UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

I.T. TELECOMUNICACIÓN (SONIDO E IMAGEN)

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

"Diseño de una interfaz gráfica para simulación en Acústica de Salas"

TRABAJO FINAL DE CARRERA

Autor/es:
Eugenio Garcés Leante

Director/es:
Francisco Javier Redondo Pastor
Rubén Picó Vila

GANDIA, 2011

ÍNDICE.

Introducción.

1. **La Acústica de Salas.**

1.1 Introducción a la **Acústica de Salas**.

1.2 Parámetros acústicos básicos.

1.3 Simulación en **Acústica de Salas**.

1.3.1 Métodos de predicción.

1.3.1.1 Programas de simulación.

1.3.1.2 Pruebas con modelos a escala.

1.3.1.3 Auralización.

1.4 Técnica de simulación ***Finite difference time domain (FDTD)***.

1.4.1 Explicación teórica del método.

1.4.2 Simplificación para dos dimensiones.

1.4.3 Aproximación por diferencias finitas.

2. **Justificación del proyecto.**

2.1 Objetivo principal del proyecto.

2.2 Inconvenientes del ***Vi_Ac 1.0***.

2.3 Solución de los inconvenientes.

2.4 Innovaciones del nuevo programa.

3. Resultados.

3.1 Desarrollo del nuevo programa.

3.1.1 Primeros pasos.

3.1.2 Datos de salida e información sobre elementos del programa.

3.1.3 Nuevas funciones y herramientas.

3.1.3.1 Función ***Opciones.***

3.1.3.2 Archivos *.WAV como señal impulso.

3.1.3.3 Función ***design2.***

3.1.3.4 Diseños mediante coordenadas.

3.1.3.5 Tiempo de fuente activa.

3.1.3.6 Simulación con varias fuentes.

3.1.3.7 Función ***ayuda.***

3.1.3.8 Función ***viac2.***

3.1.3.9 Banco de sonidos.

3.1.4 Generar archivo ejecutable.

3.2 Resumen de resultados.

4. Conclusión.

5. Bibliografía.

6. Anexos

6.1 Máxima memoria virtual usada por **MATLAB**.

6.2 Capturas de pantalla.

6.3 Diseño del código.

6.4 Código.

Introducción.

En este proyecto se desarrolla una interfaz gráfica de simulación que permite estudiar, de manera gráfica y acústica, la influencia de los diferentes elementos arquitectónicos sobre la propagación del sonido en diseños gráficos de dos dimensiones. La aplicación tiene el nombre de "*Virtual Room Acoustics 2*", está desarrollada en **MATLAB** y es la continuación del proyecto final de carrera "*Diseño de una interfaz gráfica para la simulación mediante Diferencias Finitas*" del cual surgió el programa "*Virtual Room Acoustics 1.0*".

Lejos de ser una herramienta profesional, lo que pretende "*Virtual Room Acoustics 2*" es crear una idea sencilla y clara al usuario de como se propagan las ondas sonoras y de que manera influyen los elementos acústicos usados habitualmente en la **Acústica de Salas** en dicha propagación.
El enfoque dado para este proyecto es el de crear un programa útil a la hora de asentar conocimientos relacionados con la **Acústica** tales como: propagación, sonido reflejado, reverberación, eco, absorción, respuesta en frecuencia, etc... Es decir, crear una herramienta didáctica ya sea por su utilización, o por su futuro desarrollo.

El *Virtual Room Acoustics 1.0* (a partir de ahora *Vi_Ac 1.0*) es una interfaz gráfica que utiliza el método de las **Diferencias Finitas** para simular la evolución del sonido dentro de un recinto. Mediante este método se representa la solución discreta de las ecuaciones fundamentales de primer orden para unas condiciones de contorno. Fue propuesto para estudiar fenómenos de *scattering* para ondas electromagnéticas y es útil ya que permite representar las simulaciones en formato de video.

Como ya se ha dicho anteriormente, *Vi_Ac 1.0* es una interfaz gráfica, está programada en **MATLAB** y está distribuida en varios archivos tipo ***p,** cada uno de los cuales dispone de las funciones principales del programa. Para arrancar la aplicación es necesario llamar a un archivo (llamado *start.p*) desde **MATLAB.**

Los datos de entrada para el programa son los diseños que se realizan con éste mismo, después de una simulación, se obtienen los datos de salida que son todos en modo video ya que se muestra en pantalla la evolución del sonido a través del recinto y, las gráficas con el impulso y la respuesta al impulso a lo largo del tiempo (todo ello en dos dimensiones).

