end
3560                app.Airplane.Route = R;
3561            end
3562            app.RouteTree.SelectedNodes = [];
3563            RouteTreeSelectionChanged(app);
3564        end
3565
3566        % Selection changed function: RouteTree
3567        function RouteTreeSelectionChanged(app, event)
3568            selNode = app.RouteTree.SelectedNodes;
3569            app.RouteWP = '';
3570            app.RouteWPEditField.Value = '';
3571            plotEditorgraph(app);
3572            hold(app.Editorgraph,'on');
3573            if ~isempty(selNode)
3574                k = selNode.NodeData;
3575                wp = app.Airplane.Route{k};
3576                pos = findwp(app,wp);
3577                text(app.Editorgraph,pos(1),pos(2),wp,'HorizontalAlignment',"
                        right",'VerticalAlignment',"top",'EdgeColor','m');
3578            end
3579            hold(app.Editorgraph,'off');
3580        end
3581
3582        % Value changed function: TipodeIncidenciaDropDown
3583        function TipodeIncidenciaDropDownValueChanged(app, event)
3584            value = app.TipodeIncidenciaDropDown.Value;
3585            clearIncident(app);
3586            if value == '0'
3587                app.CambioFLPanel.Visible = 'off';
3588                app.SolicitarDirectoPanel.Visible = 'off';
3589                app.CRNPPanel.Visible = 'off';
3590                app.Incident.Type = value;
3591            elseif value == '1'
3592                app.CambioFLPanel.Visible = 'on';
3593                app.SolicitarDirectoPanel.Visible = 'off';
3594                app.CRNPPanel.Visible = 'off';
3595                app.Incident.Type = value;
3596            elseif value == '2'
3597                app.CambioFLPanel.Visible = 'off';
3598                app.SolicitarDirectoPanel.Visible = 'on';
3599                app.CRNPPanel.Visible = 'off';
3600                app.Incident.Type = value;
```

```
            elseif value == '3'
                app.CambioFLPanel.Visible = 'off';
                app.SolicitarDirectoPanel.Visible = 'off';
                app.CRNPPanel.Visible = 'on';
                app.Incident.Type = value;
            end
        end

        % Value changed function: WaypointTriggerCFLEditField
        function WaypointTriggerCFLEditFieldValueChanged(app, event)
            app.Incident.Trigger = app.WaypointTriggerCFLEditField.Value;
        end

        % Value changed function: FLObjetivoEditField
        function FLObjetivoEditFieldValueChanged(app, event)
            app.Incident.Target = app.FLObjetivoEditField.Value;
        end

        % Value changed function: CFLDropDown
        function CFLDropDownValueChanged(app, event)
            app.Incident.Time = app.CFLDropDown.Value;
        end

        % Value changed function: WaypointTriggerSDEditField
        function WaypointTriggerSDEditFieldValueChanged(app, event)
            app.Incident.Trigger = app.WaypointTriggerSDEditField.Value;
        end

        % Value changed function: WPDirectObjetivoEditField
        function WPDirectObjetivoEditFieldValueChanged(app, event)
            app.Incident.Target = app.WPDirectObjetivoEditField.Value;
        end

        % Value changed function: SDDropDown
        function SDDropDownValueChanged(app, event)
            app.Incident.Time = app.SDDropDown.Value;
        end

        % Value changed function: WaypointTriggerCRNPEditField
        function WaypointTriggerCRNPEditFieldValueChanged(app, event)
            app.Incident.Trigger = app.WaypointTriggerCRNPEditField.Value;
        end

        % Value changed function: RumboNuevoEditField
        function RumboNuevoEditFieldValueChanged(app, event)
            app.Incident.Target = app.RumboNuevoEditField.Value;
        end

        % Value changed function: CRNPDropDown
        function CRNPDropDownValueChanged(app, event)
            app.Incident.Time = app.CRNPDropDown.Value;
        end

        % Button pushed function: GuardarIncidenciaButton
        function GuardarIncidenciaButtonPushed(app, event)
            if app.Incident.Type == '0'
                clearIncident(app);
```

```
3658                app.Airplane.Incident = app.Incident;
3659            elseif app.Incident.Type == '2'
3660                wp1 = app.Incident.Trigger;
3661                wp2 = app.Incident.Target;
3662                h1 = iswp(app,wp1);
3663                h2 = iswp(app,wp2);
3664                if h1 && h2
3665                    app.Airplane.Incident = app.Incident;
3666                end
3667            else
3668                wp = app.Incident.Trigger;
3669                h = iswp(app,wp);
3670                if h
3671                    app.Airplane.Incident = app.Incident;
3672                end
3673            end
3674        end
3675
3676        % Button pushed function: EliminarIncidenciaButton
3677        function EliminarIncidenciaButtonPushed(app, event)
3678            app.TipodeIncidenciaDropDown.Value = 0;
3679            TipodeIncidenciaDropDownValueChanged(app);
3680            app.AirplaneIncident = app.Incident;
3681        end
3682
3683        % Button pushed function: GuardarAirplaneButton
3684        function GuardarAirplaneButtonPushed(app, event)
3685            callsign = app.Airplane.Callsign;
3686            eto = app.Airplane.ETO;
3687            if ~isempty(callsign) && ~isempty(eto)
3688                selNode = app.AirplaneTree.SelectedNodes;
3689                if ~isempty(selNode)
3690                    k = selNode.NodeData;
3691                elseif ~isempty(app.AirplaneTree.Children)
3692                    k = app.AirplaneTree.Children(end).NodeData + 1;
3693                else
3694                    k = 1;
3695                end
3696                app.SectorEditor.Sector.Airplane(k) = app.Airplane;
3697                if ~isempty(selNode)
3698                    app.AirplaneTree.Children(k).Text = callsign;
3699                else
3700                    uitreenode(app.AirplaneTree,'NodeData',k,'Text',callsign);
3701                end
3702                app.AirplaneTree.SelectedNodes = [];
3703                app.AirplaneTreeSelectionChanged(app);
3704            end
3705        end
3706
3707        % Selection changed function: AirplaneTree
3708        function AirplaneTreeSelectionChanged(app, event)
3709            selNode = app.AirplaneTree.SelectedNodes;
3710            plotEditorgraph(app);
3711            hold(app.Editorgraph,'on');
3712            if ~isempty(selNode)
3713                k = selNode.NodeData;
3714                app.Airplane = app.SectorEditor.Sector.Airplane(k);
```

```
3715                updateAirplaneTab(app);
3716                plotRoute(app,app.Airplane.Route,app.Airplane.HDG);
3717            else
3718                clearAirplaneTab(app);
3719            end
3720            hold(app.Editorgraph,'off');
3721        end
3722
3723        % Button pushed function: EliminarAirplaneButton
3724        function EliminarAirplaneButtonPushed(app, event)
3725            selNode = app.AirplaneTree.SelectedNodes;
3726            if ~isempty(selNode)
3727                k = selNode.NodeData;
3728                app.SectorEditor.Sector.Airplane(k) = [];
3729                delete(app.AirplaneTree.Children(k));
3730                for i = k:length(app.AirplaneTree.Children)
3731                    app.AirplaneTree.Children(i).NodeData = i;
3732                end
3733            end
3734            plotEditorgraph(app);
3735            hold(app.Editorgraph,'off');
3736        end
3737
3738        % Button pushed function: GuardarSectorButton
3739        function GuardarSectorButtonPushed(app, event)
3740            SecName = app.SectorEditor.Sector.Name{1};
3741            archivo = char([app.libapp,SecName,app.pmat]);
3742            Sector = app.SectorEditor.Sector;
3743            save(archivo,'Sector');
3744        end
3745
3746        % Button pushed function: GuardarHorarioButton
3747        function GuardarHorarioButtonPushed(app, event)
3748            if ~isempty(app.TimeSector.init) && ~isempty(app.TimeSector.fin)
3749                app.SectorEditor.Sector.Time = app.TimeSector;
3750            else
3751                app.SectorEditor.Sector.Time = [];
3752            end
3753        end
3754
3755        % Value changed function: HorainicioEditField
3756        function HorainicioEditFieldValueChanged(app, event)
3757            value = app.HorainicioEditField.Value;
3758            eto = [0 0 0 0];
3759            if length(value) == 5
3760                h = isvalidETO(app,value);
3761                if h
3762                    j = 1;
3763                    for i = [1:2,4:5]
3764                        eto(j) = char2dec(app,value(i));
3765                        j = j + 1;
3766                    end
3767                    H = 10*eto(1) + eto(2);
3768                    Min = 10*eto(3) + eto(4);
3769                    Sec = 0;
3770                    app.TimeSector.init = duration(H,Min,Sec);
3771                else
```

```
3772                    app.HorainicioEditField.Value = '00:00';
3773                    app.TimeSector.init = [];
3774                end
3775            else
3776                app.HorainicioEditField.Value = '00:00';
3777                app.TimeSector.init = [];
3778            end
3779        end
3780
3781        % Value changed function: HorafinalEditField
3782        function HorafinalEditFieldValueChanged(app, event)
3783            value = app.HorafinalEditField.Value;
3784            eto = [0 0 0 0];
3785            if length(value) == 5
3786                h = isvalidETO(app,value);
3787                if h
3788                    j = 1;
3789                    for i = [1:2,4:5]
3790                        eto(j) = char2dec(app,value(i));
3791                        j = j + 1;
3792                    end
3793                    H = 10*eto(1) + eto(2);
3794                    Min = 10*eto(3) + eto(4);
3795                    Sec = 0;
3796                    app.TimeSector.fin = duration(H,Min,Sec);
3797                else
3798                    app.HorafinalEditField.Value = '00:00';
3799                    app.TimeSector.fin = [];
3800                end
3801            else
3802                app.HorafinalEditField.Value = '00:00';
3803                app.TimeSector.fin = [];
3804            end
3805        end
3806
3807        % Button pushed function: CargarEscenarioSimButton
3808        function CargarEscenarioSimButtonPushed(app, event)
3809            selectednode = app.TreeSimulador.SelectedNodes;
3810            if ~isempty(selectednode)
3811                sect = uint16(char(selectednode.Text));
3812                archivo = char([app.libapp,sect,app.pmat]);
3813                app.SectorSimulador = load(archivo);
3814                LoadScenarioSim(app);
3815                app.Simuladorseleccion.Visible = 'off';
3816                app.Simulador.Visible = 'on';
3817                app.TreeSimulador.SelectedNodes = [];
3818            end
3819        end
3820
3821        % Value changed function: TPWPEF
3822        function TPWPEFValueChanged(app, event)
3823            value = event.Value;
3824            j = event.Source.UserData;
3825            if length(value) == 6 && value(1) == ' '
3826                value = value(2:6);
3827                app.TPWPEF(j).Value = value;
3828            end
```

```
3829            if length(value) == 5
3830                h = isvalidETO(app,value);
3831            else
3832                h = false;
3833            end
3834            for i = 1:length(app.FLSRButton)
3835                if app.FLSRButton(i).Value
3836                    k = app.FLSRButton(i).UserData;
3837                    break
3838                end
3839            end
3840            if h || length(value) == 1
3841                app.FlightStrip(k).TPWP{j} = {value};
3842            end
3843        end
3844
3845        % Value changed function: FLSRButton
3846        function FLSRButtonValueChanged(app, event)
3847            value = event.Value;
3848            k = event.Source.UserData;
3849            clearFlightStrip(app);
3850            app.FSCallsignEditField.Value = ' ';
3851            app.FSFLEditField.Value = ' ';
3852            app.FSSPDEditField.Value = ' ';
3853            app.FSETOEditField.Value = ' ';
3854            for i = 1:length(app.FLSRButton)
3855                if app.FLSRButton(i).UserData ~= k
3856                    app.FLSRButton(i).Value = 0;
3857                end
3858            end
3859            if value
3860                if ~isempty(app.FlightStrip(k))
3861                    app.FSCallsignEditField.Value = app.FlightStrip(k).Callsign;
3862                    FLchar = dec2char(app,app.FlightStrip(k).FL);
3863                    SPDchar = dec2char(app,app.FlightStrip(k).FSPD);
3864                    app.FSFLEditField.Value = char(['FL',FLchar]);
3865                    app.FSSPDEditField.Value = char([SPDchar,'kt']);
3866                    app.FSETOEditField.Value = char(app.FlightStrip(k).ETO,'hh:mm
                            ');
3867                    if  ~isempty(app.FlightStrip(k).Route)
3868                        app.WPRNEF(1).Value = app.FlightStrip(k).Route{1};
3869                        app.TPWPEF(1).Value = app.FlightStrip(k).TPWP{1}{1};
3870                        for j = 2:length(app.FlightStrip(k).Route)
3871                            x = 10 + (j-1)*60;
3872                            app.WPRNEF(j) = uieditfield(app.FlightRoutePanel,'
                                    text','Value',app.FlightStrip(k).Route{j},'
                                    BackgroundColor',[1.00,0.94,0.78],'Position',[x
                                    ,47,55,22],'Editable','off');
3873                            app.TPWPEF(j) = uieditfield(app.FlightRoutePanel,'
                                    text','Value',app.FlightStrip(k).TPWP{j}{1},'
                                    BackgroundColor',[1.00,0.94,0.78],'Position',[x
                                    ,20,55,22],'UserData',j);
3874                            app.TPWPEF(j).ValueChangedFcn = createCallbackFcn(app,
                                     @TPWPEFValueChanged, true);
3875                        end
3876                    end
3877                end
```

```
3878            end
3879        end
3880
3881        % Button pushed function: SalirSimButton
3882        function SalirSimButtonPushed(app, event)
3883            app.Simulador.Visible = 'off';
3884            clearSectorSim(app);
3885            app.SectorSimulador.Sector.Border.X = {0};
3886            app.SectorSimulador.Sector.Border.Y = {0};
3887            plotSimuladorgraph(app);
3888            hold(app.Simuladorgraph,'off');
3889            clearSectorSim(app);
3890            app.DropDown.Items = {'--/--'};
3891            app.DropDown.ItemsData = {'0'};
3892            app.InputmsgTextArea.Value = {' '};
3893            app.CommsTextArea.Value = {' '};
3894            clearlamps(app);
3895            clearTimers(app);
3896            clearFlightStrip(app);
3897            app.FSCallsignEditField.Value = ' ';
3898            app.FSFLEditField.Value = ' ';
3899            app.FSSPDEditField.Value = ' ';
3900            app.FSETOEditField.Value = ' ';
3901            clearFLSR(app);
3902            clearAirplane(app);
3903            app.AirplaneSim = app.Airplane;
3904            app.Comminput.Data = {};
3905            app.Comminput.msg = {' '};
3906            app.Simuladorseleccion.Visible = 'on';
3907        end
3908
3909        % Value changed function: DropDown
3910        function DropDownValueChanged(app, event)
3911            value = app.DropDown.Value;
3912            if value == '0'
3913                app.Comminput.Data = {};
3914            else
3915                i = char2dec(app,value);
3916                app.Comminput.Data{1} = app.AirplaneSim(i).Callsign;
3917                if ~ischar(app.AirplaneSim(i).Incident.Target)
3918                    if app.AirplaneSim(i).Incident.Type == '1'
3919                        FLchar = dec2char(app,app.AirplaneSim(i).Incident.Target)
                                ;
3920                        app.Comminput.Data{2} = char(['FL',FLchar]);
3921                    else
3922                        app.Comminput.Data{2} = ' ';
3923                    end
3924                else
3925                    app.Comminput.Data{2} = app.AirplaneSim(i).Incident.Target;
3926                end
3927                if app.AirplaneSim(i).Incident.Time == '1'
3928                    app.Comminput.Data{3} = ' antes';
3929                elseif app.AirplaneSim(i).Incident.Time == '2'
3930                    app.Comminput.Data{3} = ' después';
3931                end
3932                app.Comminput.Data{4} = app.AirplaneSim(i).Incident.Trigger;
3933            end
```

```
3934            app.Comminput.msg = {' '};
3935            app.InputmsgTextArea.Value = ' ';
3936        end
3937
3938        % Button pushed function: AutorizarEntradaButton
3939        function AutorizarEntradaButtonPushed(app, event)
3940            if app.DropDown.Value ~= '0'
3941                app.Comminput.msg = cell(1,3);
3942                app.Comminput.msg{1} = app.ATCCommLine{1}{1}{1};
3943                app.Comminput.msg{2} = app.Comminput.Data{1};
3944                app.Comminput.msg{3} = app.ATCCommLine{1}{2}{1};
3945                msg = app.Comminput.msg;
3946                app.InputmsgTextArea.Value = char([msg{1},msg{2},msg{3}]);
3947            end
3948        end
3949
3950        % Button pushed function: AutorizarCambioFLButton
3951        function AutorizarCambioFLButtonPushed(app, event)
3952            if app.DropDown.Value ~= '0'
3953                app.Comminput.msg = cell(1,7);
3954                app.Comminput.msg{1} = app.ATCCommLine{2}{1}{1};
3955                app.Comminput.msg{2} = app.Comminput.Data{1};
3956                app.Comminput.msg{3} = app.ATCCommLine{2}{2}{1};
3957                app.Comminput.msg{4} = app.Comminput.Data{2};
3958                app.Comminput.msg{5} = app.Comminput.Data{3};
3959                app.Comminput.msg{6} = app.ATCCommLine{2}{3}{1};
3960                app.Comminput.msg{7} = app.Comminput.Data{4};
3961                msg = app.Comminput.msg;
3962                app.InputmsgTextArea.Value = char([msg{1},msg{2},msg{3},msg{4},
                        msg{5},msg{6},msg{7}]);
3963                i = char2dec(app,app.DropDown.Value);
3964                app.SimData.Autoriz.Type(i) = 1;
3965            end
3966        end
3967
3968        % Button pushed function: NoAutorizarCambioFLButton
3969        function NoAutorizarCambioFLButtonPushed(app, event)
3970            if app.DropDown.Value ~= '0'
3971                app.Comminput.msg = cell(1,3);
3972                app.Comminput.msg{1} = app.ATCCommLine{3}{1}{1};
3973                app.Comminput.msg{2} = app.Comminput.Data{1};
3974                app.Comminput.msg{3} = app.ATCCommLine{3}{2}{1};
3975                msg = app.Comminput.msg;
3976                app.InputmsgTextArea.Value = char([msg{1},msg{2},msg{3}]);
3977                i = char2dec(app,app.DropDown.Value);
3978                app.SimData.DoneIncident(i) = true;
3979            end
3980        end
3981
3982        % Button pushed function: AutorizarDirectoButton
3983        function AutorizarDirectoButtonPushed(app, event)
3984            if app.DropDown.Value ~= '0'
3985                app.Comminput.msg = cell(1,7);
3986                app.Comminput.msg{1} = app.ATCCommLine{4}{1}{1};
3987                app.Comminput.msg{2} = app.Comminput.Data{1};
3988                app.Comminput.msg{3} = app.ATCCommLine{4}{2}{1};
3989                app.Comminput.msg{4} = app.Comminput.Data{2};
```

