Índice General

[1 INTEGRACIÓN CON LA INTERFAZ EXISTENTE 2](#_Toc242543481)

[1.1 Descripción de la Interfaz 2](#_Toc242543482)

[1.1.1 Diagrama de funcionamiento del sistema BCI 2](#_Toc242543483)

[1.1.2 Adquisición de los datos 3](#_Toc242543484)

[1.1.3 Procesado de los datos 4](#_Toc242543485)

[1.1.4 Clasificador 6](#_Toc242543486)

[1.1.5 Realimentación o biofeedback al sujeto. 6](#_Toc242543487)

[1.2 Colisiones: sensores de proximidad 7](#_Toc242543488)

[1.3 Conversión a VRML 97 9](#_Toc242543489)

[1.4 Integración de Vivienda y ETSIT 12](#_Toc242543490)

[1.5 Integración de Simulador de Vuelo 12](#_Toc242543491)

[1.5.1 Integración con el interfaz de navegación 13](#_Toc242543492)

[1.5.2 Implementación del movimiento 14](#_Toc242543493)

[2 PRUEBAS Y EVALUACIÓN 16](#_Toc242543494)

Índice de Figuras

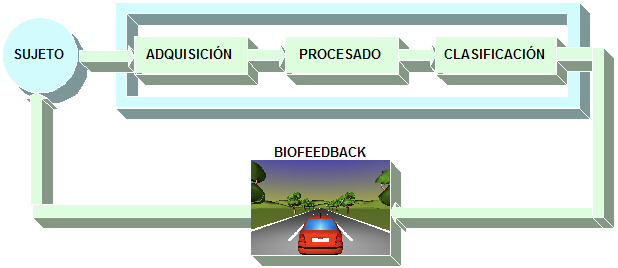
[Figura 1‑11: Cuadro de diálogo VRML 97 Exporter 2](#_Toc241207525)

# INTEGRACIÓN CON LA INTERFAZ EXISTENTE

## Descripción de la Interfaz

### Diagrama de funcionamiento del sistema BCI

El diagrama del sistema BCI con el que se realiza la integración se puede representar con el esquema de la siguiente figura.



*Figura 5.2: Diagrama básico de la BCI creada.*

Como se ha podido ver en la figura anterior, el sistema BCI consta de las siguientes fases: *adquisición*, *procesado*, *clasificación* y *realimentación*.

* **Adquisición:** Obtiene las señales electroencefalográficas del sujeto bajo prueba, mediante una serie de electrodos colocados en el cuero cabelludo. Las señales captadas pasan a un polígrafo, dispositivo encargado de amplificarlas. Los valores de salida del polígrafo son introducidos en una tarjeta de adquisición, dedicada a la conversión analógica-digital para que puedan ser procesadas por el ordenador.
* **Procesado de señal:** Extrae los parámetros de interés de las señales adquiridas, que se emplearán posteriormente en la modificación de la interfaz gráfica.
* **Clasificación:** Traduce los parámetros obtenidos durante el procesado en modificaciones de la interfaz gráfica. El encargado de realizar las modificaciones, en tiempo real, sobre la interfaz es el *trasladador*. En este caso, el trasladador está formado por la interfaz gráfica y los comandos que la gestionan.
* **Realimentación o biofeedback:** Informa al sujeto de cómo su actividad mental está afectando a lo que se le presenta a través de la interfaz gráfica. Se emplea feedback continuo, permitiendo al sujeto ver durante toda la prueba cómo está ejecutando la tarea mental.

El último bloque es el que se ha desarrollado en este proyecto, así como la integración con el resto de ellos. El resto, es decir, adquisición, procesado y clasificación, fueron realizados por proyectos fin de carrera anteriores dentro del Departamento de Tecnología Electrónica.

### Adquisición de los datos

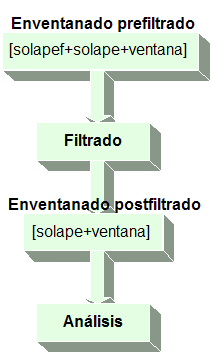
Los datos son capturados mediante una tarjeta de adquisición configurada a través del *DAQ Toolbox* de MATLAB. Se encarga de transferir las muestras de las señales adquiridas.

La captura de las señales del sujeto es realizada por la tarjeta *DAQCard-6024E*. La comunicación tarjeta-aplicación es proporcionada por los propios *drivers* de *DAQCard-6024E*, de *National Instruments* (*NI-DAQ*).