En el diseño de los recintos, permite la elección de los materiales a usar ya que se dispone de una base de datos (ampliable) con diferentes materiales. También se puede seleccionar el tamaño del recinto así como el rango de frecuencias.
A la hora de simular, son varias las opciones que propone: número de oyentes, tipo de fuente y tiempo de duración de la simulación.

Estos son los archivos de los que se compone el programa:

- ***design.p:*** es la función que crea la pantalla para diseñar proyectos.
- ***new_materials.p:*** mediante esta función se pueden agregar materiales a la lista.
- ***materials_list.mat:*** contiene la lista de materiales.
- ***remove_materials.p*:*** nos permite eliminar materiales de la lista.
- ***restore_materials.p:*** vuelve a la lista a su estado por defecto.
- ***simulation.p:*** arranca la simulación del proyecto diseñado.

- ***start.p:*** esta función crea la primera pantalla de inicio y es la que sirve para llamar a todas las demás funciones.

- *En un momento del desarrollo del programa,el archivo y su función "remove_materials.p", pasa a llamarse "materials.p". Por este motivo "remove_materials" es a partir de ahora "materials".*

El funcionamiento es bastante sencillo, al ejecutar el programa aparece una primera pantalla con las opciones siguientes:

- Introducir nuevos materiales: permite introducir a la base de datos del programa nuevos materiales para la creación de recintos y luego su posterior simulación. Los datos de entrada para nuevos materiales son: nombre del material, coeficiente de absorción para 125, 250, 500, 1000, 2000 y 4000 Hz y color con el que aparecerá en la pantalla de diseño.

- Resetear lista de materiales: por defecto se manejan una serie de materiales que están guardados en un archivo tipo **.mat* llamado *materials_list.mat*; en este archivo se guardan los materiales que va agregando el usuario. Con esta función lo que hace el programa es borrar todos los elementos introducidos posteriormente por el usuario y dejar la lista por defecto.

- Borrar materiales: borra de la lista de materiales, el material que el usuario elija.

- Diseñar recintos: se abre la pantalla de diseño donde el usuario puede diseñar en dos dimensiones un recinto con diferentes materiales para, después, poder simularlo.
  El programa permite dibujar tanto lineas rectas como lineas curvas o circunferencias.
  Las variables de entrada son: el rango de frecuencias a simular y el tamaño de la figura tipo *axes* donde se realiza el diseño. Una vez hecho el diseño, éste se guarda en un archivo tipo **.mat* en el lugar y con el nombre que se especifique.

- Simulación: el programa carga el diseño guardado y propone varias opciones de simulación. Éstas son: el número de oyentes o receptores, el tiempo de simulación y el tipo de fuente que genera la onda sonora con la frecuencia se quiere generar; puede ser un tono puro o un impulso. Una vez introducidos estos datos la simulación comienza mostrando en pantalla la evolución del sonido en el recinto. Además se van generando unas gráficas donde se puede observar el tono o impulso generado y la respuesta que genera el diseño sobre el oyente.

A parte del archivo *materials_list.mat,* la interfaz también maneja un archivo llamado *grosor.mat.* Este archivo es utilizado por algunas funciones ya que en él se guardan datos sobre la resolución del *axes* del diseño, ésta depende del intervalo de frecuencias seleccionado.

1. La Acústica de Salas.

1.1 Introducción a la Acústica de Salas.

La **Acústica de Salas** está englobada dentro de lo que se denomina la **Acústica Arquitectónica,** que es la disciplina que estudia la influencia de los diferentes elementos arquitectónicos sobre la transmisión del sonido. Por lo tanto la **Acústica de Salas** estudia la influencia de dichos elementos sobre la transmisión del sonido en salas musicales y/o de audición verbal.
Comenzó a mediados del siglo XIX cuando el arquitecto francés *Lachez* publicó el tratado *"Acoustique et optique des salles de reunion"*. Más tarde, *Lord Rayleigh* introdujo el tratamiento ondulatorio en el estudio acústico de salas (publicó en 1.887 un tratado con el título de "*Theory of sound*").