```
3990                app.Comminput.msg{5} = app.Comminput.Data{3};
3991                app.Comminput.msg{6} = app.ATCCommLine{4}{3}{1};
3992                app.Comminput.msg{7} = app.Comminput.Data{4};
3993                msg = app.Comminput.msg;
3994                app.InputmsgTextArea.Value = char([msg{1},msg{2},msg{3},msg{4},
                        msg{5},msg{6},msg{7}]);
3995                i = char2dec(app,app.DropDown.Value);
3996                app.SimData.Autoriz.Type(i) = 2;
3997            end
3998        end
3999
4000        % Button pushed function: NoAutorizarDirectoButton
4001        function NoAutorizarDirectoButtonPushed(app, event)
4002            if app.DropDown.Value ~= '0'
4003                app.Comminput.msg = cell(1,3);
4004                app.Comminput.msg{1} = app.ATCCommLine{5}{1}{1};
4005                app.Comminput.msg{2} = app.Comminput.Data{1};
4006                app.Comminput.msg{3} = app.ATCCommLine{5}{2}{1};
4007                msg = app.Comminput.msg;
4008                app.InputmsgTextArea.Value = char([msg{1},msg{2},msg{3}]);
4009                i = char2dec(app,app.DropDown.Value);
4010                app.SimData.DoneIncident(i) = true;
4011            end
4012        end
4013
4014        % Button pushed function: CambioaRumboAutorizadoButton
4015        function CambioaRumboAutorizadoButtonPushed(app, event)
4016            if app.DropDown.Value ~= '0'
4017                i = char2dec(app,app.DropDown.Value);
4018                if app.AirplaneSim(i).Incident.Type == '3'
4019                    app.Comminput.msg = cell(1,7);
4020                    app.Comminput.msg{1} = app.Comminput.Data{1};
4021                    app.Comminput.msg{2} = app.ATCCommLine{6}{1}{1};
4022                    postrig = findwpsim(app,app.Comminput.Data{4});
4023                    for k = 1:length(app.AirplaneSim(i).Route)
4024                        L = find(app.Comminput.Data{4}==app.AirplaneSim(i).Route{
                                k});
4025                        if length(L) == 5
4026                            postarg = findwpsim(app,app.AirplaneSim(i).Route{k+1})
                                    ;
4027                            break
4028                        end
4029                    end
4030                    Rad = encuentraRadial(app,postrig,postarg);
4031                    achdg = app.AirplaneSim(i).HDG;
4032                    turn = (achdg-Rad)/abs(achdg-Rad);
4033                    if abs(achdg-Rad) > 180
4034                        achdg = achdg - turn*360;
4035                    end
4036                    turn = (achdg-Rad)/abs(achdg-Rad);
4037                    nwhdg = Rad - turn*90;
4038                    if nwhdg > 359
4039                        nwhdg = nwhdg-360;
4040                    elseif nwhdg < 0
4041                        nwhdg = nwhdg + 360;
4042                    end
4043                    NWHdgChar = dec2char(app,nwhdg);
```

```
4044                    Radchar = dec2char(app,Rad);
4045                    app.Comminput.msg{3} = NWHdgChar;
4046                    app.Comminput.msg{4} = app.ATCCommLine{6}{2}{1};
4047                    app.Comminput.msg{5} = Radchar;
4048                    app.Comminput.msg{6} = app.ATCCommLine{6}{3}{1};
4049                    app.Comminput.msg{7} = app.Comminput.Data{4};
4050                    msg = app.Comminput.msg;
4051                    app.InputmsgTextArea.Value = char([msg{1},msg{2},msg{3},msg
                            {4},msg{5},msg{6},msg{7}]);
4052                    app.SimData.Autoriz.Type(i) = 3;
4053                end
4054            end
4055        end
4056
4057        % Button pushed function: CambioFrecuenciaButton
4058        function CambioFrecuenciaButtonPushed(app, event)
4059            if app.DropDown.Value ~= '0'
4060                app.Comminput.msg = cell(1,5);
4061                app.Comminput.msg{1} = app.ATCCommLine{7}{1}{1};
4062                app.Comminput.msg{2} = app.Comminput.Data{1};
4063                app.Comminput.msg{3} = app.ATCCommLine{7}{2}{1};
4064                i = char2dec(app,app.DropDown.Value);
4065                WPexit = app.AirplaneSim(i).Route{end};
4066                k = locatewp(app,WPexit);
4067                Freq = app.SectorSimulador.Sector.Waypoint(k).Freq;
4068                if ~isempty(Freq)
4069                    Freqchar = dec2char(app,Freq);
4070                    app.Comminput.msg{4} = char([Freqchar,' MHz']);
4071                else
4072                    app.Comminput.msg{4} = '';
4073                end
4074                app.Comminput.msg{5} = app.ATCCommLine{7}{3}{1};
4075                msg = app.Comminput.msg;
4076                app.InputmsgTextArea.Value = char([msg{1},msg{2},msg{3},msg{4},
                        msg{5}]);
4077                app.SimData.Autoriz.Type(i) = 4;
4078            end
4079        end
4080
4081        % Button pushed function: SendButton
4082        function SendButtonPushed(app, event)
4083            msg = app.InputmsgTextArea.Value;
4084            comm = app.CommsTextArea.Value;
4085            if length(comm) == 1 && comm{1}(1) == ' '
4086                Text = sprintf('%s %s\n ','Control:',char(msg));
4087            else
4088                Text = sprintf('%s %s\n\n','Control:',char(msg));
4089            end
4090            Comm = cell(length(comm)+1,1);
4091            for t = 1:length(comm) + 1
4092                if t == 1
4093                    Comm{1} = Text;
4094                else
4095                    Comm{t} = app.CommsTextArea.Value{t-1};
4096                end
4097            end
4098            app.CommsTextArea.Value = Comm;
```

```
4099            i = char2dec(app,app.DropDown.Value);
4100            if i ~= 0
4101                t = app.SimData.Autoriz.Type(i);
4102                if t ~= 0
4103                    app.SimData.Autoriz.Value(i) = true;
4104                    app.SimData.Comms(i) = 4;
4105                    if t == 3
4106                        app.AirplaneSim(i).Incident.Target = char2dec(app,app.
                                Comminput.msg{3});
4107                    end
4108                elseif t == 0 && app.SimData.DoneIncident(i)
4109                    app.SimData.Comms(i) = 4;
4110                end
4111            end
4112            app.DropDown.Value = '0';
4113            DropDownValueChanged(app);
4114        end
4115
4116        % Value changed function: PlaySimButton
4117        function PlaySimButtonValueChanged(app, event)
4118            value = app.PlaySimButton.Value;
4119            if value == true
4120                app.PlaySimButton.Enable = 'off';
4121                app.PauseSimButton.Value = false;
4122                app.PauseSimButton.Enable = 'on';
4123                app.SalirSimButton.Enable = 'off';
4124                app.SimTimer = timer;
4125                app.SimTimer.Period = 10;
4126                app.SimTimer.BusyMode = 'queue';
4127                Ho = app.TimeSim;
4128                Hf = app.SectorSimulador.Sector.Time.fin;
4129                app.SimTimer.TasksToExecute = seconds(Hf-Ho)/10;
4130                app.SimTimer.ExecutionMode = 'fixedRate';
4131                app.SimTimer.StartDelay = 10;
4132                app.SimTimer.StartFcn = {@StartTimer, app};
4133                app.SimTimer.TimerFcn = {@RunTimer,app};
4134                app.SimTimer.StopFcn = {@EndTimer,app};
4135                start(app.SimTimer);
4136            end
4137            function StartTimer(~,~,app)
4138                app.TimeSimEditField.Value = char(app.TimeSim,'hh:mm:ss');
4139                SimNav(app);
4140                ETOCheck(app);
4141                STCACheck(app);
4142                CommCheck(app);
4143            end
4144            function RunTimer(~,~,app)
4145                app.TimeSim = app.TimeSim + duration(0,0,10);
4146                app.TimeSimEditField.Value = char(app.TimeSim,'hh:mm:ss');
4147                SimNav(app);
4148                ETOCheck(app);
4149                STCACheck(app);
4150                CommCheck(app);
4151            end
4152            function EndTimer(~,~,app)
4153                if ~app.PauseSimButton.Value
4154                    app.TimeSim = app.SectorSimulador.Sector.Time.init;
```

```
4155                    app.TimeSimEditField.Value = char(app.TimeSim,'hh:mm:ss');
4156                    app.PauseSimButton.Value = true;
4157                    app.PauseSimButton.Enable = 'off';
4158                    app.PlaySimButton.Value = false;
4159                    app.PlaySimButton.Enable = 'on';
4160                    app.SalirSimButton.Enable = 'on';
4161                    plotSimuladorgraph(app);
4162                end
4163                app.SimTimer = [];
4164            end
4165        end
4166
4167        % Value changed function: PauseSimButton
4168        function PauseSimButtonValueChanged(app, event)
4169            value = app.PauseSimButton.Value;
4170            if value == true
4171                app.PlaySimButton.Value = false;
4172                app.PlaySimButton.Enable = 'on';
4173                app.PauseSimButton.Enable = 'off';
4174                app.SalirSimButton.Enable = 'on';
4175                stop(app.SimTimer);
4176            end
4177        end
4178    end
4179
4180    % Component initialization
4181    methods (Access = private)
4182
4183        % Create UIFigure and components
4184        function createComponents(app)
4185
4186            % Create ATCMakerApp and hide until all components are created
4187            app.ATCMakerApp = uifigure('Visible', 'off');
4188            app.ATCMakerApp.Position = [1 1 1280 720];
4189            app.ATCMakerApp.Name = 'ATC Maker';
4190
4191            % Create PantalladeInicio
4192            app.PantalladeInicio = uipanel(app.ATCMakerApp);
4193            app.PantalladeInicio.BackgroundColor = [0 0.1098 0.2196];
4194            app.PantalladeInicio.Position = [1 1 1280 720];
4195
4196            % Create EditordeEscenariosButton
4197            app.EditordeEscenariosButton = uibutton(app.PantalladeInicio, 'push'
                    );
4198            app.EditordeEscenariosButton.ButtonPushedFcn = createCallbackFcn(app
                    , @EditordeEscenariosButtonPushed, true);
4199            app.EditordeEscenariosButton.BackgroundColor = [0 0.451 0.7412];
4200            app.EditordeEscenariosButton.FontColor = [1 1 1];
4201            app.EditordeEscenariosButton.Tooltip = {'Abrir selección de
                    Escenarios para el Editor'};
4202            app.EditordeEscenariosButton.Position = [554 320 176 52];
4203            app.EditordeEscenariosButton.Text = 'Editor de Escenarios';
4204
4205            % Create SimuladorButton
4206            app.SimuladorButton = uibutton(app.PantalladeInicio, 'push');
4207            app.SimuladorButton.ButtonPushedFcn = createCallbackFcn(app,
                    @SimuladorButtonPushed, true);
```

```
4208            app.SimuladorButton.BackgroundColor = [0 0.4471 0.7412];
4209            app.SimuladorButton.FontWeight = 'bold';
4210            app.SimuladorButton.FontColor = [1 1 1];
4211            app.SimuladorButton.Tooltip = {'Abrir selección de Escenarios para
                    el Simulador'};
4212            app.SimuladorButton.Position = [554 260 176 52];
4213            app.SimuladorButton.Text = 'Simulador';
4214
4215            % Create ATCMakerLogo
4216            app.ATCMakerLogo = uiimage(app.PantalladeInicio);
4217            app.ATCMakerLogo.Position = [514 392 254 258];
4218            app.ATCMakerLogo.ImageSource = 'ATC Maker Logo.png';
4219
4220            % Create EditordeEscenarios
4221            app.EditordeEscenarios = uipanel(app.ATCMakerApp);
4222            app.EditordeEscenarios.Visible = 'off';
4223            app.EditordeEscenarios.BackgroundColor = [0 0.1098 0.2196];
4224            app.EditordeEscenarios.Position = [1 1 1280 720];
4225
4226            % Create Editorbackbutton
4227            app.Editorbackbutton = uibutton(app.EditordeEscenarios, 'push');
4228            app.Editorbackbutton.ButtonPushedFcn = createCallbackFcn(app,
                    @EditorbackbuttonButtonPushed, true);
4229            app.Editorbackbutton.BackgroundColor = [0 0.451 0.7412];
4230            app.Editorbackbutton.FontColor = [1 1 1];
4231            app.Editorbackbutton.Position = [1100 640 100 30];
4232            app.Editorbackbutton.Text = 'Atrás';
4233
4234            % Create TreeSector
4235            app.TreeSector = uitree(app.EditordeEscenarios);
4236            app.TreeSector.SelectionChangedFcn = createCallbackFcn(app,
                    @TreeSectorSelectionChanged, true);
4237            app.TreeSector.Position = [775 235 284 415];
4238
4239            % Create SectorBorder
4240            app.SectorBorder = uitreenode(app.TreeSector);
4241            app.SectorBorder.NodeData = 1;
4242            app.SectorBorder.Text = 'Frontera del Sector';
4243
4244            % Create ZonasRestringidas
4245            app.ZonasRestringidas = uitreenode(app.TreeSector);
4246            app.ZonasRestringidas.NodeData = 2;
4247            app.ZonasRestringidas.Text = 'Zonas Restringidas';
4248
4249            % Create Aviones
4250            app.Aviones = uitreenode(app.TreeSector);
4251            app.Aviones.NodeData = 3;
4252            app.Aviones.Text = 'Aviones';
4253
4254            % Create Aeropuertos
4255            app.Aeropuertos = uitreenode(app.TreeSector);
4256            app.Aeropuertos.NodeData = 4;
4257            app.Aeropuertos.Text = 'Aeropuertos';
4258
4259            % Create Waypoints
4260            app.Waypoints = uitreenode(app.TreeSector);
4261            app.Waypoints.NodeData = 5;
```

```
4262            app.Waypoints.Text = 'Puntos de Paso';
4263
4264            % Create SectorBorderPanel
4265            app.SectorBorderPanel = uipanel(app.EditordeEscenarios);
4266            app.SectorBorderPanel.Visible = 'off';
4267            app.SectorBorderPanel.BackgroundColor = [0.0196 0.0196 0.3804];
4268            app.SectorBorderPanel.Position = [50 25 1100 170];
4269
4270            % Create BorderVertexPanel
4271            app.BorderVertexPanel = uipanel(app.SectorBorderPanel);
4272            app.BorderVertexPanel.Visible = 'off';
4273            app.BorderVertexPanel.BackgroundColor = [0.0196 0.0196 0.3804];
4274            app.BorderVertexPanel.Position = [160 8 507 154];
4275
4276            % Create XnEditFieldLabel
4277            app.XnEditFieldLabel = uilabel(app.BorderVertexPanel);
4278            app.XnEditFieldLabel.HorizontalAlignment = 'right';
4279            app.XnEditFieldLabel.FontColor = [1 1 1];
4280            app.XnEditFieldLabel.Position = [38 78 25 22];
4281            app.XnEditFieldLabel.Text = 'Xn';
4282
4283            % Create XnvertEditField
4284            app.XnvertEditField = uieditfield(app.BorderVertexPanel, 'numeric');
4285            app.XnvertEditField.ValueChangedFcn = createCallbackFcn(app,
                    @XnvertEditFieldValueChanged, true);
4286            app.XnvertEditField.Position = [78 78 90 22];
4287
4288            % Create YnEditFieldLabel
4289            app.YnEditFieldLabel = uilabel(app.BorderVertexPanel);
4290            app.YnEditFieldLabel.HorizontalAlignment = 'right';
4291            app.YnEditFieldLabel.FontColor = [1 1 1];
4292            app.YnEditFieldLabel.Position = [223 78 25 22];
4293            app.YnEditFieldLabel.Text = 'Yn';
4294
4295            % Create YnvertEditField
4296            app.YnvertEditField = uieditfield(app.BorderVertexPanel, 'numeric');
4297            app.YnvertEditField.ValueChangedFcn = createCallbackFcn(app,
                    @YnvertEditFieldValueChanged, true);
4298            app.YnvertEditField.Position = [263 78 90 22];
4299
4300            % Create Xn1EditFieldLabel
4301            app.Xn1EditFieldLabel = uilabel(app.BorderVertexPanel);
4302            app.Xn1EditFieldLabel.HorizontalAlignment = 'right';
4303            app.Xn1EditFieldLabel.FontColor = [1 1 1];
4304            app.Xn1EditFieldLabel.Position = [29 40 34 22];
4305            app.Xn1EditFieldLabel.Text = 'Xn+1';
4306
4307            % Create Xn1vertEditField
4308            app.Xn1vertEditField = uieditfield(app.BorderVertexPanel, 'numeric')
                    ;
4309            app.Xn1vertEditField.ValueChangedFcn = createCallbackFcn(app,
                    @Xn1vertEditFieldValueChanged, true);
4310            app.Xn1vertEditField.Position = [78 40 90 22];
4311
4312            % Create Yn1EditFieldLabel
4313            app.Yn1EditFieldLabel = uilabel(app.BorderVertexPanel);
4314            app.Yn1EditFieldLabel.HorizontalAlignment = 'right';
```