Los paquetes de datos procedentes de cada canal de adquisición del polígrafo son recibidos por la tarjeta. Cuando esto ocurre, avisa al módulo de adquisición mediante funciones “callback”, que contienen las instrucciones necesarias para el tratamiento de estos datos. Finalmente, es interesante comentar que el sistema BCI implementado en el proyecto solamente utilizará dos de los cuatro canales de los que dispone el sistema de referencia.

### Procesado de los datos

La aplicación realiza distintos procesos sobre cada una de las señales EEG: *enventanado prefiltrado*, *filtrado*, *enventanado postfiltrado* y *análisis*. La figura 5.3 muestra el esquema que sigue el procesado de datos.



*Figura 5.3: Proceso de Análisis seguido por la señal.*

A continuación, se comentarán los distintos pasos que sigue la señal en el proceso de análisis:

* **Enventanado prefiltrado:** Toma una secuencia de muestras de longitud *solapef+solape+ventana*. El solape adicional (*solapef*), se incluye con el objetivo de que absorba el efecto del transitorio del filtrado.
* **Filtrado:** La señal pasa por un filtro creado en un editor. Este editor es mostrado al pulsar el botón *Editor Filtros* del panel de control. En él se puede definir el tipo de filtro, el orden, la banda de frecuencia, etc. De hecho, el filtro creado en la aplicación es de *Butterworth* debido a que es la primera opción dada por el editor y a que es fácil de implementar. También es Paso Banda, de orden 5 y suele estar en la banda de frecuencia entre 8 y 12 Hz, que es la banda donde suelen aparecer las ondas μ empleadas en el sistema BCI, aunque dependiendo del individuo la banda de frecuencia será distinta.
* **Enventanado postfiltrado:** Elimina las muestras incluidas por *solapef*, ya que después de haber absorbido el efecto transitorio del filtrado no son necesarias.
* **Análisis:** Se analizan las muestras correspondientes a *solape+ventana*. Los parámetros *solape* y *ventana* son elegidos por el usuario a través del panel de control. En esta etapa del procesado de datos se extraen los parámetros de interés de las muestras obtenidas. En este caso, se ha obtenido la potencia de las muestras que forman la ventana. Para ello, se eleva cada muestra al cuadrado, se suman y se dividen por el número de muestras, con esto se consigue realizar un promediado de la potencia.

Una ventana de muestras estará formada por las muestras procedentes de ventanas anteriores (*solape*), más una serie de muestras nuevas (*ventana*). La figura 5.4 muestra gráficamente el enventanado de las señales adquiridas.



*5.4: Superposición de secuencias analizadas.*

Para finalizar, comentar que se capturará una nueva ventana de muestras cada 31.25 mseg. que corresponden a una frecuencia de muestreo de 128 Hz y un tamaño de ventana de 64 muestras, de las cuales 60 son de *solape* y 4 son muestras nuevas (*ventana*). Por tanto, cada 128 Hz y 4 muestras () se debe realizar todo el procesado de datos y la posterior clasificación de los valores obtenidos.

### Clasificador

La clasificación establece un valor tras el cálculo de la potencia media de las muestras filtradas. Este valor obtenido se transfiere al mundo virtual, produciendo un desplazamiento a derecha o a izquierda del objeto biofeedback.

El clasificador que utiliza la BCI es de tipo lineal (*LDA, Linear Discriminant Analisis*) y viene caracterizado por la expresión:



donde:

* pot1 y pot2: Representan las potencias medias de los dos canales EEG registrados en un determinado intervalo. Esta potencia media se calcula directamente elevando al cuadrado cada muestra y haciendo un promedio de ellas.
* w0, w1 y w2: son constantes que actúan a modo de pesos, proporcionados por el clasificador tras una fase de entrenamiento del sujeto.

El valor *dist* obtenido es multiplicado por una serie de valores:

* Si , .
* Si , .

donde:

* dist\_der y dist\_izq: Determina en qué medida se desplaza el objeto biofeedback a la derecha y a la izquierda.
* Factor\_der y Factor\_izq: Se emplea para ajustar la potencia obtenida con el clasificador a las condiciones del entrenamiento.
* Cte\_der y Cte\_izq: Realizan un escalado al desplazamiento para ajustarlo a las dimensiones del mundo virtual implementado debido a que el desplazamiento del objeto biofeedback (coche) va de -3 a 3.