A través de extensos estudios experimentales de las propiedades acústicas de un recinto, *Wallace Clement Sabine* a principios del siglo XX, llegó a una relación empírica entre las características de reverberación de un recinto, su tamaño y la cantidad de material absorbente.

$$T \propto V/A$$

Su definición del tiempo de reverberación como el tiempo requerido para que la presión sonora disminuya 60 dB, supone un importante parámetro acústico fácilmente medible. La ecuación de *Sabine* relaciona el tiempo de reverberación de un recinto a su volumen y un parámetro acústico que especifica su absorción sonora total.
Todas las derivaciones teóricas de esta ecuación por lo general se basan en un modelo de rayos en el cual se supone que el sonido sale de la fuente como rayos divergentes. Cuando los rayos llegan a las fronteras del recinto, éstos son absorbidos y reflejados. Después de un gran número de reflexiones se puede suponer que la densidad promedio de energía es la misma en todo el recinto, y todas las direcciones de propagación son igualmente posibles (sonido difuso). Este hecho simplificaría mucho el comportamiento real del sonido en un recinto.

La acústica se consolidó como una nueva ciencia a partir de los años 30 fundamentalmente por el desarrollo de la tecnología de micrófonos, amplificadores y altavoces, y su utilización como herramienta en trabajos de campo. Posteriormente, con la evolución de los equipos electrónicos, ha sido posible relacionar parámetros subjetivos (claridad musical, intimidad,etc...) con otros parámetros objetivos obtenidos directamente de mediciones *"in situ"*.
En la década de los 80 y sobre todo de los 90, aparecen los primeros programas de predicción acústica con los que se pueden realizar simulaciones de recintos sin tener que recurrir al uso de maquetas. Finalmente y unido a los programas de predicción acústica, aparecen los denominados sistemas de creación de sonido virtual (auralización).
En la actualidad, los programas de simulación junto con la auralización, facilitan significativamente el diseño de nuevos recintos, haciendo que las fase de diseño (en recintos en fase de construcción) y de análisis (en recintos ya construidos) sean menos costosas.

1.2 Parámetros acústicos básicos.

Existen varios parámetros acústicos independientes a partir de los cuales es posible representar la calidad acústica de una sala. He aquí una definición de cada uno de ellos:

Parámetros acústicos de salas para la palabra:

- **%Alcons (sala ocupada)**: del inglés *"Articulation Loss of Consonants"* mide el grado de pérdida de las consonantes en la transmisión del mensaje. La comprensión de un mensaje oral depende fundamentalmente de la correcta percepción de sus consonantes.

- **STI/RASTI (sala ocupada)**: *"Speech Transmission Index"*, permite cuantificar el grado de inteligibilidad de la palabra entre los valores 0 (inteligibilidad nula) y 1 (inteligibilidad óptima). El índice *RASTI ("Rapid Speech Transmission Index"),* es una versión simplificada del *STI*.

- **Sonoridad media (sala ocupada)**: indica el grado de amplificación que produce una sala sobre el mensaje oral emitido.

- **Tiempo de reverberación medio (sala ocupada):** es indicativo del grado de reverberación o "viveza" de una sala. Si el valor del tiempo de reverberación es elevado la sala resulta excesivamente "viva", haciéndola más adecuada para música.
  Es conveniente que este valor se mantenga constante con la frecuencia, ya que si aumenta para las bajas frecuencias empeora el grado de inteligibilidad de la palabra.

- **Claridad de la voz (sala ocupada)**: se define como la relación entre la energía sonora que llega al oyente durante los primeros 50 mS desde la llegada del sonido directo y la que llega después de los primeros 50 mS. Cuanto más elevado sea este valor, mejor será la inteligibilidad de la palabra.

- **Definición (sala ocupada)**:es la relación entre la energía que llega al oyente dentro de los primeros 50 mS desde la llegada del sonido directo y la energía total recibida por éste. Cuanto más elevado sea este valor, mejor será la inteligibilidad de la palabra.

- **Relación de primeras reflexiones (sala vacía u ocupada)**: se define como la relación entre la energía que llega al oyente dentro de los primeros 50 mS desde la llegada del sonido directo y la energía correspondiente al sonido directo. Indica el número de primeras reflexiones existentes en un punto determinado de la sala.

Parámetros acústicos de salas para la música:

- **Tiempo de reverberación (sala ocupada)**: indica el grado de "viveza" de la sala.

- **Calidez acústica y brillo (sala ocupada)**: determina si la sala posee riqueza en sonidos graves y a su vez éste es claro y rico en armónicos.