```
4315            app.Yn1EditFieldLabel.FontColor = [1 1 1];
4316            app.Yn1EditFieldLabel.Position = [223 40 34 22];
4317            app.Yn1EditFieldLabel.Text = 'Yn+1';
4318
4319            % Create Yn1vertEditField
4320            app.Yn1vertEditField = uieditfield(app.BorderVertexPanel, 'numeric')
                    ;
4321            app.Yn1vertEditField.ValueChangedFcn = createCallbackFcn(app,
                    @Yn1vertEditFieldValueChanged, true);
4322            app.Yn1vertEditField.Position = [263 40 90 22];
4323
4324            % Create FronteralneavrticeLabel
4325            app.FronteralneavrticeLabel = uilabel(app.BorderVertexPanel);
4326            app.FronteralneavrticeLabel.BackgroundColor = [0 0.4471 0.7412];
4327            app.FronteralneavrticeLabel.FontColor = [1 1 1];
4328            app.FronteralneavrticeLabel.Position = [29 118 119 22];
4329            app.FronteralneavrticeLabel.Text = 'Frontera línea vértice';
4330
4331            % Create MostrarBorderVButton
4332            app.MostrarBorderVButton = uibutton(app.BorderVertexPanel, 'push');
4333            app.MostrarBorderVButton.ButtonPushedFcn = createCallbackFcn(app,
                    @MostrarBorderVButtonPushed, true);
4334            app.MostrarBorderVButton.BackgroundColor = [0 0.451 0.7412];
4335            app.MostrarBorderVButton.FontColor = [1 1 1];
4336            app.MostrarBorderVButton.Position = [394 97 100 22];
4337            app.MostrarBorderVButton.Text = 'Mostrar';
4338
4339            % Create GuardarBorderVButton
4340            app.GuardarBorderVButton = uibutton(app.BorderVertexPanel, 'push');
4341            app.GuardarBorderVButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarBorderVButtonPushed, true);
4342            app.GuardarBorderVButton.BackgroundColor = [0 0.451 0.7412];
4343            app.GuardarBorderVButton.FontColor = [1 1 1];
4344            app.GuardarBorderVButton.Position = [394 57 100 22];
4345            app.GuardarBorderVButton.Text = 'Guardar';
4346
4347            % Create TipodeFronteraButtonGroup
4348            app.TipodeFronteraButtonGroup = uibuttongroup(app.SectorBorderPanel)
                    ;
4349            app.TipodeFronteraButtonGroup.SelectionChangedFcn =
                    createCallbackFcn(app,
                    @TipodeFronteraButtonGroupSelectionChanged, true);
4350            app.TipodeFronteraButtonGroup.ForegroundColor = [1 1 1];
4351            app.TipodeFronteraButtonGroup.Title = 'Tipo de Frontera';
4352            app.TipodeFronteraButtonGroup.BackgroundColor = [0 0.4471 0.7412];
4353            app.TipodeFronteraButtonGroup.Position = [30 22 121 126];
4354
4355            % Create SeleccionaunaButton
4356            app.SeleccionaunaButton = uitogglebutton(app.
                    TipodeFronteraButtonGroup);
4357            app.SeleccionaunaButton.Text = 'Selecciona una';
4358            app.SeleccionaunaButton.Position = [11 73 100 22];
4359            app.SeleccionaunaButton.Value = true;
4360
4361            % Create VerticeButton
4362            app.VerticeButton = uitogglebutton(app.TipodeFronteraButtonGroup);
4363            app.VerticeButton.Text = 'Vertice';
```

```
4364            app.VerticeButton.Position = [11 52 100 22];
4365
4366            % Create ArcoButton
4367            app.ArcoButton = uitogglebutton(app.TipodeFronteraButtonGroup);
4368            app.ArcoButton.Text = 'Arco';
4369            app.ArcoButton.Position = [11 31 100 22];
4370
4371            % Create AltitudButton
4372            app.AltitudButton = uitogglebutton(app.TipodeFronteraButtonGroup);
4373            app.AltitudButton.Text = 'Altitud';
4374            app.AltitudButton.Position = [12 10 100 22];
4375
4376            % Create BorderTree
4377            app.BorderTree = uitree(app.SectorBorderPanel);
4378            app.BorderTree.SelectionChangedFcn = createCallbackFcn(app,
                    @BorderTreeSelectionChanged, true);
4379            app.BorderTree.Position = [777 8 315 154];
4380
4381            % Create EliminarBorderButton
4382            app.EliminarBorderButton = uibutton(app.SectorBorderPanel, 'push');
4383            app.EliminarBorderButton.ButtonPushedFcn = createCallbackFcn(app,
                    @EliminarBorderButtonPushed, true);
4384            app.EliminarBorderButton.BackgroundColor = [0 0.451 0.7412];
4385            app.EliminarBorderButton.FontColor = [1 1 1];
4386            app.EliminarBorderButton.Position = [679 78 84 22];
4387            app.EliminarBorderButton.Text = 'Eliminar';
4388
4389            % Create BorderArchPanel
4390            app.BorderArchPanel = uipanel(app.SectorBorderPanel);
4391            app.BorderArchPanel.Visible = 'off';
4392            app.BorderArchPanel.BackgroundColor = [0.0196 0.0196 0.3804];
4393            app.BorderArchPanel.Position = [160 8 507 154];
4394
4395            % Create XnEditFieldLabel_2
4396            app.XnEditFieldLabel_2 = uilabel(app.BorderArchPanel);
4397            app.XnEditFieldLabel_2.HorizontalAlignment = 'right';
4398            app.XnEditFieldLabel_2.FontColor = [1 1 1];
4399            app.XnEditFieldLabel_2.Position = [38 88 25 19];
4400            app.XnEditFieldLabel_2.Text = 'Xn';
4401
4402            % Create XnarchEditField
4403            app.XnarchEditField = uieditfield(app.BorderArchPanel, 'numeric');
4404            app.XnarchEditField.ValueChangedFcn = createCallbackFcn(app,
                    @XnarchEditFieldValueChanged, true);
4405            app.XnarchEditField.Position = [78 88 90 21];
4406
4407            % Create YnEditFieldLabel_2
4408            app.YnEditFieldLabel_2 = uilabel(app.BorderArchPanel);
4409            app.YnEditFieldLabel_2.HorizontalAlignment = 'right';
4410            app.YnEditFieldLabel_2.FontColor = [1 1 1];
4411            app.YnEditFieldLabel_2.Position = [223 89 25 20];
4412            app.YnEditFieldLabel_2.Text = 'Yn';
4413
4414            % Create YnarchEditField
4415            app.YnarchEditField = uieditfield(app.BorderArchPanel, 'numeric');
4416            app.YnarchEditField.ValueChangedFcn = createCallbackFcn(app,
                    @YnarchEditFieldValueChanged, true);
```

```
4417            app.YnarchEditField.Position = [263 88 90 21];
4418
4419            % Create Xn1EditFieldLabel_2
4420            app.Xn1EditFieldLabel_2 = uilabel(app.BorderArchPanel);
4421            app.Xn1EditFieldLabel_2.HorizontalAlignment = 'right';
4422            app.Xn1EditFieldLabel_2.FontColor = [1 1 1];
4423            app.Xn1EditFieldLabel_2.Position = [29 60 34 17];
4424            app.Xn1EditFieldLabel_2.Text = 'Xn+1';
4425
4426            % Create Xn1archEditField
4427            app.Xn1archEditField = uieditfield(app.BorderArchPanel, 'numeric');
4428            app.Xn1archEditField.ValueChangedFcn = createCallbackFcn(app,
                    @Xn1archEditFieldValueChanged, true);
4429            app.Xn1archEditField.Position = [78 57 90 22];
4430
4431            % Create Yn1EditFieldLabel_2
4432            app.Yn1EditFieldLabel_2 = uilabel(app.BorderArchPanel);
4433            app.Yn1EditFieldLabel_2.HorizontalAlignment = 'right';
4434            app.Yn1EditFieldLabel_2.FontColor = [1 1 1];
4435            app.Yn1EditFieldLabel_2.Position = [223 60 34 17];
4436            app.Yn1EditFieldLabel_2.Text = 'Yn+1';
4437
4438            % Create Yn1archEditField
4439            app.Yn1archEditField = uieditfield(app.BorderArchPanel, 'numeric');
4440            app.Yn1archEditField.ValueChangedFcn = createCallbackFcn(app,
                    @Yn1archEditFieldValueChanged, true);
4441            app.Yn1archEditField.Position = [263 60 90 19];
4442
4443            % Create FronteraarcoLabel
4444            app.FronteraarcoLabel = uilabel(app.BorderArchPanel);
4445            app.FronteraarcoLabel.BackgroundColor = [0 0.4471 0.7412];
4446            app.FronteraarcoLabel.FontColor = [1 1 1];
4447            app.FronteraarcoLabel.Position = [29 118 78 22];
4448            app.FronteraarcoLabel.Text = 'Frontera arco';
4449
4450            % Create MostrarBorderAButton
4451            app.MostrarBorderAButton = uibutton(app.BorderArchPanel, 'push');
4452            app.MostrarBorderAButton.ButtonPushedFcn = createCallbackFcn(app,
                    @MostrarBorderAButtonPushed, true);
4453            app.MostrarBorderAButton.BackgroundColor = [0 0.451 0.7412];
4454            app.MostrarBorderAButton.FontColor = [1 1 1];
4455            app.MostrarBorderAButton.Position = [394 97 100 22];
4456            app.MostrarBorderAButton.Text = 'Mostrar';
4457
4458            % Create GuardarBorderAButton
4459            app.GuardarBorderAButton = uibutton(app.BorderArchPanel, 'push');
4460            app.GuardarBorderAButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarBorderAButtonPushed, true);
4461            app.GuardarBorderAButton.BackgroundColor = [0 0.451 0.7412];
4462            app.GuardarBorderAButton.FontColor = [1 1 1];
4463            app.GuardarBorderAButton.Position = [394 57 100 22];
4464            app.GuardarBorderAButton.Text = 'Guardar';
4465
4466            % Create RadioLabel
4467            app.RadioLabel = uilabel(app.BorderArchPanel);
4468            app.RadioLabel.HorizontalAlignment = 'right';
4469            app.RadioLabel.FontColor = [1 1 1];
```

```
4470            app.RadioLabel.Position = [26 24 37 23];
4471            app.RadioLabel.Text = 'Radio';
4472
4473            % Create BARadiusEditField
4474            app.BARadiusEditField = uieditfield(app.BorderArchPanel, 'numeric');
4475            app.BARadiusEditField.Limits = [0 Inf];
4476            app.BARadiusEditField.ValueChangedFcn = createCallbackFcn(app,
                    @BARadiusEditFieldValueChanged, true);
4477            app.BARadiusEditField.Position = [78 24 90 24];
4478
4479            % Create BAConcaveConvexSwitch
4480            app.BAConcaveConvexSwitch = uiswitch(app.BorderArchPanel, 'slider');
4481            app.BAConcaveConvexSwitch.Items = {'^', 'v'};
4482            app.BAConcaveConvexSwitch.ValueChangedFcn = createCallbackFcn(app,
                    @BAConcaveConvexSwitchValueChanged, true);
4483            app.BAConcaveConvexSwitch.FontColor = [1 1 1];
4484            app.BAConcaveConvexSwitch.Position = [209 24 52 23];
4485            app.BAConcaveConvexSwitch.Value = '^';
4486
4487            % Create BACortoLargoSwitch
4488            app.BACortoLargoSwitch = uiswitch(app.BorderArchPanel, 'slider');
4489            app.BACortoLargoSwitch.Items = {'Corto', 'Largo'};
4490            app.BACortoLargoSwitch.ValueChangedFcn = createCallbackFcn(app,
                    @BACortoLargoSwitchValueChanged, true);
4491            app.BACortoLargoSwitch.FontColor = [1 1 1];
4492            app.BACortoLargoSwitch.Position = [326 24 54 24];
4493            app.BACortoLargoSwitch.Value = 'Corto';
4494
4495            % Create BorderAltitudePanel
4496            app.BorderAltitudePanel = uipanel(app.SectorBorderPanel);
4497            app.BorderAltitudePanel.Visible = 'off';
4498            app.BorderAltitudePanel.BackgroundColor = [0.0196 0.0196 0.3804];
4499            app.BorderAltitudePanel.Position = [160 8 507 154];
4500
4501            % Create LmiteInferiorLabel
4502            app.LmiteInferiorLabel = uilabel(app.BorderAltitudePanel);
4503            app.LmiteInferiorLabel.HorizontalAlignment = 'center';
4504            app.LmiteInferiorLabel.FontColor = [1 1 1];
4505            app.LmiteInferiorLabel.Position = [53 72 43 35];
4506            app.LmiteInferiorLabel.Text = {'Límite'; 'Inferior'};
4507
4508            % Create ZinfEditField
4509            app.ZinfEditField = uieditfield(app.BorderAltitudePanel, 'numeric');
4510            app.ZinfEditField.Limits = [0 Inf];
4511            app.ZinfEditField.ValueDisplayFormat = '%.0f';
4512            app.ZinfEditField.ValueChangedFcn = createCallbackFcn(app,
                    @ZinfEditFieldValueChanged, true);
4513            app.ZinfEditField.Position = [111 78 99 22];
4514
4515            % Create LmiteSuperiorLabel
4516            app.LmiteSuperiorLabel = uilabel(app.BorderAltitudePanel);
4517            app.LmiteSuperiorLabel.HorizontalAlignment = 'center';
4518            app.LmiteSuperiorLabel.FontColor = [1 1 1];
4519            app.LmiteSuperiorLabel.Position = [53 34 51 33];
4520            app.LmiteSuperiorLabel.Text = {'Límite'; 'Superior'};
4521
4522            % Create ZsupEditField
```

```
4523            app.ZsupEditField = uieditfield(app.BorderAltitudePanel, 'numeric');
4524            app.ZsupEditField.Limits = [0 Inf];
4525            app.ZsupEditField.ValueDisplayFormat = '%.0f';
4526            app.ZsupEditField.ValueChangedFcn = createCallbackFcn(app,
                    @ZsupEditFieldValueChanged, true);
4527            app.ZsupEditField.Position = [111 40 99 22];
4528
4529            % Create LmitesverticalesLabel
4530            app.LmitesverticalesLabel = uilabel(app.BorderAltitudePanel);
4531            app.LmitesverticalesLabel.BackgroundColor = [0 0.4471 0.7412];
4532            app.LmitesverticalesLabel.FontColor = [1 1 1];
4533            app.LmitesverticalesLabel.Position = [29 118 98 22];
4534            app.LmitesverticalesLabel.Text = 'Límites verticales';
4535
4536            % Create GuardarBorderAltButton
4537            app.GuardarBorderAltButton = uibutton(app.BorderAltitudePanel, 'push
                    ');
4538            app.GuardarBorderAltButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarBorderAltButtonPushed, true);
4539            app.GuardarBorderAltButton.BackgroundColor = [0 0.451 0.7412];
4540            app.GuardarBorderAltButton.FontColor = [1 1 1];
4541            app.GuardarBorderAltButton.Position = [352 70 108 22];
4542            app.GuardarBorderAltButton.Text = 'Guardar';
4543
4544            % Create FLftinfDropDown
4545            app.FLftinfDropDown = uidropdown(app.BorderAltitudePanel);
4546            app.FLftinfDropDown.Items = {'FL', 'ft'};
4547            app.FLftinfDropDown.ItemsData = {'1', '2'};
4548            app.FLftinfDropDown.Position = [237 78 48 22];
4549            app.FLftinfDropDown.Value = '1';
4550
4551            % Create FLftsupDropDown
4552            app.FLftsupDropDown = uidropdown(app.BorderAltitudePanel);
4553            app.FLftsupDropDown.Items = {'FL', 'ft'};
4554            app.FLftsupDropDown.ItemsData = {'1', '2'};
4555            app.FLftsupDropDown.Position = [237 40 48 22];
4556            app.FLftsupDropDown.Value = '1';
4557
4558            % Create Editorgraph
4559            app.Editorgraph = uiaxes(app.EditordeEscenarios);
4560            title(app.Editorgraph, '')
4561            xlabel(app.Editorgraph, '')
4562            ylabel(app.Editorgraph, '')
4563            app.Editorgraph.PlotBoxAspectRatio = [1 1 1];
4564            app.Editorgraph.GridColor = [0 1 0];
4565            app.Editorgraph.MinorGridColor = [0 1 0];
4566            app.Editorgraph.XColor = [0 1 0];
4567            app.Editorgraph.YColor = [0 1 0];
4568            app.Editorgraph.ZColor = [0 1 0];
4569            app.Editorgraph.Color = [0 0 0];
4570            app.Editorgraph.XGrid = 'on';
4571            app.Editorgraph.YGrid = 'on';
4572            app.Editorgraph.BackgroundColor = [0 0.1098 0.2196];
4573            app.Editorgraph.Position = [150 204 491 482];
4574
4575            % Create FLsupEditField
4576            app.FLsupEditField = uieditfield(app.EditordeEscenarios, 'text');
```

```
4577            app.FLsupEditField.Editable = 'off';
4578            app.FLsupEditField.HorizontalAlignment = 'right';
4579            app.FLsupEditField.FontSize = 14;
4580            app.FLsupEditField.FontColor = [0 1 0];
4581            app.FLsupEditField.BackgroundColor = [0 0 0];
4582            app.FLsupEditField.Position = [651 639 108 25];
4583
4584            % Create FLinfEditField
4585            app.FLinfEditField = uieditfield(app.EditordeEscenarios, 'text');
4586            app.FLinfEditField.Editable = 'off';
4587            app.FLinfEditField.HorizontalAlignment = 'right';
4588            app.FLinfEditField.FontSize = 14;
4589            app.FLinfEditField.FontColor = [0 1 0];
4590            app.FLinfEditField.BackgroundColor = [0 0 0];
4591            app.FLinfEditField.Position = [651 614 108 25];
4592
4593            % Create SectorRestricZonePanel
4594            app.SectorRestricZonePanel = uipanel(app.EditordeEscenarios);
4595            app.SectorRestricZonePanel.Visible = 'off';
4596            app.SectorRestricZonePanel.BackgroundColor = [0.0196 0.0196 0.3804];
4597            app.SectorRestricZonePanel.Position = [50 25 1100 170];
4598
4599            % Create RZVertexPanel
4600            app.RZVertexPanel = uipanel(app.SectorRestricZonePanel);
4601            app.RZVertexPanel.Visible = 'off';
4602            app.RZVertexPanel.BackgroundColor = [0.0196 0.0196 0.3804];
4603            app.RZVertexPanel.Position = [160 8 507 154];
4604
4605            % Create XnEditFieldLabel_3
4606            app.XnEditFieldLabel_3 = uilabel(app.RZVertexPanel);
4607            app.XnEditFieldLabel_3.HorizontalAlignment = 'right';
4608            app.XnEditFieldLabel_3.FontColor = [1 1 1];
4609            app.XnEditFieldLabel_3.Position = [38 78 25 22];
4610            app.XnEditFieldLabel_3.Text = 'Xn';
4611
4612            % Create RZXnvertEditField
4613            app.RZXnvertEditField = uieditfield(app.RZVertexPanel, 'numeric');
4614            app.RZXnvertEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZXnvertEditFieldValueChanged, true);
4615            app.RZXnvertEditField.Position = [78 78 90 22];
4616
4617            % Create YnEditFieldLabel_3
4618            app.YnEditFieldLabel_3 = uilabel(app.RZVertexPanel);
4619            app.YnEditFieldLabel_3.HorizontalAlignment = 'right';
4620            app.YnEditFieldLabel_3.FontColor = [1 1 1];
4621            app.YnEditFieldLabel_3.Position = [223 78 25 22];
4622            app.YnEditFieldLabel_3.Text = 'Yn';
4623
4624            % Create RZYnvertEditField
4625            app.RZYnvertEditField = uieditfield(app.RZVertexPanel, 'numeric');
4626            app.RZYnvertEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZYnvertEditFieldValueChanged, true);
4627            app.RZYnvertEditField.Position = [263 78 90 22];
4628
4629            % Create Xn1EditFieldLabel_3
4630            app.Xn1EditFieldLabel_3 = uilabel(app.RZVertexPanel);
4631            app.Xn1EditFieldLabel_3.HorizontalAlignment = 'right';
```