### Realimentación o biofeedback al sujeto.

El biofeedback se encarga de presentar al sujeto el resultado de su actividad mental. Para ello, se transfieren los valores *dist\_der* y *dist\_izq* obtenidos en la clasificación a la interfaz gráfica. En el sistema implementado, los cambios producidos se traducen en un desplazamiento a izquierda o derecha del vehículo, que hace de objeto biofeedback en el mundo virtual.

## Colisiones: sensores de proximidad

En un paso previo a la exportación al estándar VRML97 de los mundos virtuales de la Vivienda y de la Escuela hay que dotar a los objetos que los fabrican y decoran de la capacidad de producir eventos al aproximarse o colisionar con ellos.

Cuando un ambiente virtual contiene varios objetos con los que se debe interactuar, la detección de colisiones es uno de los problemas fundamentales, ya que si no se presta atención especial a la intersección entre los objetos, se podrían originar estados no deseados entre ellos.

Todos los elementos de los mundo virtuales modelados deben de poder interactuar con el usuario observador. El observador es capaz de introducirse en el mundo virtual a través de un objeto cámara o punto de vista (ViewPoint) a través del cual percibe la escena en la que se encuentra y éste debe ser capaz de interactuar con todos los elementos que se encuentra a su paso, desde las propias paredes que limitan la estructura básica hasta los elementos de decoración que se pueden interponer en su camino y con los puede colisionar.

Se deben evitar así situaciones inverosímiles en una situación real como los casos en los que el observador pueda por ejemplo atravesar paredes o situarse físicamente en el mismo espacio geométrico que ya esté ocupado por un elemento de decoración, columnas o incluso otros observadores.

De esta forma también se delimita el itinerario de navegación que se puede realizar a lo largo del mundo virtual, se puede prohibir por ejemplo entrar en una habitación o recinto o transcurrir por determinados lugares de la escena.

Para dotar a los objetos de un mundo virtual de estas características se utilizan los sensores de proximidad o *ProximitySensor*.

Estos sensores definen una región del espacio, normalmente en forma de caja, que permite detectar cuando un usuario ha entrado, salido o se mueve alrededor del interior de ella, reportando la localización y orientación del usuario dentro de la región.

Por tanto es necesario tanto para el mundo virtual de la Vivienda como de la Escuela rodear cada uno de los elementos que las componen de una región rectangular en forma de caja que disparen eventos de colisión cuando el punto de vista del usuario observador penetre en ellos. Esta regiones rectangulares se pueden fabricar directamente utilizando 3DStudio, como hemos hecho durante todo el modelado de los mundos de realidad virtual elaborados, y al final es tan sencillol como crear cajas con las dimensiones oportunas que envuelvan las geometrías con las cuales se quiere detectar la colisión.

Estas cajas se modelan con el objeto de 3DStudio *ProxSensor* ubicado en el menú de creación *Create/Helpers/VRML97* y tras la exportación a VRML se realiza su equivalencia en nodos *ProximitySensor.*

Como cualquier caja los parámetros que pueden ser configurados son su centro y dimensiones (alto, largo y ancho) además de un flag “*Enable*” que indica si el sensor esta activo o desactivo.

En el siguiente ejemplo se muestra como se modela, dimensiona y ubica un sensor de proximidad con 3DStudio para el elemento de decoración que da forma los tablones de notas en el mundo virtual de la Escuela de Telecomunicación, procedimiento que debe seguirse para cada uno de los elementos tanto de este mundo como para los de la Vivienda.