- **"Early Decay Time" (EDT), sala ocupada**: se define como seis veces el tiempo que transcurre desde que la fuente emisora deja de radiar hasta que el nivel de presión sonora cae 10 dB. Como el tiempo de reverberación, también es indicativo del grado de "viveza " de la sala.

- **Sonoridad (sala vacía)**: representa el grado de amplificación producido por la sala.

- **"Initial-Time-Delay Gap"**: representa la impresión subjetiva de "intimidad" acústica (grado de identificación con la orquesta, es decir si el oyente se encuentra inmerso o distante de la música que está escuchando).

- **Claridad musical**: indica el grado de separación entre los diferentes sonidos individuales integrantes de una composición musical.

- **Eficiencia lateral**: define la impresión espacial del sonido, es decir, la amplitud aparente de la fuente sonora.

- **Correlación cruzada interaural**: se define como la correlación entre los sonidos que llegan a ambos oídos, y es indicativa del grado de similitud existente entre las dos señales. También representa la amplitud aparente de la fuente sonora.

- **Índice de difusión (SDI), "Surface Difussivity Index"**: cuanto mayor sea este valor, mayor será la impresión espacial de la sala (sensación de envolvente).

- **Soporte objetivo medio (sala vacía y escenario sin músicos pero con los elementos que le son propios como sillas, atriles,etc..):** es la capacidad que tienen los músicos en una determinada sala de escucharse a si mismos y al resto de la orquesta.

1.3 Simulación en Acústica de Salas.

La simulación en **Acústica de Salas** es un método que se utiliza para poder predecir el comportamiento de una determinada sala antes, incluso, de su construcción para así:

- Efectuar un diagnóstico o predicción del comportamiento acústico del recinto basado en la interpretación de los valores de los parámetros medidos (tiempo de reverberación, EDT, etc...).

- Sugerir posibles actuaciones con el fin de mejorar las condiciones acústicas (optimización del diseño).

- Detectar errores en la construcción que puedan influir de manera negativa (ecos, focalizaciones, etc...).

- Auralización.

1.3.1 Métodos de predicción.

1.3.1.1 Programas de simulación.

Los primeros programas de simulación acústica aparecieron a principios de los 80 pero eran muy limitados a lo que prestaciones se refiere debido, entre otras cosas, a la reducida capacidad y velocidad de los ordenadores de aquella época. Entonces los resultados obtenidos por este tipo de programas eran útiles sólo a título orientativo.
El desarrollo de los ordenadores ha supuesto que estas herramientas evolucionen de tal forma, que hallan permitido una notable reducción de costes en los proyectos de acondicionamiento acústico respecto a otros que se realizan mediante métodos de predicción acústica, como los modelos a escala.
Mediante su utilización es posible calcular, con un grado de aproximación elevado y de forma mucho más rápida, todos aquellos parámetros que representan la calidad acústica de una sala.

Entre los métodos más usados para la simulación acústica se encuentran los siguientes:

- Trazado de rayos: basado en la ***Teoría Geométrica***, tiene como principal fundamento el ***Principio de Fermat***. Esta teoría proporciona una descripción simplificada del campo sonoro en el interior del recinto. Se utiliza para frecuencias altas ya que el modelo es tanto más válido si las dimensiones de la sala son grandes comparadas con la longitud de onda del sonido.
  Utiliza el concepto de rayo sonoro que consiste en una fracción de onda esférica de abertura despreciable con una dirección de propagación muy definida.
  Cuanto mayor sea el número de rayos de la fuente, mayor es la precisión que ofrece esta técnica. Sin embargo un mayor número de rayos conlleva un aumento de la complejidad de los cálculos.
  Se pueden distinguir tres clases de rayos sonoros:

  - Axiales: se propagan paralelamente a los ejes de coordenadas del recinto.
  - **Tangenciales**: se propagan paralelos a los planos de coordenadas.
  - **Oblicuos**: no guardan ningún paralelismo ni con los ejes, ni con los planos de coordenadas.

- Método de los elementos de contorno (BEM): proporciona enorme exactitud en los cálculos aunque requiere largos y costosos procesos en lo que a código se refiere, sobretodo para frecuencias altas y superficies de gran tamaño.

- Análisis de los elementos finitos (FEA): es un método que proporciona predicciones de *scattering* con mucha exactitud. Es más pesado en cuanto a código que el ***BEM,*** lo que hace que sea el método menos eficiente.