```
4632            app.Xn1EditFieldLabel_3.FontColor = [1 1 1];
4633            app.Xn1EditFieldLabel_3.Position = [29 40 34 22];
4634            app.Xn1EditFieldLabel_3.Text = 'Xn+1';
4635
4636            % Create RZXn1vertEditField
4637            app.RZXn1vertEditField = uieditfield(app.RZVertexPanel, 'numeric');
4638            app.RZXn1vertEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZXn1vertEditFieldValueChanged, true);
4639            app.RZXn1vertEditField.Position = [78 40 90 22];
4640
4641            % Create Yn1EditFieldLabel_3
4642            app.Yn1EditFieldLabel_3 = uilabel(app.RZVertexPanel);
4643            app.Yn1EditFieldLabel_3.HorizontalAlignment = 'right';
4644            app.Yn1EditFieldLabel_3.FontColor = [1 1 1];
4645            app.Yn1EditFieldLabel_3.Position = [223 40 34 22];
4646            app.Yn1EditFieldLabel_3.Text = 'Yn+1';
4647
4648            % Create RZYn1vertEditField
4649            app.RZYn1vertEditField = uieditfield(app.RZVertexPanel, 'numeric');
4650            app.RZYn1vertEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZYn1vertEditFieldValueChanged, true);
4651            app.RZYn1vertEditField.Position = [263 40 90 22];
4652
4653            % Create ZonaRestringidalneavrticeLabel
4654            app.ZonaRestringidalneavrticeLabel = uilabel(app.RZVertexPanel);
4655            app.ZonaRestringidalneavrticeLabel.BackgroundColor = [0 0.4471
                    0.7412];
4656            app.ZonaRestringidalneavrticeLabel.FontColor = [1 1 1];
4657            app.ZonaRestringidalneavrticeLabel.Position = [29 118 165 22];
4658            app.ZonaRestringidalneavrticeLabel.Text = 'Zona Restringida línea vé
                    rtice';
4659
4660            % Create MostrarRZVButton
4661            app.MostrarRZVButton = uibutton(app.RZVertexPanel, 'push');
4662            app.MostrarRZVButton.ButtonPushedFcn = createCallbackFcn(app,
                    @MostrarRZVButtonPushed, true);
4663            app.MostrarRZVButton.BackgroundColor = [0 0.451 0.7412];
4664            app.MostrarRZVButton.FontColor = [1 1 1];
4665            app.MostrarRZVButton.Position = [394 97 100 22];
4666            app.MostrarRZVButton.Text = 'Mostrar';
4667
4668            % Create GuardarRZVButton
4669            app.GuardarRZVButton = uibutton(app.RZVertexPanel, 'push');
4670            app.GuardarRZVButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarRZVButtonPushed, true);
4671            app.GuardarRZVButton.BackgroundColor = [0 0.451 0.7412];
4672            app.GuardarRZVButton.FontColor = [1 1 1];
4673            app.GuardarRZVButton.Position = [394 57 100 22];
4674            app.GuardarRZVButton.Text = 'Guardar';
4675
4676            % Create TipodeFronteraRZButtonGroup
4677            app.TipodeFronteraRZButtonGroup = uibuttongroup(app.
                    SectorRestricZonePanel);
4678            app.TipodeFronteraRZButtonGroup.SelectionChangedFcn =
                    createCallbackFcn(app,
                    @TipodeFronteraRZButtonGroupSelectionChanged, true);
4679            app.TipodeFronteraRZButtonGroup.ForegroundColor = [1 1 1];
```

```
4680            app.TipodeFronteraRZButtonGroup.Title = 'Zona Restr. Tipo';
4681            app.TipodeFronteraRZButtonGroup.BackgroundColor = [0 0.4471 0.7412];
4682            app.TipodeFronteraRZButtonGroup.Position = [30 22 121 126];
4683
4684            % Create SeleccionaunaButtonRZ
4685            app.SeleccionaunaButtonRZ = uitogglebutton(app.
                    TipodeFronteraRZButtonGroup);
4686            app.SeleccionaunaButtonRZ.Text = 'Selecciona una';
4687            app.SeleccionaunaButtonRZ.Position = [11 73 100 22];
4688            app.SeleccionaunaButtonRZ.Value = true;
4689
4690            % Create VerticeButtonRZ
4691            app.VerticeButtonRZ = uitogglebutton(app.TipodeFronteraRZButtonGroup
                    );
4692            app.VerticeButtonRZ.Text = 'Vertice';
4693            app.VerticeButtonRZ.Position = [11 52 100 22];
4694
4695            % Create ArcoButtonRZ
4696            app.ArcoButtonRZ = uitogglebutton(app.TipodeFronteraRZButtonGroup);
4697            app.ArcoButtonRZ.Text = 'Arco';
4698            app.ArcoButtonRZ.Position = [11 31 100 22];
4699
4700            % Create AltitudButtonRZ
4701            app.AltitudButtonRZ = uitogglebutton(app.TipodeFronteraRZButtonGroup
                    );
4702            app.AltitudButtonRZ.Text = 'Altitud';
4703            app.AltitudButtonRZ.Position = [12 10 100 22];
4704
4705            % Create RZTree
4706            app.RZTree = uitree(app.SectorRestricZonePanel);
4707            app.RZTree.SelectionChangedFcn = createCallbackFcn(app,
                    @RZTreeSelectionChanged, true);
4708            app.RZTree.Position = [777 8 315 154];
4709
4710            % Create EliminarRZButton
4711            app.EliminarRZButton = uibutton(app.SectorRestricZonePanel, 'push');
4712            app.EliminarRZButton.ButtonPushedFcn = createCallbackFcn(app,
                    @EliminarRZButtonPushed, true);
4713            app.EliminarRZButton.BackgroundColor = [0 0.451 0.7412];
4714            app.EliminarRZButton.FontColor = [1 1 1];
4715            app.EliminarRZButton.Position = [679 78 84 22];
4716            app.EliminarRZButton.Text = 'Eliminar';
4717
4718            % Create RZArchPanel
4719            app.RZArchPanel = uipanel(app.SectorRestricZonePanel);
4720            app.RZArchPanel.Visible = 'off';
4721            app.RZArchPanel.BackgroundColor = [0.0196 0.0196 0.3804];
4722            app.RZArchPanel.Position = [160 8 507 154];
4723
4724            % Create XnEditFieldLabel_4
4725            app.XnEditFieldLabel_4 = uilabel(app.RZArchPanel);
4726            app.XnEditFieldLabel_4.HorizontalAlignment = 'right';
4727            app.XnEditFieldLabel_4.FontColor = [1 1 1];
4728            app.XnEditFieldLabel_4.Position = [38 88 25 19];
4729            app.XnEditFieldLabel_4.Text = 'Xn';
4730
4731            % Create RZXnarchEditField
```

```
4732            app.RZXnarchEditField = uieditfield(app.RZArchPanel, 'numeric');
4733            app.RZXnarchEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZXnarchEditFieldValueChanged, true);
4734            app.RZXnarchEditField.Position = [78 88 90 21];
4735
4736            % Create YnEditFieldLabel_4
4737            app.YnEditFieldLabel_4 = uilabel(app.RZArchPanel);
4738            app.YnEditFieldLabel_4.HorizontalAlignment = 'right';
4739            app.YnEditFieldLabel_4.FontColor = [1 1 1];
4740            app.YnEditFieldLabel_4.Position = [223 89 25 20];
4741            app.YnEditFieldLabel_4.Text = 'Yn';
4742
4743            % Create RZYnarchEditField
4744            app.RZYnarchEditField = uieditfield(app.RZArchPanel, 'numeric');
4745            app.RZYnarchEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZYnarchEditFieldValueChanged, true);
4746            app.RZYnarchEditField.Position = [263 88 90 21];
4747
4748            % Create Xn1EditFieldLabel_4
4749            app.Xn1EditFieldLabel_4 = uilabel(app.RZArchPanel);
4750            app.Xn1EditFieldLabel_4.HorizontalAlignment = 'right';
4751            app.Xn1EditFieldLabel_4.FontColor = [1 1 1];
4752            app.Xn1EditFieldLabel_4.Position = [29 60 34 17];
4753            app.Xn1EditFieldLabel_4.Text = 'Xn+1';
4754
4755            % Create RZXn1archEditField
4756            app.RZXn1archEditField = uieditfield(app.RZArchPanel, 'numeric');
4757            app.RZXn1archEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZXn1archEditFieldValueChanged, true);
4758            app.RZXn1archEditField.Position = [78 57 90 22];
4759
4760            % Create Yn1EditFieldLabel_4
4761            app.Yn1EditFieldLabel_4 = uilabel(app.RZArchPanel);
4762            app.Yn1EditFieldLabel_4.HorizontalAlignment = 'right';
4763            app.Yn1EditFieldLabel_4.FontColor = [1 1 1];
4764            app.Yn1EditFieldLabel_4.Position = [223 60 34 17];
4765            app.Yn1EditFieldLabel_4.Text = 'Yn+1';
4766
4767            % Create RZYn1archEditField
4768            app.RZYn1archEditField = uieditfield(app.RZArchPanel, 'numeric');
4769            app.RZYn1archEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZYn1archEditFieldValueChanged, true);
4770            app.RZYn1archEditField.Position = [263 60 90 19];
4771
4772            % Create ZonaRestringidaarcoLabel
4773            app.ZonaRestringidaarcoLabel = uilabel(app.RZArchPanel);
4774            app.ZonaRestringidaarcoLabel.BackgroundColor = [0 0.4471 0.7412];
4775            app.ZonaRestringidaarcoLabel.FontColor = [1 1 1];
4776            app.ZonaRestringidaarcoLabel.Position = [29 118 124 22];
4777            app.ZonaRestringidaarcoLabel.Text = 'Zona Restringida arco';
4778
4779            % Create MostrarRZAButton
4780            app.MostrarRZAButton = uibutton(app.RZArchPanel, 'push');
4781            app.MostrarRZAButton.ButtonPushedFcn = createCallbackFcn(app,
                    @MostrarRZAButtonPushed, true);
4782            app.MostrarRZAButton.BackgroundColor = [0 0.451 0.7412];
4783            app.MostrarRZAButton.FontColor = [1 1 1];
```

```
4784            app.MostrarRZAButton.Position = [394 97 100 22];
4785            app.MostrarRZAButton.Text = 'Mostrar';
4786
4787            % Create GuardarRZAButton
4788            app.GuardarRZAButton = uibutton(app.RZArchPanel, 'push');
4789            app.GuardarRZAButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarRZAButtonPushed, true);
4790            app.GuardarRZAButton.BackgroundColor = [0 0.451 0.7412];
4791            app.GuardarRZAButton.FontColor = [1 1 1];
4792            app.GuardarRZAButton.Position = [394 57 100 22];
4793            app.GuardarRZAButton.Text = 'Guardar';
4794
4795            % Create RadioLabel_2
4796            app.RadioLabel_2 = uilabel(app.RZArchPanel);
4797            app.RadioLabel_2.HorizontalAlignment = 'right';
4798            app.RadioLabel_2.FontColor = [1 1 1];
4799            app.RadioLabel_2.Position = [26 24 37 23];
4800            app.RadioLabel_2.Text = 'Radio';
4801
4802            % Create RZARadiusEditField
4803            app.RZARadiusEditField = uieditfield(app.RZArchPanel, 'numeric');
4804            app.RZARadiusEditField.Limits = [0 Inf];
4805            app.RZARadiusEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZARadiusEditFieldValueChanged, true);
4806            app.RZARadiusEditField.Position = [78 24 90 24];
4807
4808            % Create RZAConcaveConvexSwitch
4809            app.RZAConcaveConvexSwitch = uiswitch(app.RZArchPanel, 'slider');
4810            app.RZAConcaveConvexSwitch.Items = {'^', 'v'};
4811            app.RZAConcaveConvexSwitch.ValueChangedFcn = createCallbackFcn(app,
                    @RZAConcaveConvexSwitchValueChanged, true);
4812            app.RZAConcaveConvexSwitch.FontColor = [1 1 1];
4813            app.RZAConcaveConvexSwitch.Position = [209 24 52 23];
4814            app.RZAConcaveConvexSwitch.Value = '^';
4815
4816            % Create RZACortoLargoSwitch
4817            app.RZACortoLargoSwitch = uiswitch(app.RZArchPanel, 'slider');
4818            app.RZACortoLargoSwitch.Items = {'Corto', 'Largo'};
4819            app.RZACortoLargoSwitch.ValueChangedFcn = createCallbackFcn(app,
                    @RZACortoLargoSwitchValueChanged, true);
4820            app.RZACortoLargoSwitch.FontColor = [1 1 1];
4821            app.RZACortoLargoSwitch.Position = [326 24 54 24];
4822            app.RZACortoLargoSwitch.Value = 'Corto';
4823
4824            % Create RZAltitudePanel
4825            app.RZAltitudePanel = uipanel(app.SectorRestricZonePanel);
4826            app.RZAltitudePanel.Visible = 'off';
4827            app.RZAltitudePanel.BackgroundColor = [0.0196 0.0196 0.3804];
4828            app.RZAltitudePanel.Position = [160 8 507 154];
4829
4830            % Create LmiteInferiorLabel_2
4831            app.LmiteInferiorLabel_2 = uilabel(app.RZAltitudePanel);
4832            app.LmiteInferiorLabel_2.HorizontalAlignment = 'center';
4833            app.LmiteInferiorLabel_2.FontColor = [1 1 1];
4834            app.LmiteInferiorLabel_2.Position = [53 72 43 35];
4835            app.LmiteInferiorLabel_2.Text = {'Límite'; 'Inferior'};
4836
```

```
4837            % Create RZZinfEditField
4838            app.RZZinfEditField = uieditfield(app.RZAltitudePanel, 'numeric');
4839            app.RZZinfEditField.Limits = [0 Inf];
4840            app.RZZinfEditField.ValueDisplayFormat = '%.0f';
4841            app.RZZinfEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZZinfEditFieldValueChanged, true);
4842            app.RZZinfEditField.Position = [111 78 99 22];
4843
4844            % Create LmiteSuperiorLabel_2
4845            app.LmiteSuperiorLabel_2 = uilabel(app.RZAltitudePanel);
4846            app.LmiteSuperiorLabel_2.HorizontalAlignment = 'center';
4847            app.LmiteSuperiorLabel_2.FontColor = [1 1 1];
4848            app.LmiteSuperiorLabel_2.Position = [53 34 51 33];
4849            app.LmiteSuperiorLabel_2.Text = {'Límite'; 'Superior'};
4850
4851            % Create RZZsupEditField
4852            app.RZZsupEditField = uieditfield(app.RZAltitudePanel, 'numeric');
4853            app.RZZsupEditField.Limits = [0 Inf];
4854            app.RZZsupEditField.ValueDisplayFormat = '%.0f';
4855            app.RZZsupEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RZZsupEditFieldValueChanged, true);
4856            app.RZZsupEditField.Position = [111 40 99 22];
4857
4858            % Create LmitesverticalesLabelRZ
4859            app.LmitesverticalesLabelRZ = uilabel(app.RZAltitudePanel);
4860            app.LmitesverticalesLabelRZ.BackgroundColor = [0 0.4471 0.7412];
4861            app.LmitesverticalesLabelRZ.FontColor = [1 1 1];
4862            app.LmitesverticalesLabelRZ.Position = [29 118 98 22];
4863            app.LmitesverticalesLabelRZ.Text = 'Límites verticales';
4864
4865            % Create GuardarRZAltButton
4866            app.GuardarRZAltButton = uibutton(app.RZAltitudePanel, 'push');
4867            app.GuardarRZAltButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarRZAltButtonPushed, true);
4868            app.GuardarRZAltButton.BackgroundColor = [0 0.451 0.7412];
4869            app.GuardarRZAltButton.FontColor = [1 1 1];
4870            app.GuardarRZAltButton.Position = [352 70 108 22];
4871            app.GuardarRZAltButton.Text = 'Guardar';
4872
4873            % Create FLftinfDropDownRZ
4874            app.FLftinfDropDownRZ = uidropdown(app.RZAltitudePanel);
4875            app.FLftinfDropDownRZ.Items = {'FL', 'ft'};
4876            app.FLftinfDropDownRZ.ItemsData = {'1', '2'};
4877            app.FLftinfDropDownRZ.Position = [237 78 48 22];
4878            app.FLftinfDropDownRZ.Value = '1';
4879
4880            % Create FLftsupDropDownRZ
4881            app.FLftsupDropDownRZ = uidropdown(app.RZAltitudePanel);
4882            app.FLftsupDropDownRZ.Items = {'FL', 'ft'};
4883            app.FLftsupDropDownRZ.ItemsData = {'1', '2'};
4884            app.FLftsupDropDownRZ.Position = [237 40 48 22];
4885            app.FLftsupDropDownRZ.Value = '1';
4886
4887            % Create NuevaZonaRestringidaButton
4888            app.NuevaZonaRestringidaButton = uibutton(app.SectorRestricZonePanel
                    , 'push');
```

```
4889            app.NuevaZonaRestringidaButton.ButtonPushedFcn = createCallbackFcn(
                    app, @NuevaZonaRestringidaButtonPushed, true);
4890            app.NuevaZonaRestringidaButton.BackgroundColor = [0 0.451 0.7412];
4891            app.NuevaZonaRestringidaButton.FontColor = [1 1 1];
4892            app.NuevaZonaRestringidaButton.Position = [679 28 84 36];
4893            app.NuevaZonaRestringidaButton.Text = {'Nueva Zona'; 'Restringida'};
4894
4895            % Create RZFLsupEditField
4896            app.RZFLsupEditField = uieditfield(app.EditordeEscenarios, 'text');
4897            app.RZFLsupEditField.Editable = 'off';
4898            app.RZFLsupEditField.HorizontalAlignment = 'right';
4899            app.RZFLsupEditField.FontSize = 14;
4900            app.RZFLsupEditField.FontColor = [1 0 0];
4901            app.RZFLsupEditField.BackgroundColor = [0 0 0];
4902            app.RZFLsupEditField.Position = [651 585 108 25];
4903
4904            % Create RZFLinfEditField
4905            app.RZFLinfEditField = uieditfield(app.EditordeEscenarios, 'text');
4906            app.RZFLinfEditField.Editable = 'off';
4907            app.RZFLinfEditField.HorizontalAlignment = 'right';
4908            app.RZFLinfEditField.FontSize = 14;
4909            app.RZFLinfEditField.FontColor = [1 0 0];
4910            app.RZFLinfEditField.BackgroundColor = [0 0 0];
4911            app.RZFLinfEditField.Position = [651 560 108 25];
4912
4913            % Create SectorAirportPanel
4914            app.SectorAirportPanel = uipanel(app.EditordeEscenarios);
4915            app.SectorAirportPanel.Visible = 'off';
4916            app.SectorAirportPanel.BackgroundColor = [0.0196 0.0196 0.3804];
4917            app.SectorAirportPanel.Position = [50 25 1100 170];
4918
4919            % Create XLabel
4920            app.XLabel = uilabel(app.SectorAirportPanel);
4921            app.XLabel.HorizontalAlignment = 'right';
4922            app.XLabel.FontColor = [1 1 1];
4923            app.XLabel.Position = [163 80 25 22];
4924            app.XLabel.Text = 'X';
4925
4926            % Create XairportEditField
4927            app.XairportEditField = uieditfield(app.SectorAirportPanel, 'numeric
                    ');
4928            app.XairportEditField.ValueChangedFcn = createCallbackFcn(app,
                    @XairportEditFieldValueChanged, true);
4929            app.XairportEditField.Position = [200 80 90 22];
4930
4931            % Create YLabel
4932            app.YLabel = uilabel(app.SectorAirportPanel);
4933            app.YLabel.HorizontalAlignment = 'right';
4934            app.YLabel.FontColor = [1 1 1];
4935            app.YLabel.Position = [366 80 25 22];
4936            app.YLabel.Text = 'Y';
4937
4938            % Create YairportEditField
4939            app.YairportEditField = uieditfield(app.SectorAirportPanel, 'numeric
                    ');
4940            app.YairportEditField.ValueChangedFcn = createCallbackFcn(app,
                    @YairportEditFieldValueChanged, true);
```