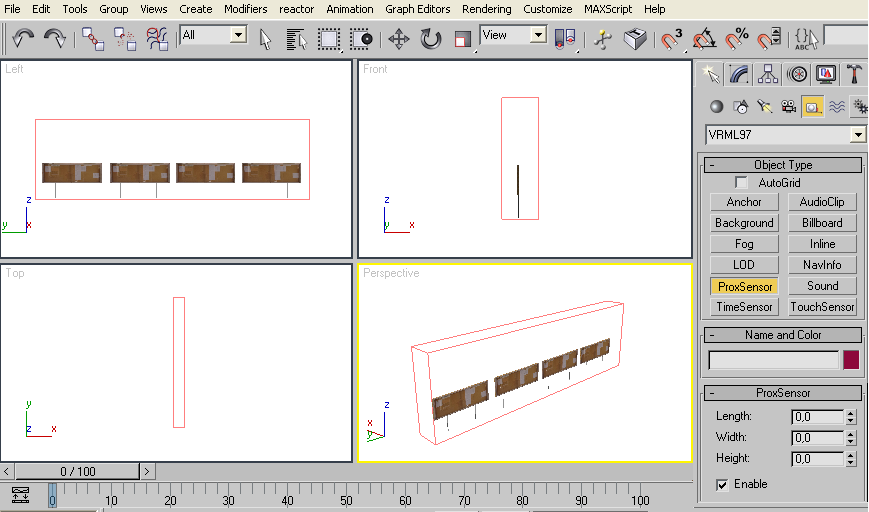


Figura 4‑1: Vistas Left, Front, Top y Perspective en la creación de un sensor.

Y la traducción del sensor anterior a VRML sería la siguiente:

DEF ProxSensorTablones Transform {

translation -9.53 0.005 61.4

children [

DEF ProxSensorTablones ProximitySensor {

enabled TRUE

center 0 2.86 0

size 0.747 5.72 8.56

}

]

}

## Conversión a VRML 97

Una vez que la escena ha sido creada y todo está emplazado en su lugar correspondiente, la escena está preparada para ser exportada a VRML.

Desde la versión 5, 3DStudio crea ficheros VRML (con extensión “.wrl”) que son compatibles con el estándar VRML97.

Cuando se realiza la exportación de un entorno virtual desarrollado en 3DStudio al lenguaje propio de VRML, se tiene la oportunidad de configurar un número importante de parámetros en el cuadro de diálogo *VRML 97 Exporter* que se encuentra en el menú *File/Export* de este programa y que se muestra a continuación:

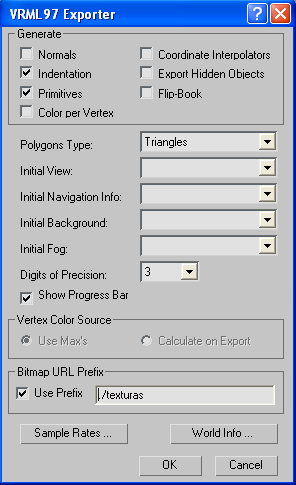


Figura ‑: Cuadro de diálogo VRML 97 Exporter

Los parámetros que se pueden configurar a la hora de realizar la exportación son:

* *Normals*: crea normales reales, creando fichero VRML más grandes. Esta casilla se selecciona si la geometría usada en el modelado necesita de suavizado o “*smoothing*”, y los visores de VRML utilizados para visualizar el mundo requieren de estas normales para la correcta visualización.

Los mundos virtuales del la Vivienda y de la Escuela no tienen geometrías que necesiten de esta propiedad, además se comprueba que con los visores VRML utilizados, es decir, Cortona 3D Viewer y Orbisnap (visor VRML de Matlab), se obtienen resultados muy óptimos sin marcar esta opción.

* *Indentation*: realiza el sangrado apropiado del fichero VRML para una mejor lectura del mismo.

Los ficheros VRML resultados de la exportación son ficheros de texto plano. Si no se realiza un sangrado adecuado del código resultado, es prácticamente imposible distinguir los objetos y parámetros de cada objeto en el fichero WRL, por tanto se marca esta opción para obtener un código más legible.

* *Primitives*: exporta las primitivas de VRML en lugar de las primitivas de 3DStudio. Se consiguen ficheros más cortos.

Es importante que el fichero resultado de la exportación tenga el menor contenido posible. De esta manera al visor VRML le será más fácil procesar el código y menor uso de CPU será necesario para visualizar el mundo virtual en tiempo real.

* *Color per Vertex*: exporta el color de los vértices de la geometría.

No es de especial relevancia que los vértices de las geometrías que componen el mundo virtual se exporten con un color concreto, ni con mucha precisión. Lo que sí interesa es que las caras de los polígonos que limitan esos vértices sí se exporten con el color correctamente.

* *Coordinate Interpolators*: habilita exportar animaciones que utilizan interpolación de coordenadas, no sólo las transformaciones simples de movimiento, rotación y escalado habituales. Se crean ficheros VRML más extensos.

Para los mundos virtuales creados donde la animación y movimiento de la escena se produce en fases posteriores a la exportación no es necesaria.