```
4941            app.YairportEditField.Position = [400 80 90 22];
4942
4943            % Create AeropuertosdelSectorLabel
4944            app.AeropuertosdelSectorLabel = uilabel(app.SectorAirportPanel);
4945            app.AeropuertosdelSectorLabel.BackgroundColor = [0 0.4471 0.7412];
4946            app.AeropuertosdelSectorLabel.FontColor = [1 1 1];
4947            app.AeropuertosdelSectorLabel.Position = [20 130 128 22];
4948            app.AeropuertosdelSectorLabel.Text = 'Aeropuertos del Sector';
4949
4950            % Create NombreEditFieldLabel
4951            app.NombreEditFieldLabel = uilabel(app.SectorAirportPanel);
4952            app.NombreEditFieldLabel.HorizontalAlignment = 'right';
4953            app.NombreEditFieldLabel.FontColor = [1 1 1];
4954            app.NombreEditFieldLabel.Position = [238 130 48 22];
4955            app.NombreEditFieldLabel.Text = 'Nombre';
4956
4957            % Create NombreAirportEditField
4958            app.NombreAirportEditField = uieditfield(app.SectorAirportPanel, '
                    text');
4959            app.NombreAirportEditField.ValueChangedFcn = createCallbackFcn(app,
                    @NombreAirportEditFieldValueChanged, true);
4960            app.NombreAirportEditField.Tooltip = {'5 letras en'; 'mayúsculas'; '
                    que sean leíbles'};
4961            app.NombreAirportEditField.Position = [300 130 100 22];
4962
4963            % Create AirportTree
4964            app.AirportTree = uitree(app.SectorAirportPanel);
4965            app.AirportTree.SelectionChangedFcn = createCallbackFcn(app,
                    @AirportTreeSelectionChanged, true);
4966            app.AirportTree.Position = [777 8 315 154];
4967
4968            % Create GuardarAirportButton
4969            app.GuardarAirportButton = uibutton(app.SectorAirportPanel, 'push');
4970            app.GuardarAirportButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarAirportButtonPushed, true);
4971            app.GuardarAirportButton.BackgroundColor = [0 0.451 0.7412];
4972            app.GuardarAirportButton.FontColor = [1 1 1];
4973            app.GuardarAirportButton.Position = [195 35 100 22];
4974            app.GuardarAirportButton.Text = 'Guardar';
4975
4976            % Create AddRWYButton
4977            app.AddRWYButton = uibutton(app.SectorAirportPanel, 'push');
4978            app.AddRWYButton.ButtonPushedFcn = createCallbackFcn(app,
                    @AddRWYButtonPushed, true);
4979            app.AddRWYButton.BackgroundColor = [0 0.451 0.7412];
4980            app.AddRWYButton.FontColor = [1 1 1];
4981            app.AddRWYButton.Position = [395 35 100 22];
4982            app.AddRWYButton.Text = 'Añadir Pista';
4983
4984            % Create EliminarAirportButton
4985            app.EliminarAirportButton = uibutton(app.SectorAirportPanel, 'push')
                    ;
4986            app.EliminarAirportButton.ButtonPushedFcn = createCallbackFcn(app,
                    @EliminarAirportButtonPushed, true);
4987            app.EliminarAirportButton.BackgroundColor = [0 0.451 0.7412];
4988            app.EliminarAirportButton.FontColor = [1 1 1];
4989            app.EliminarAirportButton.Position = [295 35 100 22];
```

```
4990            app.EliminarAirportButton.Text = 'Eliminar';
4991
4992            % Create RWYPanel
4993            app.RWYPanel = uipanel(app.SectorAirportPanel);
4994            app.RWYPanel.Visible = 'off';
4995            app.RWYPanel.BackgroundColor = [0.0196 0.0196 0.3804];
4996            app.RWYPanel.Position = [505 5 245 159];
4997
4998            % Create PistadeaterrizajeLabel
4999            app.PistadeaterrizajeLabel = uilabel(app.RWYPanel);
5000            app.PistadeaterrizajeLabel.BackgroundColor = [0 0.4471 0.7412];
5001            app.PistadeaterrizajeLabel.FontColor = [1 1 1];
5002            app.PistadeaterrizajeLabel.Position = [8 127 102 22];
5003            app.PistadeaterrizajeLabel.Text = 'Pista de aterrizaje';
5004
5005            % Create RWYNewEditSwitch
5006            app.RWYNewEditSwitch = uiswitch(app.RWYPanel, 'slider');
5007            app.RWYNewEditSwitch.Items = {'Nueva', 'Editar'};
5008            app.RWYNewEditSwitch.Orientation = 'vertical';
5009            app.RWYNewEditSwitch.ValueChangedFcn = createCallbackFcn(app,
                    @RWYNewEditSwitchValueChanged, true);
5010            app.RWYNewEditSwitch.FontColor = [1 1 1];
5011            app.RWYNewEditSwitch.Position = [20 40 20 45];
5012            app.RWYNewEditSwitch.Value = 'Nueva';
5013
5014            % Create NewRWYPanel
5015            app.NewRWYPanel = uipanel(app.RWYPanel);
5016            app.NewRWYPanel.BorderType = 'none';
5017            app.NewRWYPanel.Visible = 'off';
5018            app.NewRWYPanel.BackgroundColor = [0.0196 0.0196 0.3804];
5019            app.NewRWYPanel.Position = [59 1 185 122];
5020
5021            % Create RWYSpinnerLabel
5022            app.RWYSpinnerLabel = uilabel(app.NewRWYPanel);
5023            app.RWYSpinnerLabel.HorizontalAlignment = 'right';
5024            app.RWYSpinnerLabel.FontColor = [1 1 1];
5025            app.RWYSpinnerLabel.Position = [51 82 33 22];
5026            app.RWYSpinnerLabel.Text = 'RWY';
5027
5028            % Create NewRWYSpinner
5029            app.NewRWYSpinner = uispinner(app.NewRWYPanel);
5030            app.NewRWYSpinner.Limits = [0 36];
5031            app.NewRWYSpinner.ValueChangedFcn = createCallbackFcn(app,
                    @NewRWYSpinnerValueChanged, true);
5032            app.NewRWYSpinner.Position = [95 82 60 22];
5033
5034            % Create GuardarNewRWYButton
5035            app.GuardarNewRWYButton = uibutton(app.NewRWYPanel, 'push');
5036            app.GuardarNewRWYButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarNewRWYButtonPushed, true);
5037            app.GuardarNewRWYButton.BackgroundColor = [0 0.451 0.7412];
5038            app.GuardarNewRWYButton.FontColor = [1 1 1];
5039            app.GuardarNewRWYButton.Position = [51 50 112 21];
5040            app.GuardarNewRWYButton.Text = 'Guardar';
5041
5042            % Create EditRWYPanel
5043            app.EditRWYPanel = uipanel(app.RWYPanel);
```

```
5044            app.EditRWYPanel.BorderType = 'none';
5045            app.EditRWYPanel.Visible = 'off';
5046            app.EditRWYPanel.BackgroundColor = [0.0196 0.0196 0.3804];
5047            app.EditRWYPanel.Position = [59 1 185 122];
5048
5049            % Create RWYDropDownLabel
5050            app.RWYDropDownLabel = uilabel(app.EditRWYPanel);
5051            app.RWYDropDownLabel.HorizontalAlignment = 'right';
5052            app.RWYDropDownLabel.FontColor = [1 1 1];
5053            app.RWYDropDownLabel.Position = [38 89 33 22];
5054            app.RWYDropDownLabel.Text = 'RWY';
5055
5056            % Create RWYEditDropDown
5057            app.RWYEditDropDown = uidropdown(app.EditRWYPanel);
5058            app.RWYEditDropDown.Items = {'--'};
5059            app.RWYEditDropDown.ItemsData = {'0'};
5060            app.RWYEditDropDown.ValueChangedFcn = createCallbackFcn(app,
                    @RWYEditDropDownValueChanged, true);
5061            app.RWYEditDropDown.Position = [83 93 82 20];
5062            app.RWYEditDropDown.Value = '0';
5063
5064            % Create RWYEditSpinner
5065            app.RWYEditSpinner = uispinner(app.EditRWYPanel);
5066            app.RWYEditSpinner.Limits = [0 36];
5067            app.RWYEditSpinner.ValueChangedFcn = createCallbackFcn(app,
                    @RWYEditSpinnerValueChanged, true);
5068            app.RWYEditSpinner.Position = [90 64 75 22];
5069
5070            % Create GuardarRWYEditButton
5071            app.GuardarRWYEditButton = uibutton(app.EditRWYPanel, 'push');
5072            app.GuardarRWYEditButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarRWYEditButtonPushed, true);
5073            app.GuardarRWYEditButton.BackgroundColor = [0 0.451 0.7412];
5074            app.GuardarRWYEditButton.FontColor = [1 1 1];
5075            app.GuardarRWYEditButton.Position = [51 40 112 21];
5076            app.GuardarRWYEditButton.Text = 'Guardar';
5077
5078            % Create EliminarRWYEditButton
5079            app.EliminarRWYEditButton = uibutton(app.EditRWYPanel, 'push');
5080            app.EliminarRWYEditButton.ButtonPushedFcn = createCallbackFcn(app,
                    @EliminarRWYEditButtonPushed, true);
5081            app.EliminarRWYEditButton.BackgroundColor = [0 0.451 0.7412];
5082            app.EliminarRWYEditButton.FontColor = [1 1 1];
5083            app.EliminarRWYEditButton.Position = [51 14 112 22];
5084            app.EliminarRWYEditButton.Text = 'Eliminar';
5085
5086            % Create SectorWaypointPanel
5087            app.SectorWaypointPanel = uipanel(app.EditordeEscenarios);
5088            app.SectorWaypointPanel.Visible = 'off';
5089            app.SectorWaypointPanel.BackgroundColor = [0.0196 0.0196 0.3804];
5090            app.SectorWaypointPanel.Position = [50 25 1100 170];
5091
5092            % Create XLabel_2
5093            app.XLabel_2 = uilabel(app.SectorWaypointPanel);
5094            app.XLabel_2.HorizontalAlignment = 'right';
5095            app.XLabel_2.FontColor = [1 1 1];
5096            app.XLabel_2.Position = [163 80 25 22];
```

```
5097            app.XLabel_2.Text = 'X';
5098
5099            % Create XwaypointEditField
5100            app.XwaypointEditField = uieditfield(app.SectorWaypointPanel, '
                    numeric');
5101            app.XwaypointEditField.ValueChangedFcn = createCallbackFcn(app,
                    @XwaypointEditFieldValueChanged, true);
5102            app.XwaypointEditField.Position = [200 80 90 22];
5103
5104            % Create YLabel_2
5105            app.YLabel_2 = uilabel(app.SectorWaypointPanel);
5106            app.YLabel_2.HorizontalAlignment = 'right';
5107            app.YLabel_2.FontColor = [1 1 1];
5108            app.YLabel_2.Position = [366 80 25 22];
5109            app.YLabel_2.Text = 'Y';
5110
5111            % Create YwaypointEditField
5112            app.YwaypointEditField = uieditfield(app.SectorWaypointPanel, '
                    numeric');
5113            app.YwaypointEditField.ValueChangedFcn = createCallbackFcn(app,
                    @YwaypointEditFieldValueChanged, true);
5114            app.YwaypointEditField.Position = [400 80 90 22];
5115
5116            % Create PuntosdePasodelSectorLabel
5117            app.PuntosdePasodelSectorLabel = uilabel(app.SectorWaypointPanel);
5118            app.PuntosdePasodelSectorLabel.BackgroundColor = [0 0.4471 0.7412];
5119            app.PuntosdePasodelSectorLabel.FontColor = [1 1 1];
5120            app.PuntosdePasodelSectorLabel.Position = [20 130 148 22];
5121            app.PuntosdePasodelSectorLabel.Text = 'Puntos de Paso del Sector';
5122
5123            % Create NombreEditField_2Label
5124            app.NombreEditField_2Label = uilabel(app.SectorWaypointPanel);
5125            app.NombreEditField_2Label.HorizontalAlignment = 'right';
5126            app.NombreEditField_2Label.FontColor = [1 1 1];
5127            app.NombreEditField_2Label.Position = [238 130 48 22];
5128            app.NombreEditField_2Label.Text = 'Nombre';
5129
5130            % Create NombreWPEditField
5131            app.NombreWPEditField = uieditfield(app.SectorWaypointPanel, 'text')
                    ;
5132            app.NombreWPEditField.ValueChangedFcn = createCallbackFcn(app,
                    @NombreWPEditFieldValueChanged, true);
5133            app.NombreWPEditField.Tooltip = {'5 letras en'; 'mayúsculas'; 'que
                    sean leíbles'};
5134            app.NombreWPEditField.Position = [300 130 100 22];
5135
5136            % Create WaypointTree
5137            app.WaypointTree = uitree(app.SectorWaypointPanel);
5138            app.WaypointTree.SelectionChangedFcn = createCallbackFcn(app,
                    @WaypointTreeSelectionChanged, true);
5139            app.WaypointTree.Position = [777 8 315 154];
5140
5141            % Create GuardarWaypointButton
5142            app.GuardarWaypointButton = uibutton(app.SectorWaypointPanel, 'push'
                    );
5143            app.GuardarWaypointButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarWaypointButtonPushed, true);
```

```
5144            app.GuardarWaypointButton.BackgroundColor = [0 0.451 0.7412];
5145            app.GuardarWaypointButton.FontColor = [1 1 1];
5146            app.GuardarWaypointButton.Position = [195 35 100 22];
5147            app.GuardarWaypointButton.Text = 'Guardar';
5148
5149            % Create EliminarWaypointButton
5150            app.EliminarWaypointButton = uibutton(app.SectorWaypointPanel, 'push
                    ');
5151            app.EliminarWaypointButton.ButtonPushedFcn = createCallbackFcn(app,
                    @EliminarWaypointButtonPushed, true);
5152            app.EliminarWaypointButton.BackgroundColor = [0 0.451 0.7412];
5153            app.EliminarWaypointButton.FontColor = [1 1 1];
5154            app.EliminarWaypointButton.Position = [295 35 100 22];
5155            app.EliminarWaypointButton.Text = 'Eliminar';
5156
5157            % Create WaypointFronterizoButton
5158            app.WaypointFronterizoButton = uibutton(app.SectorWaypointPanel, '
                    push');
5159            app.WaypointFronterizoButton.ButtonPushedFcn = createCallbackFcn(app
                    , @WaypointFronterizoButtonPushed, true);
5160            app.WaypointFronterizoButton.BackgroundColor = [0 0.451 0.7412];
5161            app.WaypointFronterizoButton.FontColor = [1 1 1];
5162            app.WaypointFronterizoButton.Position = [395 35 100 22];
5163            app.WaypointFronterizoButton.Text = 'Fronterizo';
5164
5165            % Create WaypointFronterizoFreqPanel
5166            app.WaypointFronterizoFreqPanel = uipanel(app.SectorWaypointPanel);
5167            app.WaypointFronterizoFreqPanel.Visible = 'off';
5168            app.WaypointFronterizoFreqPanel.BackgroundColor = [0.0196 0.0196
                    0.3804];
5169            app.WaypointFronterizoFreqPanel.Position = [505 5 232 159];
5170
5171            % Create FrecuenciadelSectorFronterizoLabel
5172            app.FrecuenciadelSectorFronterizoLabel = uilabel(app.
                    WaypointFronterizoFreqPanel);
5173            app.FrecuenciadelSectorFronterizoLabel.BackgroundColor = [0 0.4471
                    0.7412];
5174            app.FrecuenciadelSectorFronterizoLabel.FontColor = [1 1 1];
5175            app.FrecuenciadelSectorFronterizoLabel.Position = [8 127 180 22];
5176            app.FrecuenciadelSectorFronterizoLabel.Text = 'Frecuencia del Sector
                     Fronterizo';
5177
5178            % Create FreqMHzEditFieldLabel
5179            app.FreqMHzEditFieldLabel = uilabel(app.WaypointFronterizoFreqPanel)
                    ;
5180            app.FreqMHzEditFieldLabel.HorizontalAlignment = 'right';
5181            app.FreqMHzEditFieldLabel.FontColor = [1 1 1];
5182            app.FreqMHzEditFieldLabel.Position = [20 91 69 22];
5183            app.FreqMHzEditFieldLabel.Text = 'Freq. (MHz)';
5184
5185            % Create FreqMHzEditField
5186            app.FreqMHzEditField = uieditfield(app.WaypointFronterizoFreqPanel,
                    'numeric');
5187            app.FreqMHzEditField.Limits = [0 Inf];
5188            app.FreqMHzEditField.ValueDisplayFormat = '%.2f';
5189            app.FreqMHzEditField.ValueChangedFcn = createCallbackFcn(app,
                    @FreqMHzEditFieldValueChanged, true);
```

```
5190            app.FreqMHzEditField.Position = [104 91 100 22];
5191
5192            % Create GuardarWPFreqButton
5193            app.GuardarWPFreqButton = uibutton(app.WaypointFronterizoFreqPanel,
                    'push');
5194            app.GuardarWPFreqButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarWPFreqButtonPushed, true);
5195            app.GuardarWPFreqButton.BackgroundColor = [0 0.451 0.7412];
5196            app.GuardarWPFreqButton.FontColor = [1 1 1];
5197            app.GuardarWPFreqButton.Position = [65 50 100 22];
5198            app.GuardarWPFreqButton.Text = 'Guardar';
5199
5200            % Create SectorAirplanePanel
5201            app.SectorAirplanePanel = uipanel(app.EditordeEscenarios);
5202            app.SectorAirplanePanel.Visible = 'off';
5203            app.SectorAirplanePanel.BackgroundColor = [0.0196 0.0196 0.3804];
5204            app.SectorAirplanePanel.Position = [50 25 1100 170];
5205
5206            % Create AirplaneTabGroup
5207            app.AirplaneTabGroup = uitabgroup(app.SectorAirplanePanel);
5208            app.AirplaneTabGroup.Position = [20 8 650 154];
5209
5210            % Create IGTab
5211            app.IGTab = uitab(app.AirplaneTabGroup);
5212            app.IGTab.Title = 'Información General';
5213            app.IGTab.BackgroundColor = [0.0196 0.0196 0.3804];
5214
5215            % Create NiveldeVueloEditFieldLabel
5216            app.NiveldeVueloEditFieldLabel = uilabel(app.IGTab);
5217            app.NiveldeVueloEditFieldLabel.HorizontalAlignment = 'right';
5218            app.NiveldeVueloEditFieldLabel.FontColor = [1 1 1];
5219            app.NiveldeVueloEditFieldLabel.Position = [409 82 89 22];
5220            app.NiveldeVueloEditFieldLabel.Text = 'Nivel de Vuelo';
5221
5222            % Create NiveldeVueloEditField
5223            app.NiveldeVueloEditField = uieditfield(app.IGTab, 'numeric');
5224            app.NiveldeVueloEditField.Limits = [0 Inf];
5225            app.NiveldeVueloEditField.ValueChangedFcn = createCallbackFcn(app,
                    @NiveldeVueloEditFieldValueChanged, true);
5226            app.NiveldeVueloEditField.Tooltip = {'El FL debe estar'; 'dentro del
                     espacio'; 'del Sector'};
5227            app.NiveldeVueloEditField.Position = [514 82 67 22];
5228
5229            % Create MatrculadelaaeronaveEditFieldLabel
5230            app.MatrculadelaaeronaveEditFieldLabel = uilabel(app.IGTab);
5231            app.MatrculadelaaeronaveEditFieldLabel.HorizontalAlignment = 'right'
                    ;
5232            app.MatrculadelaaeronaveEditFieldLabel.FontColor = [1 1 1];
5233            app.MatrculadelaaeronaveEditFieldLabel.Position = [22 82 138 22];
5234            app.MatrculadelaaeronaveEditFieldLabel.Text = 'Matrícula de la
                    aeronave';
5235
5236            % Create MatrculadelaaeronaveEditField
5237            app.MatrculadelaaeronaveEditField = uieditfield(app.IGTab, 'text');
5238            app.MatrculadelaaeronaveEditField.ValueChangedFcn =
                    createCallbackFcn(app,
                    @MatrculadelaaeronaveEditFieldValueChanged, true);
```

```
5239            app.MatrculadelaaeronaveEditField.Tooltip = {'Formato'; 'EC-AAA
                    hasta'; 'EC-WZZ'};
5240            app.MatrculadelaaeronaveEditField.Position = [175 82 92 22];
5241
5242            % Create VelocidaddeVueloktEditFieldLabel
5243            app.VelocidaddeVueloktEditFieldLabel = uilabel(app.IGTab);
5244            app.VelocidaddeVueloktEditFieldLabel.HorizontalAlignment = 'right';
5245            app.VelocidaddeVueloktEditFieldLabel.FontColor = [1 1 1];
5246            app.VelocidaddeVueloktEditFieldLabel.Position = [31 32 128 22];
5247            app.VelocidaddeVueloktEditFieldLabel.Text = 'Velocidad de Vuelo (kt)
                    ';
5248
5249            % Create VelocidaddeVueloktEditField
5250            app.VelocidaddeVueloktEditField = uieditfield(app.IGTab, 'numeric');
5251            app.VelocidaddeVueloktEditField.Limits = [0 Inf];
5252            app.VelocidaddeVueloktEditField.ValueChangedFcn = createCallbackFcn(
                    app, @VelocidaddeVueloktEditFieldValueChanged, true);
5253            app.VelocidaddeVueloktEditField.Position = [174 32 93 22];
5254
5255            % Create ETOEditFieldLabel
5256            app.ETOEditFieldLabel = uilabel(app.IGTab);
5257            app.ETOEditFieldLabel.HorizontalAlignment = 'right';
5258            app.ETOEditFieldLabel.FontColor = [1 1 1];
5259            app.ETOEditFieldLabel.Tooltip = {'Estimated '; 'Time Over'; 'Tiempo
                    Estimado'; 'de Entrada'};
5260            app.ETOEditFieldLabel.Position = [470 32 30 22];
5261            app.ETOEditFieldLabel.Text = 'ETO';
5262
5263            % Create ETOEditField
5264            app.ETOEditField = uieditfield(app.IGTab, 'text');
5265            app.ETOEditField.ValueChangedFcn = createCallbackFcn(app,
                    @ETOEditFieldValueChanged, true);
5266            app.ETOEditField.HorizontalAlignment = 'right';
5267            app.ETOEditField.Position = [515 32 66 22];
5268            app.ETOEditField.Value = '00:00';
5269
5270            % Create RumbodeEntradaEditFieldLabel
5271            app.RumbodeEntradaEditFieldLabel = uilabel(app.IGTab);
5272            app.RumbodeEntradaEditFieldLabel.HorizontalAlignment = 'right';
5273            app.RumbodeEntradaEditFieldLabel.FontColor = [1 1 1];
5274            app.RumbodeEntradaEditFieldLabel.Position = [275 32 106 22];
5275            app.RumbodeEntradaEditFieldLabel.Text = 'Rumbo de Entrada';
5276
5277            % Create EHDGEditField
5278            app.EHDGEditField = uieditfield(app.IGTab, 'numeric');
5279            app.EHDGEditField.Limits = [0 359];
5280            app.EHDGEditField.ValueChangedFcn = createCallbackFcn(app,
                    @EHDGEditFieldValueChanged, true);
5281            app.EHDGEditField.Position = [396 32 71 22];
5282
5283            % Create RouteTab
5284            app.RouteTab = uitab(app.AirplaneTabGroup);
5285            app.RouteTab.Title = 'Ruta';
5286            app.RouteTab.BackgroundColor = [0.0196 0.0196 0.3804];
5287
5288            % Create RouteTree
5289            app.RouteTree = uitree(app.RouteTab);
```

```
5290            app.RouteTree.SelectionChangedFcn = createCallbackFcn(app,
                    @RouteTreeSelectionChanged, true);
5291            app.RouteTree.Position = [396 11 238 108];
5292
5293            % Create PuntodePasodelaRutaEditFieldLabel
5294            app.PuntodePasodelaRutaEditFieldLabel = uilabel(app.RouteTab);
5295            app.PuntodePasodelaRutaEditFieldLabel.HorizontalAlignment = 'right';
5296            app.PuntodePasodelaRutaEditFieldLabel.FontColor = [1 1 1];
5297            app.PuntodePasodelaRutaEditFieldLabel.Position = [4 87 84 28];
5298            app.PuntodePasodelaRutaEditFieldLabel.Text = {'Punto de Paso'; 'de
                    la Ruta'};
5299
5300            % Create RouteWPEditField
5301            app.RouteWPEditField = uieditfield(app.RouteTab, 'text');
5302            app.RouteWPEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RouteWPEditFieldValueChanged, true);
5303            app.RouteWPEditField.Position = [103 93 100 22];
5304
5305            % Create GuardarRouteWPButton
5306            app.GuardarRouteWPButton = uibutton(app.RouteTab, 'push');
5307            app.GuardarRouteWPButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarRouteWPButtonPushed, true);
5308            app.GuardarRouteWPButton.BackgroundColor = [0 0.451 0.7412];
5309            app.GuardarRouteWPButton.FontColor = [1 1 1];
5310            app.GuardarRouteWPButton.Position = [250 90 92 22];
5311            app.GuardarRouteWPButton.Text = 'Guardar';
5312
5313            % Create EliminarRouteWPButton
5314            app.EliminarRouteWPButton = uibutton(app.RouteTab, 'push');
5315            app.EliminarRouteWPButton.ButtonPushedFcn = createCallbackFcn(app,
                    @EliminarRouteWPButtonPushed, true);
5316            app.EliminarRouteWPButton.BackgroundColor = [0 0.451 0.7412];
5317            app.EliminarRouteWPButton.FontColor = [1 1 1];
5318            app.EliminarRouteWPButton.Position = [250 40 92 22];
5319            app.EliminarRouteWPButton.Text = 'Eliminar';
5320
5321            % Create IncidenciasTab
5322            app.IncidenciasTab = uitab(app.AirplaneTabGroup);
5323            app.IncidenciasTab.Title = 'Incidencias';
5324            app.IncidenciasTab.BackgroundColor = [0.0196 0.0196 0.3804];
5325
5326            % Create TipodeIncidenciaDropDownLabel
5327            app.TipodeIncidenciaDropDownLabel = uilabel(app.IncidenciasTab);
5328            app.TipodeIncidenciaDropDownLabel.HorizontalAlignment = 'center';
5329            app.TipodeIncidenciaDropDownLabel.FontColor = [1 1 1];
5330            app.TipodeIncidenciaDropDownLabel.Position = [50 93 111 22];
5331            app.TipodeIncidenciaDropDownLabel.Text = 'Tipo de Incidencia';
5332
5333            % Create TipodeIncidenciaDropDown
5334            app.TipodeIncidenciaDropDown = uidropdown(app.IncidenciasTab);
5335            app.TipodeIncidenciaDropDown.Items = {'--/--', 'Cambio FL', '
                    Solicitar Directo', 'Desvío No Programado'};
5336            app.TipodeIncidenciaDropDown.ItemsData = {'0', '1', '2', '3'};
5337            app.TipodeIncidenciaDropDown.ValueChangedFcn = createCallbackFcn(app
                    , @TipodeIncidenciaDropDownValueChanged, true);
5338            app.TipodeIncidenciaDropDown.Position = [13 55 185 27];
5339            app.TipodeIncidenciaDropDown.Value = '0';
```

```
5340
5341            % Create CambioFLPanel
5342            app.CambioFLPanel = uipanel(app.IncidenciasTab);
5343            app.CambioFLPanel.BorderType = 'none';
5344            app.CambioFLPanel.Visible = 'off';
5345            app.CambioFLPanel.BackgroundColor = [0.0196 0.0196 0.3804];
5346            app.CambioFLPanel.Position = [218 1 429 128];
5347
5348            % Create PuntodePasoTriggerEditFieldLabel
5349            app.PuntodePasoTriggerEditFieldLabel = uilabel(app.CambioFLPanel);
5350            app.PuntodePasoTriggerEditFieldLabel.HorizontalAlignment = 'center';
5351            app.PuntodePasoTriggerEditFieldLabel.FontColor = [1 1 1];
5352            app.PuntodePasoTriggerEditFieldLabel.Position = [18 83 84 28];
5353            app.PuntodePasoTriggerEditFieldLabel.Text = {'Punto de Paso'; '
                    Trigger'};
5354
5355            % Create WaypointTriggerCFLEditField
5356            app.WaypointTriggerCFLEditField = uieditfield(app.CambioFLPanel, '
                    text');
5357            app.WaypointTriggerCFLEditField.ValueChangedFcn = createCallbackFcn(
                    app, @WaypointTriggerCFLEditFieldValueChanged, true);
5358            app.WaypointTriggerCFLEditField.Position = [117 89 76 22];
5359
5360            % Create FLObjetivoEditFieldLabel
5361            app.FLObjetivoEditFieldLabel = uilabel(app.CambioFLPanel);
5362            app.FLObjetivoEditFieldLabel.HorizontalAlignment = 'right';
5363            app.FLObjetivoEditFieldLabel.FontColor = [1 1 1];
5364            app.FLObjetivoEditFieldLabel.Position = [249 89 67 22];
5365            app.FLObjetivoEditFieldLabel.Text = 'FL Objetivo';
5366
5367            % Create FLObjetivoEditField
5368            app.FLObjetivoEditField = uieditfield(app.CambioFLPanel, 'numeric');
5369            app.FLObjetivoEditField.ValueChangedFcn = createCallbackFcn(app,
                    @FLObjetivoEditFieldValueChanged, true);
5370            app.FLObjetivoEditField.Position = [331 89 79 22];
5371
5372            % Create TiempodeAlcanceDropDownLabel
5373            app.TiempodeAlcanceDropDownLabel = uilabel(app.CambioFLPanel);
5374            app.TiempodeAlcanceDropDownLabel.HorizontalAlignment = 'right';
5375            app.TiempodeAlcanceDropDownLabel.FontColor = [1 1 1];
5376            app.TiempodeAlcanceDropDownLabel.Position = [53 36 108 22];
5377            app.TiempodeAlcanceDropDownLabel.Text = 'Tiempo de Alcance';
5378
5379            % Create CFLDropDown
5380            app.CFLDropDown = uidropdown(app.CambioFLPanel);
5381            app.CFLDropDown.Items = {'Antes', 'Después'};
5382            app.CFLDropDown.ItemsData = {'1', '2'};
5383            app.CFLDropDown.ValueChangedFcn = createCallbackFcn(app,
                    @CFLDropDownValueChanged, true);
5384            app.CFLDropDown.Position = [176 36 100 22];
5385            app.CFLDropDown.Value = '1';
5386
5387            % Create GuardarIncidenciaButton
5388            app.GuardarIncidenciaButton = uibutton(app.IncidenciasTab, 'push');
5389            app.GuardarIncidenciaButton.ButtonPushedFcn = createCallbackFcn(app,
                     @GuardarIncidenciaButtonPushed, true);
5390            app.GuardarIncidenciaButton.BackgroundColor = [0 0.451 0.7412];
```

```
5391            app.GuardarIncidenciaButton.FontColor = [1 1 1];
5392            app.GuardarIncidenciaButton.Position = [22 20 80 22];
5393            app.GuardarIncidenciaButton.Text = 'Guardar';
5394
5395            % Create SolicitarDirectoPanel
5396            app.SolicitarDirectoPanel = uipanel(app.IncidenciasTab);
5397            app.SolicitarDirectoPanel.BorderType = 'none';
5398            app.SolicitarDirectoPanel.Visible = 'off';
5399            app.SolicitarDirectoPanel.BackgroundColor = [0.0196 0.0196 0.3804];
5400            app.SolicitarDirectoPanel.Position = [218 1 429 128];
5401
5402            % Create PuntodePasoTriggerEditField_2Label
5403            app.PuntodePasoTriggerEditField_2Label = uilabel(app.
                    SolicitarDirectoPanel);
5404            app.PuntodePasoTriggerEditField_2Label.HorizontalAlignment = 'center
                    ';
5405            app.PuntodePasoTriggerEditField_2Label.FontColor = [1 1 1];
5406            app.PuntodePasoTriggerEditField_2Label.Position = [18 83 84 28];
5407            app.PuntodePasoTriggerEditField_2Label.Text = {'Punto de Paso'; '
                    Trigger'};
5408
5409            % Create WaypointTriggerSDEditField
5410            app.WaypointTriggerSDEditField = uieditfield(app.
                    SolicitarDirectoPanel, 'text');
5411            app.WaypointTriggerSDEditField.ValueChangedFcn = createCallbackFcn(
                    app, @WaypointTriggerSDEditFieldValueChanged, true);
5412            app.WaypointTriggerSDEditField.Position = [117 89 76 22];
5413
5414            % Create TiempodeSolicituddeDirectoLabel
5415            app.TiempodeSolicituddeDirectoLabel = uilabel(app.
                    SolicitarDirectoPanel);
5416            app.TiempodeSolicituddeDirectoLabel.HorizontalAlignment = 'center';
5417            app.TiempodeSolicituddeDirectoLabel.FontColor = [1 1 1];
5418            app.TiempodeSolicituddeDirectoLabel.Position = [51 30 110 28];
5419            app.TiempodeSolicituddeDirectoLabel.Text = {'Tiempo de Solicitud'; '
                    de Directo'};
5420
5421            % Create SDDropDown
5422            app.SDDropDown = uidropdown(app.SolicitarDirectoPanel);
5423            app.SDDropDown.Items = {'Antes', 'Después'};
5424            app.SDDropDown.ItemsData = {'1', '2'};
5425            app.SDDropDown.ValueChangedFcn = createCallbackFcn(app,
                    @SDDropDownValueChanged, true);
5426            app.SDDropDown.Position = [176 36 100 22];
5427            app.SDDropDown.Value = '1';
5428
5429            % Create PuntodePasoObjetivodeDirectoEditFieldLabel
5430            app.PuntodePasoObjetivodeDirectoEditFieldLabel = uilabel(app.
                    SolicitarDirectoPanel);
5431            app.PuntodePasoObjetivodeDirectoEditFieldLabel.HorizontalAlignment =
                     'center';
5432            app.PuntodePasoObjetivodeDirectoEditFieldLabel.FontColor = [1 1 1];
5433            app.PuntodePasoObjetivodeDirectoEditFieldLabel.Position = [216 83
                    112 28];
5434            app.PuntodePasoObjetivodeDirectoEditFieldLabel.Text = {'Punto de
                    Paso'; 'Objetivo de Directo'};
5435
```