* *Export Hidden Objects*: si se marca la exportación incluye objetos que se hayan escondido, y serán visibles en VRML. Normalmente se deja desmarcada.
* *Flip-Book*: exporta la escena en múltiples ficheros como una tasa de muestreo especificada. Se utiliza para escenas con animaciones.
* *Polygon Types*: determina cómo son traducidas las caras de la geometría como nodos “*IndexedFaceSet*” de VRML. El tipo por defecto es “*Triangles*”, que traduce las caras como composiciones triangulares. “*Ngons*” traduce tantas caras como sea posible. “*Visible edges*” traduce, únicamente, las caras que son visibles.

A mayor complicación en la segmentación de las caras de la geometría, mayor es el tamaño del fichero resultado. Ninguno de los mundos elaborados tiene formas complejas y se ha selecciona el tipo “Triangles” tal y como viene por defecto.

* *Initial View*: lista todas las cámaras o puntos de vista de la escena y pone en primer plano la vista seleccionada inicialmente. Determina qué vista es la primera en experimentar el usuario.

En los mundos virtuales elaborados, previamente a la integración con el interfaz de navegación, no existe un punto de vista o cámara por lo que no tenemos ninguno que seleccionar. No obstante en las pruebas realizadas se sitúa un punto de vista de ensayo en ciertas ubicaciones desde las cuales un usuario observador pueda visualizar la escena. Así durante los ensayos realizados, cara a mostrar el efecto conseguido, se selecciona alguno de estos punto de vista como punto de vista inicial.

* *Digits of Precision*: Número de dígitos decimales usados para calcular las dimensiones. Por defecto es 4. Un número menos da como resultado un fichero VRML de menor tamaño, pero un número mayor puede ser necesario si partes de la escena se han situado 100.000 unidades, o más, alejadas del centro de la escena.

En ninguno de los mundos diseñados existen dimensiones superiores a las 500 unidades de separación de las geometrías al centro de coordenadas de la escena, tanto es así que es suficiente utilizar 3 dígitos de precisión para este parámetro con el fin de obtener un fichero wrl de pequeño tamaño.

* *Bitmap URL prefix*: especifica la localización de las texturas asignadas a los objetos de la escena. Si no se selecciona, los ficheros de imagen deben situarse en la misma ubicación del fichero VRML.

En este parámetro sólo hay que tener cuidado en colocar las imagen de textura que se han aplicado al mundo virtual en el directorio que se determine en este parámetro, para que al visualizar el mundo en el visor de VRML sepa dónde buscarlas.

El resto de parámetros no son de demasiado interés y no se detallan en este proyecto.

El resultado de este proceso es un fichero, con extensión “wrl”, de texto plano en el que es fácilmente distinguible cada uno de los nodos VRML que forman parte del mundo virtual. La correspondencia de un elemento en 3DStudio y su equivalente en un nodo VRML es usualmente 1 a 1, aunque puede que existan agrupaciones de elementos en 3DStudio que al ser exportados a VRML se obtengan en nodos independientes, o viceversa.

## Integración de Vivienda y ETSIT

## Integración de Simulador de Vuelo

El mundo virtual del Simulador de Vuelo se exporta a VRML siguiendo el procedimiento descrito para los mundos virtuales de la Vivienda y la Escuela de Telecomunicaciones que se detalla en el apartado 4.3.1 Exportación de 3DStudio a VRML.

Una de las pasos que incluye este mundo virtual, al igual que se ha realizado con los mundos modelados anteriormente, es la integración con el sistema interfaz de navegación existente y desarrollada por el Departamento de Tecnología Electrónica.

La integración con el interfaz, en los mundos virtuales descritos en los capítulos anteriores, era casi inmediata ya que lo único necesario era incluir el nodo VRML “silla de ruedas”. Los movimientos en el interior de la escena se delegan al sistema con el que se realiza la integración, que finalmente es capaz de proporcionar órdenes a la “silla”.

Es decir, en esos casos la experiencia de navegación a través de la escena virtual, bien la Vivienda Virtual o la ETSIT Virtual, se realizaba aportando movimiento al nodo “Silla” que realiza las funciones de elemento *feedback* obedeciendo y plasmanado las ordenes del sistema BCI realizando los desplazamientos y giros interpretados tras el procesamiento del sistema de captura de señales EEG.

Los elementos 3D que componen el mundo virtual, tanto en el caso de la Vivienda como en el de la ETSIT, permanecen siempre inmóviles, siendo el punto de vista, adosado al nodo VRML “Silla” el que emula la sensación de movimiento en la escena.

En el caso del Simulador de Vuelo, la integración con el interfaz de navegación es algo más compleja, debido a que son los propios elementos de la escena, y a través de modificaciones sobre los mismos, los que hacen que la escena tome vida y movimiento.

La integración con el sistema de navegación existente necesita de dos etapas de acondicionamiento fundamentales:

1. Integración con el interfaz de navegación con el que se integra el mundo virtual, y que lleva adosado el punto de vista del espectador.
2. Implementación de algoritmos y métodos en código Matlab, para realizar los movimientos sobre los elementos 3D que componen el Simulador de Vuelo, es decir, sobre el modelo del avión y el plano de vuelo.

### Integración con el interfaz de navegación

El interfaz de navegación existente y con el que se integra el mundo virtual del Simulador de Vuelo consta de tres elementos que se agrupan bajo un mismo elemento o nodo VRML, los tres elementos son:

* Punto de vista o cámara a través de la cual el usuario percibe el mundo virtual.
* Geometría que modela las áreas de selección de movimiento de navegación: se puede realizar la toma de cada una las siguientes opciones de movimiento:
  + Ascensos: el avión toma altura sobre el relieve de la ciudad.
  + Giros a la derecha del avión.
  + Descensos: O lo que es lo mismo, pérdida de altura de vuelo.
  + Giros a la izquierda del avión.
* Barra indicadora del nivel de concentración del sujeto

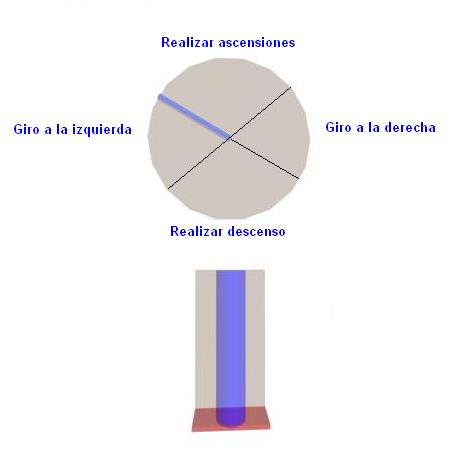


Figura ‑: Interfaz gráfico (VRML) de navegación

En el proceso de selección de una opción una flecha va recorriendo, en sentido horario, las cuatro áreas que componen el círculo de selección de opción, mientras la barra de concentración indica el nivel de concentración en la toma de cada una de las opciones. En el momento en que el individuo alcanza un umbral de concentración determinado, el entorno virtual lo interpreta y realiza las modificaciones necesarias en sus elementos 3D para simular el movimiento que determina la opción elegida.

Una vez que disponemos del mundo virtual del Simulador de Vuelo bajo el estándar VRML, para integrar el interfaz de navegación es necesario incluir en el código fuente VRML del Simulador de Vuelo el nodo VRML que modela el interfaz de navegación. Esto se lleva a cabo editando el fichero VRML del Simulador de Vuelo en un editor de texto cualquiera e incluyendo el nodo VRML que diseña el interfaz de navegación anteriormente descrito.

Si se ejecuta el Simulador de Vuelo con el visor de entornos VRML Cortona3D Viewer se puede observar el resultado tras la integración con el interfaz de navegación existente.



Figura ‑: Integración. Visualización desde el punto de vista

### Implementación del movimiento

A través del interfaz se escoge una de las cuatro opciones de navegación que deben traducirse a los correspondientes desplazamientos de los elementos del mundo virtual para producir el movimiento de la escena.

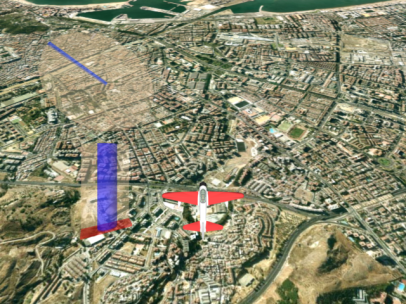
En el caso del Simulador de Vuelo al interpretar una opción de navegación, el movimiento de la escena no se consigue simulando los desplazamientos sobre el punto de vista, sino que son los propios elementos de la escena los que se mueven, mientras el punto de vista debe permanecer en todo momento inmóvil. De esta manera se describe a continuación cómo se ha implementado cada una de las cuatro opciones de navegación que soporta el interfaz con el que se ha realizado la integración.