```
5436            % Create WPDirectObjetivoEditField
5437            app.WPDirectObjetivoEditField = uieditfield(app.
                    SolicitarDirectoPanel, 'text');
5438            app.WPDirectObjetivoEditField.ValueChangedFcn = createCallbackFcn(
                    app, @WPDirectObjetivoEditFieldValueChanged, true);
5439            app.WPDirectObjetivoEditField.Position = [331 89 79 22];
5440
5441            % Create CRNPPanel
5442            app.CRNPPanel = uipanel(app.IncidenciasTab);
5443            app.CRNPPanel.BorderType = 'none';
5444            app.CRNPPanel.Visible = 'off';
5445            app.CRNPPanel.BackgroundColor = [0.0196 0.0196 0.3804];
5446            app.CRNPPanel.Position = [218 1 429 128];
5447
5448            % Create PuntodePasoTriggerEditFieldLabel_2
5449            app.PuntodePasoTriggerEditFieldLabel_2 = uilabel(app.CRNPPanel);
5450            app.PuntodePasoTriggerEditFieldLabel_2.HorizontalAlignment = 'center
                    ';
5451            app.PuntodePasoTriggerEditFieldLabel_2.FontColor = [1 1 1];
5452            app.PuntodePasoTriggerEditFieldLabel_2.Position = [18 83 84 28];
5453            app.PuntodePasoTriggerEditFieldLabel_2.Text = {'Punto de Paso'; '
                    Trigger'};
5454
5455            % Create WaypointTriggerCRNPEditField
5456            app.WaypointTriggerCRNPEditField = uieditfield(app.CRNPPanel, 'text'
                    );
5457            app.WaypointTriggerCRNPEditField.ValueChangedFcn = createCallbackFcn
                    (app, @WaypointTriggerCRNPEditFieldValueChanged, true);
5458            app.WaypointTriggerCRNPEditField.Position = [117 89 76 22];
5459
5460            % Create RumboNuevoEditFieldLabel
5461            app.RumboNuevoEditFieldLabel = uilabel(app.CRNPPanel);
5462            app.RumboNuevoEditFieldLabel.HorizontalAlignment = 'right';
5463            app.RumboNuevoEditFieldLabel.FontColor = [1 1 1];
5464            app.RumboNuevoEditFieldLabel.Position = [234 89 82 22];
5465            app.RumboNuevoEditFieldLabel.Text = 'Rumbo Nuevo';
5466
5467            % Create RumboNuevoEditField
5468            app.RumboNuevoEditField = uieditfield(app.CRNPPanel, 'numeric');
5469            app.RumboNuevoEditField.ValueChangedFcn = createCallbackFcn(app,
                    @RumboNuevoEditFieldValueChanged, true);
5470            app.RumboNuevoEditField.Position = [331 89 79 22];
5471
5472            % Create TiempodeTriggerLabel
5473            app.TiempodeTriggerLabel = uilabel(app.CRNPPanel);
5474            app.TiempodeTriggerLabel.HorizontalAlignment = 'right';
5475            app.TiempodeTriggerLabel.FontColor = [1 1 1];
5476            app.TiempodeTriggerLabel.Position = [58 36 103 22];
5477            app.TiempodeTriggerLabel.Text = 'Tiempo de Trigger';
5478
5479            % Create CRNPDropDown
5480            app.CRNPDropDown = uidropdown(app.CRNPPanel);
5481            app.CRNPDropDown.Items = {'Antes', 'Después'};
5482            app.CRNPDropDown.ItemsData = {'1', '2'};
5483            app.CRNPDropDown.ValueChangedFcn = createCallbackFcn(app,
                    @CRNPDropDownValueChanged, true);
5484            app.CRNPDropDown.Position = [176 36 100 22];
```

```
5485            app.CRNPDropDown.Value = '1';
5486
5487            % Create EliminarIncidenciaButton
5488            app.EliminarIncidenciaButton = uibutton(app.IncidenciasTab, 'push');
5489            app.EliminarIncidenciaButton.ButtonPushedFcn = createCallbackFcn(app
                    , @EliminarIncidenciaButtonPushed, true);
5490            app.EliminarIncidenciaButton.BackgroundColor = [0 0.451 0.7412];
5491            app.EliminarIncidenciaButton.FontColor = [1 1 1];
5492            app.EliminarIncidenciaButton.Position = [110 20 80 22];
5493            app.EliminarIncidenciaButton.Text = 'Eliminar';
5494
5495            % Create AirplaneTree
5496            app.AirplaneTree = uitree(app.SectorAirplanePanel);
5497            app.AirplaneTree.SelectionChangedFcn = createCallbackFcn(app,
                    @AirplaneTreeSelectionChanged, true);
5498            app.AirplaneTree.Position = [777 8 315 154];
5499
5500            % Create GuardarAirplaneButton
5501            app.GuardarAirplaneButton = uibutton(app.SectorAirplanePanel, 'push'
                    );
5502            app.GuardarAirplaneButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarAirplaneButtonPushed, true);
5503            app.GuardarAirplaneButton.BackgroundColor = [0 0.451 0.7412];
5504            app.GuardarAirplaneButton.FontColor = [1 1 1];
5505            app.GuardarAirplaneButton.Position = [679 101 92 22];
5506            app.GuardarAirplaneButton.Text = 'Guardar';
5507
5508            % Create EliminarAirplaneButton
5509            app.EliminarAirplaneButton = uibutton(app.SectorAirplanePanel, 'push
                    ');
5510            app.EliminarAirplaneButton.ButtonPushedFcn = createCallbackFcn(app,
                    @EliminarAirplaneButtonPushed, true);
5511            app.EliminarAirplaneButton.BackgroundColor = [0 0.451 0.7412];
5512            app.EliminarAirplaneButton.FontColor = [1 1 1];
5513            app.EliminarAirplaneButton.Position = [679 50 92 22];
5514            app.EliminarAirplaneButton.Text = 'Eliminar';
5515
5516            % Create GuardarSectorButton
5517            app.GuardarSectorButton = uibutton(app.EditordeEscenarios, 'push');
5518            app.GuardarSectorButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarSectorButtonPushed, true);
5519            app.GuardarSectorButton.BackgroundColor = [0 0.451 0.7412];
5520            app.GuardarSectorButton.FontColor = [1 1 1];
5521            app.GuardarSectorButton.Position = [1100 500 100 36];
5522            app.GuardarSectorButton.Text = {'Guardar'; 'Cambios'};
5523
5524            % Create HorariodeControlPanel
5525            app.HorariodeControlPanel = uipanel(app.EditordeEscenarios);
5526            app.HorariodeControlPanel.ForegroundColor = [1 1 1];
5527            app.HorariodeControlPanel.Title = 'Horario de Control';
5528            app.HorariodeControlPanel.BackgroundColor = [0 0.4471 0.7412];
5529            app.HorariodeControlPanel.Position = [643 371 125 170];
5530
5531            % Create HorainicioEditField
5532            app.HorainicioEditField = uieditfield(app.HorariodeControlPanel, '
                    text');
```

```
5533            app.HorainicioEditField.ValueChangedFcn = createCallbackFcn(app,
                    @HorainicioEditFieldValueChanged, true);
5534            app.HorainicioEditField.HorizontalAlignment = 'center';
5535            app.HorainicioEditField.FontSize = 14;
5536            app.HorainicioEditField.FontColor = [0 1 0];
5537            app.HorainicioEditField.BackgroundColor = [0 0 0];
5538            app.HorainicioEditField.Position = [8 110 110 25];
5539            app.HorainicioEditField.Value = '00:00';
5540
5541            % Create HorafinalEditField
5542            app.HorafinalEditField = uieditfield(app.HorariodeControlPanel, '
                    text');
5543            app.HorafinalEditField.ValueChangedFcn = createCallbackFcn(app,
                    @HorafinalEditFieldValueChanged, true);
5544            app.HorafinalEditField.HorizontalAlignment = 'center';
5545            app.HorafinalEditField.FontSize = 14;
5546            app.HorafinalEditField.FontColor = [0 1 0];
5547            app.HorafinalEditField.BackgroundColor = [0 0 0];
5548            app.HorafinalEditField.Position = [8 75 110 25];
5549            app.HorafinalEditField.Value = '00:00';
5550
5551            % Create GuardarHorarioButton
5552            app.GuardarHorarioButton = uibutton(app.HorariodeControlPanel, 'push
                    ');
5553            app.GuardarHorarioButton.ButtonPushedFcn = createCallbackFcn(app,
                    @GuardarHorarioButtonPushed, true);
5554            app.GuardarHorarioButton.BackgroundColor = [0 0 1];
5555            app.GuardarHorarioButton.FontColor = [1 1 1];
5556            app.GuardarHorarioButton.Position = [14 37 100 22];
5557            app.GuardarHorarioButton.Text = 'Guardar';
5558
5559            % Create Simulador
5560            app.Simulador = uipanel(app.ATCMakerApp);
5561            app.Simulador.Visible = 'off';
5562            app.Simulador.BackgroundColor = [0 0.1098 0.2196];
5563            app.Simulador.Position = [1 1 1280 720];
5564
5565            % Create FlightStripRackPanel
5566            app.FlightStripRackPanel = uipanel(app.Simulador);
5567            app.FlightStripRackPanel.BackgroundColor = [0 0.1098 0.2196];
5568            app.FlightStripRackPanel.Scrollable = 'on';
5569            app.FlightStripRackPanel.Position = [31 155 260 531];
5570
5571            % Create FLSRButton
5572            app.FLSRButton = uibutton(app.FlightStripRackPanel, 'state');
5573            app.FLSRButton.ValueChangedFcn = createCallbackFcn(app,
                    @FLSRButtonValueChanged, true);
5574            app.FLSRButton.Text = '';
5575            app.FLSRButton.Position = [20 20 225 100];
5576
5577            % Create FLSRCSEF
5578            app.FLSRCSEF = uieditfield(app.FlightStripRackPanel, 'text');
5579            app.FLSRCSEF.Editable = 'off';
5580            app.FLSRCSEF.Position = [40 89 100 22];
5581
5582            % Create FLSRFLEF
5583            app.FLSRFLEF = uieditfield(app.FlightStripRackPanel, 'text');
```

```
5584            app.FLSRFLEF.Editable = 'off';
5585            app.FLSRFLEF.Position = [40 59 70 22];
5586
5587            % Create FLSRSPDEF
5588            app.FLSRSPDEF = uieditfield(app.FlightStripRackPanel, 'text');
5589            app.FLSRSPDEF.Editable = 'off';
5590            app.FLSRSPDEF.Position = [40 29 70 22];
5591
5592            % Create FLSRLamp
5593            app.FLSRLamp = uilamp(app.FlightStripRackPanel);
5594            app.FLSRLamp.Enable = 'off';
5595            app.FLSRLamp.Position = [205 60 20 20];
5596            app.FLSRLamp.Color = [1 0 0];
5597
5598            % Create FlightStripPanel
5599            app.FlightStripPanel = uipanel(app.Simulador);
5600            app.FlightStripPanel.Title = 'Ficha de Progresión de Vuelo';
5601            app.FlightStripPanel.BackgroundColor = [1 0.902 0.6588];
5602            app.FlightStripPanel.Scrollable = 'on';
5603            app.FlightStripPanel.Position = [31 30 524 113];
5604
5605            % Create FlightRoutePanel
5606            app.FlightRoutePanel = uipanel(app.FlightStripPanel);
5607            app.FlightRoutePanel.BackgroundColor = [1 0.902 0.6588];
5608            app.FlightRoutePanel.Scrollable = 'on';
5609            app.FlightRoutePanel.Position = [168 3 355 89];
5610
5611            % Create WPRNEF
5612            app.WPRNEF = uieditfield(app.FlightRoutePanel, 'text');
5613            app.WPRNEF.Editable = 'off';
5614            app.WPRNEF.BackgroundColor = [1 0.9412 0.7804];
5615            app.WPRNEF.Position = [10 47 55 22];
5616
5617            % Create TPWPEF
5618            app.TPWPEF = uieditfield(app.FlightRoutePanel, 'text');
5619            app.TPWPEF.ValueChangedFcn = createCallbackFcn(app,
                    @TPWPEFValueChanged, true);
5620            app.TPWPEF.BackgroundColor = [1 0.9373 0.7804];
5621            app.TPWPEF.Position = [10 20 55 22];
5622
5623            % Create FSCallsignEditField
5624            app.FSCallsignEditField = uieditfield(app.FlightStripPanel, 'text');
5625            app.FSCallsignEditField.Editable = 'off';
5626            app.FSCallsignEditField.BackgroundColor = [1 0.9373 0.7804];
5627            app.FSCallsignEditField.Position = [25 59 100 22];
5628
5629            % Create FSFLEditField
5630            app.FSFLEditField = uieditfield(app.FlightStripPanel, 'text');
5631            app.FSFLEditField.Editable = 'off';
5632            app.FSFLEditField.BackgroundColor = [1 0.9373 0.7804];
5633            app.FSFLEditField.Position = [25 34 70 22];
5634
5635            % Create FSSPDEditField
5636            app.FSSPDEditField = uieditfield(app.FlightStripPanel, 'text');
5637            app.FSSPDEditField.Editable = 'off';
5638            app.FSSPDEditField.BackgroundColor = [1 0.9373 0.7804];
5639            app.FSSPDEditField.Position = [25 9 70 22];
```

```
5640
5641            % Create FSETOEditField
5642            app.FSETOEditField = uieditfield(app.FlightStripPanel, 'text');
5643            app.FSETOEditField.Editable = 'off';
5644            app.FSETOEditField.BackgroundColor = [1 0.9373 0.7804];
5645            app.FSETOEditField.Position = [105 9 55 22];
5646
5647            % Create Simuladorgraph
5648            app.Simuladorgraph = uiaxes(app.Simulador);
5649            title(app.Simuladorgraph, '')
5650            xlabel(app.Simuladorgraph, '')
5651            ylabel(app.Simuladorgraph, '')
5652            app.Simuladorgraph.PlotBoxAspectRatio = [1 1 1];
5653            app.Simuladorgraph.GridColor = [0 1 0];
5654            app.Simuladorgraph.XColor = [0 1 0];
5655            app.Simuladorgraph.YColor = [0 1 0];
5656            app.Simuladorgraph.Color = [0 0 0];
5657            app.Simuladorgraph.XGrid = 'on';
5658            app.Simuladorgraph.YGrid = 'on';
5659            app.Simuladorgraph.BackgroundColor = [0 0.1098 0.2196];
5660            app.Simuladorgraph.Position = [350 157 595 529];
5661
5662            % Create SalirSimButton
5663            app.SalirSimButton = uibutton(app.Simulador, 'push');
5664            app.SalirSimButton.ButtonPushedFcn = createCallbackFcn(app,
                    @SalirSimButtonPushed, true);
5665            app.SalirSimButton.BackgroundColor = [0 0.4471 0.7412];
5666            app.SalirSimButton.FontColor = [1 1 1];
5667            app.SalirSimButton.Position = [1163 685 100 22];
5668            app.SalirSimButton.Text = 'Salir';
5669
5670            % Create PlaySimButton
5671            app.PlaySimButton = uibutton(app.Simulador, 'state');
5672            app.PlaySimButton.ValueChangedFcn = createCallbackFcn(app,
                    @PlaySimButtonValueChanged, true);
5673            app.PlaySimButton.Icon = 'Playicon.png';
5674            app.PlaySimButton.IconAlignment = 'center';
5675            app.PlaySimButton.Text = '';
5676            app.PlaySimButton.BackgroundColor = [0.2392 0.949 0.2392];
5677            app.PlaySimButton.FontName = 'Kunstler Script';
5678            app.PlaySimButton.FontWeight = 'bold';
5679            app.PlaySimButton.Position = [584 121 60 30];
5680
5681            % Create PauseSimButton
5682            app.PauseSimButton = uibutton(app.Simulador, 'state');
5683            app.PauseSimButton.ValueChangedFcn = createCallbackFcn(app,
                    @PauseSimButtonValueChanged, true);
5684            app.PauseSimButton.Enable = 'off';
5685            app.PauseSimButton.Icon = 'Pauseicon.png';
5686            app.PauseSimButton.IconAlignment = 'center';
5687            app.PauseSimButton.Text = '';
5688            app.PauseSimButton.BackgroundColor = [0.2392 0.949 0.2392];
5689            app.PauseSimButton.FontName = 'Arial Black';
5690            app.PauseSimButton.Position = [670 121 60 30];
5691            app.PauseSimButton.Value = true;
5692
5693            % Create TimeSimEditField
```

```
5694            app.TimeSimEditField = uieditfield(app.Simulador, 'text');
5695            app.TimeSimEditField.Editable = 'off';
5696            app.TimeSimEditField.HorizontalAlignment = 'center';
5697            app.TimeSimEditField.FontSize = 30;
5698            app.TimeSimEditField.FontColor = [0 1 0];
5699            app.TimeSimEditField.BackgroundColor = [0 0 0];
5700            app.TimeSimEditField.Position = [575 70 168 38];
5701            app.TimeSimEditField.Value = '00:00:00';
5702
5703            % Create TimeEndEditField
5704            app.TimeEndEditField = uieditfield(app.Simulador, 'text');
5705            app.TimeEndEditField.Editable = 'off';
5706            app.TimeEndEditField.HorizontalAlignment = 'center';
5707            app.TimeEndEditField.FontSize = 18;
5708            app.TimeEndEditField.FontAngle = 'italic';
5709            app.TimeEndEditField.FontColor = [1 1 1];
5710            app.TimeEndEditField.BackgroundColor = [0 0.1098 0.2196];
5711            app.TimeEndEditField.Position = [575 34 168 26];
5712            app.TimeEndEditField.Value = '00:00:00';
5713
5714            % Create CommsTextArea
5715            app.CommsTextArea = uitextarea(app.Simulador);
5716            app.CommsTextArea.Editable = 'off';
5717            app.CommsTextArea.Position = [957 172 295 502];
5718            app.CommsTextArea.Value = {' '};
5719
5720            % Create CommsInputPanel
5721            app.CommsInputPanel = uipanel(app.Simulador);
5722            app.CommsInputPanel.BackgroundColor = [0 0.1098 0.2196];
5723            app.CommsInputPanel.Scrollable = 'on';
5724            app.CommsInputPanel.Position = [767 16 485 140];
5725
5726            % Create InputmsgTextArea
5727            app.InputmsgTextArea = uitextarea(app.CommsInputPanel);
5728            app.InputmsgTextArea.Editable = 'off';
5729            app.InputmsgTextArea.Position = [10 110 390 22];
5730
5731            % Create SendButton
5732            app.SendButton = uibutton(app.CommsInputPanel, 'push');
5733            app.SendButton.ButtonPushedFcn = createCallbackFcn(app,
                    @SendButtonPushed, true);
5734            app.SendButton.BackgroundColor = [0 0.4471 0.7412];
5735            app.SendButton.FontColor = [1 1 1];
5736            app.SendButton.Position = [407 110 67 22];
5737            app.SendButton.Text = 'Enviar';
5738
5739            % Create DropDoum
5740            app.DropDown = uidropdown(app.CommsInputPanel);
5741            app.DropDown.Items = {'--/--'};
5742            app.DropDown.ItemsData = {'0'};
5743            app.DropDown.ValueChangedFcn = createCallbackFcn(app,
                    @DropDownValueChanged, true);
5744            app.DropDown.Position = [10 81 84 22];
5745            app.DropDown.Value = '0';
5746
5747            % Create AutorizarEntradaButton
5748            app.AutorizarEntradaButton = uibutton(app.CommsInputPanel, 'push');
```

```
5749            app.AutorizarEntradaButton.ButtonPushedFcn = createCallbackFcn(app,
                    @AutorizarEntradaButtonPushed, true);
5750            app.AutorizarEntradaButton.BackgroundColor = [0 0.4471 0.7412];
5751            app.AutorizarEntradaButton.FontColor = [1 1 1];
5752            app.AutorizarEntradaButton.Position = [100 67 100 36];
5753            app.AutorizarEntradaButton.Text = {'Autorizar'; 'Entrada'};
5754
5755            % Create AutorizarCambioFLButton
5756            app.AutorizarCambioFLButton = uibutton(app.CommsInputPanel, 'push');
5757            app.AutorizarCambioFLButton.ButtonPushedFcn = createCallbackFcn(app,
                     @AutorizarCambioFLButtonPushed, true);
5758            app.AutorizarCambioFLButton.BackgroundColor = [0 0.4471 0.7412];
5759            app.AutorizarCambioFLButton.FontColor = [1 1 1];
5760            app.AutorizarCambioFLButton.Position = [210 67 100 36];
5761            app.AutorizarCambioFLButton.Text = {'Autorizar'; 'Cambio FL'};
5762
5763            % Create NoAutorizarCambioFLButton
5764            app.NoAutorizarCambioFLButton = uibutton(app.CommsInputPanel, 'push'
                    );
5765            app.NoAutorizarCambioFLButton.ButtonPushedFcn = createCallbackFcn(
                    app, @NoAutorizarCambioFLButtonPushed, true);
5766            app.NoAutorizarCambioFLButton.BackgroundColor = [0 0.4471 0.7412];
5767            app.NoAutorizarCambioFLButton.FontColor = [1 1 1];
5768            app.NoAutorizarCambioFLButton.Position = [320 67 100 36];
5769            app.NoAutorizarCambioFLButton.Text = {'No Autorizar'; 'Cambio FL'};
5770
5771            % Create AutorizarDirectoButton
5772            app.AutorizarDirectoButton = uibutton(app.CommsInputPanel, 'push');
5773            app.AutorizarDirectoButton.ButtonPushedFcn = createCallbackFcn(app,
                    @AutorizarDirectoButtonPushed, true);
5774            app.AutorizarDirectoButton.BackgroundColor = [0 0.4471 0.7412];
5775            app.AutorizarDirectoButton.FontColor = [1 1 1];
5776            app.AutorizarDirectoButton.Position = [10 22 100 36];
5777            app.AutorizarDirectoButton.Text = {'Autorizar'; 'Directo'};
5778
5779            % Create NoAutorizarDirectoButton
5780            app.NoAutorizarDirectoButton = uibutton(app.CommsInputPanel, 'push')
                    ;
5781            app.NoAutorizarDirectoButton.ButtonPushedFcn = createCallbackFcn(app
                    , @NoAutorizarDirectoButtonPushed, true);
5782            app.NoAutorizarDirectoButton.BackgroundColor = [0 0.4471 0.7412];
5783            app.NoAutorizarDirectoButton.FontColor = [1 1 1];
5784            app.NoAutorizarDirectoButton.Position = [120 22 100 36];
5785            app.NoAutorizarDirectoButton.Text = {'No Autorizar'; 'Directo'};
5786
5787            % Create CambioaRumboAutorizadoButton
5788            app.CambioaRumboAutorizadoButton = uibutton(app.CommsInputPanel, '
                    push');
5789            app.CambioaRumboAutorizadoButton.ButtonPushedFcn = createCallbackFcn
                    (app, @CambioaRumboAutorizadoButtonPushed, true);
5790            app.CambioaRumboAutorizadoButton.BackgroundColor = [0 0.4471
                    0.7412];
5791            app.CambioaRumboAutorizadoButton.FontColor = [1 1 1];
5792            app.CambioaRumboAutorizadoButton.Position = [230 22 110 36];
5793            app.CambioaRumboAutorizadoButton.Text = {'Cambio a Rumbo'; '
                    Autorizado'};
5794
```

```
5795            % Create CambioFrecuenciaButton
5796            app.CambioFrecuenciaButton = uibutton(app.CommsInputPanel, 'push');
5797            app.CambioFrecuenciaButton.ButtonPushedFcn = createCallbackFcn(app,
                    @CambioFrecuenciaButtonPushed, true);
5798            app.CambioFrecuenciaButton.BackgroundColor = [0 0.4471 0.7412];
5799            app.CambioFrecuenciaButton.FontColor = [1 1 1];
5800            app.CambioFrecuenciaButton.Position = [350 22 100 36];
5801            app.CambioFrecuenciaButton.Text = {'Cambio'; 'Frecuencia'};
5802
5803            % Create GeneralConflictLamp
5804            app.GeneralConflictLamp = uilamp(app.Simulador);
5805            app.GeneralConflictLamp.Enable = 'off';
5806            app.GeneralConflictLamp.Position = [306 664 20 20];
5807            app.GeneralConflictLamp.Color = [1 0 0];
5808
5809            % Create Editorseleccion
5810            app.Editorseleccion = uipanel(app.ATCMakerApp);
5811            app.Editorseleccion.Visible = 'off';
5812            app.Editorseleccion.BackgroundColor = [0 0.1098 0.2196];
5813            app.Editorseleccion.Position = [1 1 1280 720];
5814
5815            % Create Editorseleccionbackbutton
5816            app.Editorseleccionbackbutton = uibutton(app.Editorseleccion, 'push'
                    );
5817            app.Editorseleccionbackbutton.ButtonPushedFcn = createCallbackFcn(
                    app, @EditorseleccionbackbuttonButtonPushed, true);
5818            app.Editorseleccionbackbutton.BackgroundColor = [0 0.451 0.7412];
5819            app.Editorseleccionbackbutton.FontColor = [1 1 1];
5820            app.Editorseleccionbackbutton.Tooltip = {'Volver a la Pantalla de
                    Inicio'};
5821            app.Editorseleccionbackbutton.Position = [1100 640 100 30];
5822            app.Editorseleccionbackbutton.Text = 'Atrás';
5823
5824            % Create TreeEditor
5825            app.TreeEditor = uitree(app.Editorseleccion);
5826            app.TreeEditor.SelectionChangedFcn = createCallbackFcn(app,
                    @TreeEditorSelectionChanged, true);
5827            app.TreeEditor.Position = [80 80 400 560];
5828
5829            % Create NuevoEscenario
5830            app.NuevoEscenario = uipanel(app.Editorseleccion);
5831            app.NuevoEscenario.ForegroundColor = [1 1 1];
5832            app.NuevoEscenario.Title = 'Nuevo Escenario';
5833            app.NuevoEscenario.Visible = 'off';
5834            app.NuevoEscenario.BackgroundColor = [0 0.3333 0.5098];
5835            app.NuevoEscenario.Position = [390 235 500 250];
5836
5837            % Create NombredelescenarioEditFieldLabel
5838            app.NombredelescenarioEditFieldLabel = uilabel(app.NuevoEscenario);
5839            app.NombredelescenarioEditFieldLabel.HorizontalAlignment = 'right';
5840            app.NombredelescenarioEditFieldLabel.FontColor = [1 1 1];
5841            app.NombredelescenarioEditFieldLabel.Position = [10 136 123 22];
5842            app.NombredelescenarioEditFieldLabel.Text = 'Nombre del escenario';
5843
5844            % Create NombredelnuevoescenarioEditField
5845            app.NombredelnuevoescenarioEditField = uieditfield(app.
                    NuevoEscenario, 'text');
```

```
5846            app.NombredelnuevoescenarioEditField.ValueChangedFcn =
                    createCallbackFcn(app,
                    @NombredelnuevoescenarioEditFieldValueChanged, true);
5847            app.NombredelnuevoescenarioEditField.Position = [148 136 337 22];
5848
5849            % Create CrearButton
5850            app.CrearButton = uibutton(app.NuevoEscenario, 'push');
5851            app.CrearButton.ButtonPushedFcn = createCallbackFcn(app,
                    @CrearButtonPushed, true);
5852            app.CrearButton.BackgroundColor = [0 0.451 0.7412];
5853            app.CrearButton.FontColor = [1 1 1];
5854            app.CrearButton.Tooltip = {'Crear Escenario'};
5855            app.CrearButton.Position = [164 64 176 22];
5856            app.CrearButton.Text = 'Crear';
5857
5858            % Create AtrsButton
5859            app.AtrsButton = uibutton(app.NuevoEscenario, 'push');
5860            app.AtrsButton.ButtonPushedFcn = createCallbackFcn(app,
                    @AtrsButtonPushed, true);
5861            app.AtrsButton.BackgroundColor = [0 0.451 0.7412];
5862            app.AtrsButton.FontColor = [1 1 1];
5863            app.AtrsButton.Tooltip = {'Vover a Selección de Escenarios'};
5864            app.AtrsButton.Position = [385 194 100 22];
5865            app.AtrsButton.Text = 'Atrás';
5866
5867            % Create NuevoEscenarioButton
5868            app.NuevoEscenarioButton = uibutton(app.Editorseleccion, 'push');
5869            app.NuevoEscenarioButton.ButtonPushedFcn = createCallbackFcn(app,
                    @NuevoEscenarioButtonPushed, true);
5870            app.NuevoEscenarioButton.BackgroundColor = [0 0.451 0.7412];
5871            app.NuevoEscenarioButton.FontColor = [1 1 1];
5872            app.NuevoEscenarioButton.Tooltip = {'Crear un Nuevo Escenario'};
5873            app.NuevoEscenarioButton.Position = [522 618 148 22];
5874            app.NuevoEscenarioButton.Text = 'Nuevo Escenario';
5875
5876            % Create RenombrarEscenarioButton
5877            app.RenombrarEscenarioButton = uibutton(app.Editorseleccion, 'push')
                    ;
5878            app.RenombrarEscenarioButton.ButtonPushedFcn = createCallbackFcn(app
                    , @RenombrarEscenarioButtonPushed, true);
5879            app.RenombrarEscenarioButton.BackgroundColor = [0 0.451 0.7412];
5880            app.RenombrarEscenarioButton.FontColor = [1 1 1];
5881            app.RenombrarEscenarioButton.Tooltip = {'Renombrar Escenario
                    Seleccionado'};
5882            app.RenombrarEscenarioButton.Position = [704 618 148 22];
5883            app.RenombrarEscenarioButton.Text = 'Renombrar Escenario';
5884
5885            % Create RenombrarEscenario
5886            app.RenombrarEscenario = uipanel(app.Editorseleccion);
5887            app.RenombrarEscenario.ForegroundColor = [1 1 1];
5888            app.RenombrarEscenario.Title = 'Renombrar Escenario';
5889            app.RenombrarEscenario.Visible = 'off';
5890            app.RenombrarEscenario.BackgroundColor = [0 0.3333 0.5098];
5891            app.RenombrarEscenario.Position = [390 235 500 250];
5892
5893            % Create NombredelescenarioEditFieldLabel_2
```

```
5894            app.NombredelescenarioEditFieldLabel_2 = uilabel(app.
                    RenombrarEscenario);
5895            app.NombredelescenarioEditFieldLabel_2.HorizontalAlignment = 'right'
                    ;
5896            app.NombredelescenarioEditFieldLabel_2.FontColor = [1 1 1];
5897            app.NombredelescenarioEditFieldLabel_2.Position = [10 136 123 22];
5898            app.NombredelescenarioEditFieldLabel_2.Text = 'Nombre del escenario'
                    ;
5899
5900            % Create NombrenuevodelnuevoescenarioEditField
5901            app.NombrenuevodelnuevoescenarioEditField = uieditfield(app.
                    RenombrarEscenario, 'text');
5902            app.NombrenuevodelnuevoescenarioEditField.ValueChangedFcn =
                    createCallbackFcn(app,
                    @NombrenuevodelnuevoescenarioEditFieldValueChanged, true);
5903            app.NombrenuevodelnuevoescenarioEditField.Position = [148 136 337
                    22];
5904
5905            % Create RenombrarButton
5906            app.RenombrarButton = uibutton(app.RenombrarEscenario, 'push');
5907            app.RenombrarButton.ButtonPushedFcn = createCallbackFcn(app,
                    @RenombrarButtonPushed, true);
5908            app.RenombrarButton.BackgroundColor = [0 0.451 0.7412];
5909            app.RenombrarButton.FontColor = [1 1 1];
5910            app.RenombrarButton.Tooltip = {'Renombrar Escenario'};
5911            app.RenombrarButton.Position = [164 64 176 22];
5912            app.RenombrarButton.Text = 'Renombrar';
5913
5914            % Create AtrsButton_2
5915            app.AtrsButton_2 = uibutton(app.RenombrarEscenario, 'push');
5916            app.AtrsButton_2.ButtonPushedFcn = createCallbackFcn(app,
                    @AtrsButton_2Pushed, true);
5917            app.AtrsButton_2.BackgroundColor = [0 0.451 0.7412];
5918            app.AtrsButton_2.FontColor = [1 1 1];
5919            app.AtrsButton_2.Tooltip = {'Vover a Selección de Escenarios'};
5920            app.AtrsButton_2.Position = [385 194 100 22];
5921            app.AtrsButton_2.Text = 'Atrás';
5922
5923            % Create EliminarEscenarioButton
5924            app.EliminarEscenarioButton = uibutton(app.Editorseleccion, 'push');
5925            app.EliminarEscenarioButton.ButtonPushedFcn = createCallbackFcn(app,
                     @EliminarEscenarioButtonPushed, true);
5926            app.EliminarEscenarioButton.BackgroundColor = [0 0.451 0.7412];
5927            app.EliminarEscenarioButton.FontColor = [1 1 1];
5928            app.EliminarEscenarioButton.Position = [886 618 148 22];
5929            app.EliminarEscenarioButton.Text = 'Eliminar Escenario';
5930
5931            % Create CargarEscenarioButton
5932            app.CargarEscenarioButton = uibutton(app.Editorseleccion, 'push');
5933            app.CargarEscenarioButton.ButtonPushedFcn = createCallbackFcn(app,
                    @CargarEscenarioButtonPushed, true);
5934            app.CargarEscenarioButton.BackgroundColor = [0 0.451 0.7412];
5935            app.CargarEscenarioButton.FontColor = [1 1 1];
5936            app.CargarEscenarioButton.Position = [522 118 148 22];
5937            app.CargarEscenarioButton.Text = 'Cargar Escenario';
5938
5939            % Create Simuladorseleccion
```

```
5940            app.Simuladorseleccion = uipanel(app.ATCMakerApp);
5941            app.Simuladorseleccion.Visible = 'off';
5942            app.Simuladorseleccion.BackgroundColor = [0 0.1098 0.2196];
5943            app.Simuladorseleccion.Position = [1 1 1280 720];
5944
5945            % Create Simuladorseleccionbackcutton
5946            app.Simuladorseleccionbackcutton = uibutton(app.Simuladorseleccion,
                    'push');
5947            app.Simuladorseleccionbackcutton.ButtonPushedFcn = createCallbackFcn
                    (app, @SimuladorseleccionbackcuttonButtonPushed, true);
5948            app.Simuladorseleccionbackcutton.BackgroundColor = [0 0.451 0.7412];
5949            app.Simuladorseleccionbackcutton.FontColor = [1 1 1];
5950            app.Simuladorseleccionbackcutton.Tooltip = {'Volver a la Pantalla de
                     Inicio'};
5951            app.Simuladorseleccionbackcutton.Position = [1100 640 100 30];
5952            app.Simuladorseleccionbackcutton.Text = 'Atrás';
5953
5954            % Create TreeSimulador
5955            app.TreeSimulador = uitree(app.Simuladorseleccion);
5956            app.TreeSimulador.Position = [80 80 400 560];
5957
5958            % Create CargarEscenarioSimButton
5959            app.CargarEscenarioSimButton = uibutton(app.Simuladorseleccion, '
                    push');
5960            app.CargarEscenarioSimButton.ButtonPushedFcn = createCallbackFcn(app
                    , @CargarEscenarioSimButtonPushed, true);
5961            app.CargarEscenarioSimButton.BackgroundColor = [0 0.451 0.7412];
5962            app.CargarEscenarioSimButton.FontColor = [1 1 1];
5963            app.CargarEscenarioSimButton.Position = [522 618 148 22];
5964            app.CargarEscenarioSimButton.Text = 'Cargar Escenario';
5965
5966            % Show the figure after all components are created
5967            app.ATCMakerApp.Visible = 'on';
5968        end
5969    end
5970
5971    % App creation and deletion
5972    methods (Access = public)
5973
5974        % Construct app
5975        function app = ATC_Maker
5976
5977            % Create UIFigure and components
5978            createComponents(app)
5979
5980            % Register the app with App Designer
5981            registerApp(app, app.ATCMakerApp)
5982
5983            % Execute the startup function
5984            runStartupFcn(app, @startupFcn)
5985
5986            if nargout == 0
5987                clear app
5988            end
5989        end
5990
5991        % Code that executes before app deletion
```

```
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.ATCMakerApp)
        end
    end
end
```

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] OACI *Anexo 2 al Convenio sobre Aviación Civil Internacional, Reglamento del Aire*, en vigor desde el 24 de Noviembre de 2005, Décima Edición, Disponible en: https://www.udi.edu.co/images/biblioteca/aeronautica/anexo2.pdf

[2] OACI *Anexo 11 al Convenio sobre Aviación Civil Internacional, Servicios de Tránsito Aéreo*, en vigor desde el 10 de Noviembre de 2016, Decimocuarta Edición, Disponible en: https://www.anac.gov.ar/anac/web/uploads/normativa/anexos-oaci/anexo-11.pdf

[3] OACI *Documento 4444, Gestión del Tránsito Aéreo*, en vigor desde el 1 de Noviembre de 2001, Decimocuarta Edición, Disponible en: https://www.proteccioncivil.es/catalogo/carpeta02/carpeta24/vademecum19/vdm02515ar/doc4444sp.pdf

[4] A.V. Romero, *Gestión del Tráfico Áereo*, ETSI, Universidad de Sevilla

[5] Rams Plus Features. [Online]. Disponible en: https://www.ramsplus.com/features/

[6] Skybrary, Reorganized ATC Mathematical Simulator Plus. [Online]. Disponible en: https://skybrary.aero/articles/rams-plus

[7] BlueSky Homepage. [Online]. Disponible en: http://homepage.tudelft.nl/7p97s/bluesky/

[8] TUDelft, BlueSky. [Online]. Disponible en: https://cs.lr.tudelft.nl/atm/software/bluesky/

[9] GitHub, BlueSky. [Online]. Disponible en: https://github.com/TUDelft-CNS-ATM/bluesky

[10] Eurocontrol, Base of Aircraft Data (BADA) [Online]. Disponible en: https://www.eurocontrol.int/publication/base-aircraft-data-bada-product-management-document

[11] Github Wiki, BlueSky Tutorials. [Online]. Disponible en: https://github.com/TUDelft-CNS-ATM/bluesky/wiki/Tutorials

[12] Wikipedia.org. (2022) International Virtual Aviation Organisation. [Online]. Disponible en: https://en.wikipedia.org/wiki/International_Virtual_Aviation_Organisation

[13] IVAO Homepage. [Online]. Disponible en: https://es.ivao.aero/

[14] IVAO, Manual de Usuario ATC Aurora. [Online]. Disponible en: https://wiki.ivao.aero/es/home/devops/manuals/Aurora_Manual

[15] Eurocontrol, Aircraft Performance Database, A320. [Online]. Disponible en: https://contentzone.eurocontrol.int/aircraftperformance/details.aspx?ICAO=A320

[16] Aerowiki, Virajes Coordinados. (2013). [Online]. Disponible en: http://aerowiki-info.blogspot.com/2013/08/virajes-coordinados.html

[17] E. Méndez, *ICARD, International Codes and Route Designators*, OACI, Ciudad de México, Abril de 2019