Antes de nada hay que tener en cuenta que el avión nunca puede estar parado en el aire ya que en caso contrario la tripulación sufriría un grave accidente. Esta circunstancia obliga a que aunque no se haya tomado alguna de las opciones de navegación, el avión debe seguir avanzando, sobrevolando el terreno en la dirección actual y en línea recta. Lo primero que se debe explicar es cómo se recrea esta situación con los elementos de que disponemos.

Como se explicó en el epígrafe 5.4.2.2 Texturización del plano de vuelo, la recreación de la sensación del avión sobrevolando el terreno, no se realiza mediante un desplazamiento de avance del modelo 3D del avión sobre el terreno, ni desplazamientos del punto de vista del observador, sino que es el propio terreno (textura del plano) el que se desplaza, permaneciendo en todo momento el modelo del avión y el punto de vista inmóviles en sus coordenadas originales.

1. **Avance continuo del avión**

Se consigue realizando el desplazamiento de la textura, más bien del *Gizmo* que la representa, en dirección –Y del eje de coordenadas. Esta dirección en realidad es contraria al avance natural que tendrían que realizar el avión y el punto de vista, pero la sensación final es la misma, es decir, parece que es el avión es el que está avanzando, y en realidad lo que avanza es el terreno pero en sentido contrario como se muestran en las siguiente secuencia.

Los cambios necesarios a nivel de implementación de código Matlab han requerido que en las líneas de código en las que se realizaban movimientos del punto de vista en el Interfaz existente, ahora lo que se haga sea el desplazamiento de la textura en la dirección que se ha detallado. Se ha implementado un método Matlab, llamado “*goAhead*”, para llevarlo a cabo y este método se ejecuta en todo momento siempre y cuando no se haya tomado alguna de las opciones de navegación. El código necesario para esta implementación se muestra en el ANEXO XXX.

1. **Giro a la derecha o izquierda**

Estas opciones de navegación se simulan realizando movimientos diagonales de la textura en la dirección contraria al movimiento natural que se pretende conseguir. Además en estos casos, el propio modelo del avión también se modifica, dotando de un movimiento de rotación a su eje longitudinal en la dirección del giro que se haya seleccionado. Es decir, el avión sigue literalmente inmóvil en la escena, permaneciendo en las coordenadas originales, pero se rota su eje hacia la derecha o izquierda según se haya elegido un giro u otro, dotando a la escena de mayor realismo a la hora de recrear cada uno de los movimientos. En realidad la textura avanza y el avión rota y la sensación se aproxima a un giro y avance natural tal y como lo haría un avión real.

La siguiente secuencia de imágenes plasma el resultado de un giro a la derecha.

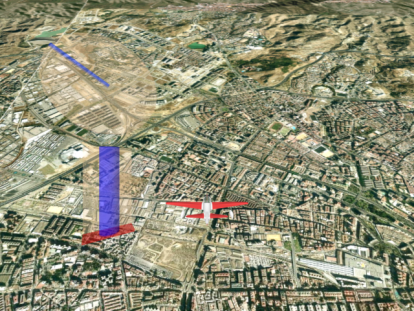
1. **Ascensos y descensos**

En este caso la textura debe de seguir avanzando tal y como se describe en el punto 1, pero además entra en juego la ubicación del plano de vuelo al que se le aplica la textura. Si se desplaza paulatinamente el plano en dirección –Y, lo que se está haciendo es alejar el plano del avión, por tanto se experimenta la sensación de ascender el vuelo y alejarnos del terreno.

Si, por el contrario, se desplaza el plano en dirección Y, se está acercando a la posición donde permanentemente se localiza el avión y punto de vista, y se consigue así la sensación de estar descendiendo.

Si además se acompañan estos desplazamientos del plano texturizado con la rotación del eje transversal del avión simulando que levanta unos grados el morro en los ascensos y lo baja en los descensos, se consigue así la experiencia completa de estar realmente efectuando cada uno de los movimientos.

Se puede observar el efecto conseguido en un descenso en la siguiente secuencia de imágenes.

# PRUEBAS Y EVALUACIÓN

CONCLUSIONES Y LINEAS FUTURAS