- Método de diferencias finitas (FDTD): este método se explica con más detalle en el punto ***1.4*** de este documento.

Según el método que utilicen para la simulación, estos programas se pueden dividir en tres tipos:

- **Numéricos**: emplean las técnicas de, por ejemplo, ***BEM***, ***FEM*** ,***FDTD***, etc...

- **Geométricos**: se basan en la ***Teoría Geométrica*** y utilizan los métodos de rayos y de imágenes virtuales.

- **Estadísticos**: se basan en la ***Teoría Estadística***.

Por otra parte, las técnicas de procesado binaural junto con los algoritmos de predicción acústica, han hecho que muchos de los programas dedicados a la predicción acústica, dispongan de un módulo de auralización que pueda funcionar con la tarjeta de audio de un ordenador o con un módulo externo.

Para poder hacer una simulación mediante este tipo de software, se tendrá que disponer del modelo arquitectónico de la sala (ya sea importado o diseñado por el propio programa) y de la información relativa a las características de absorción y difusión de los materiales constructivos utilizados.
Una vez diseñado el modelo arquitectónico y asignados los materiales correspondientes a las diferentes superficies, el programá calculará los parámetros acústicos representativos para los que ha sido diseñado (ecogramas, mapas de nivel, etc...). Si éstos se encuentran dentro de los márgenes de valores recomendados (validación objetiva) el resultado será positivo y el diseño habrá terminado. De lo contrario se tendrán que tomar las medidas oportunas para que los resultados se ajusten a los objetivos.

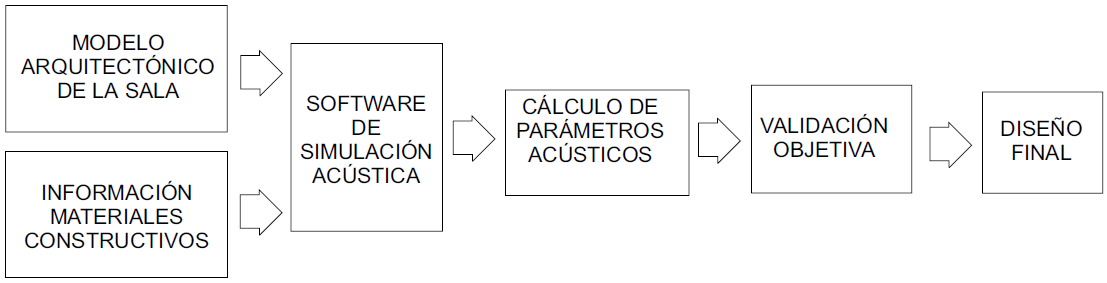

Figura 1: Proceso a seguir para realizar una simulación mediante software.

Las limitaciones de este tipo de programas vienen dadas por los errores que generan al realizar las simulaciones. Estos errores de simulación son debidos a que:

- Se presupone un campo sonoro difuso en el que el coeficiente de absorción no coincide con el coeficiente de absorción de un caso real (campo sonoro no difuso).
- A la hora de hacer los diseños de los recintos, el propio programa de diseño aplica una serie de simplificaciones geométricas que influyen en los resultados de las simulaciones.
- Los métodos de simulación de *scattering* y difracción son técnicas que, aunque alcancen un nivel de precisión elevado, no son exactas y distan en mayor o menor medida de un caso real.

1.3.1.2 Pruebas con modelos a escala.

Con este método se trata de construir un modelo a escala del recinto que se quiere simular y realizar los diferentes cálculos sobre dicho modelo para su posterior extrapolación para el modelo real, de esta manera se pueden obtener resultados orientativos en cuanto al comportamiento acústico del recinto.
El espacio requerido para la fabricación y ubicación de estas maquetas, el coste y el tiempo de construcción (además de la aparición del software específico), han hecho que actualmente estén en desuso.

1.3.1.3 Auralización.

La auralización consiste en realizar una escucha en cualquier punto de un recinto, de un mensaje oral o un pasaje musical, de manera totalmente virtual, es decir, de manera simulada. Esto permite comprobar de manera auditiva la calidad acústica de la sala aunque el recinto o sala en cuestión no esté construido físicamente. También es útil si el recinto ya está construido pero hay que introducirle cambios para la mejora de la acústica.

Existen varios tipos de auralización:

- Auralización monoaural por convolución : este sistema calcula la señal auralizada y(t) por convolución de la respuesta al impulso h(t), con cualquier señal grabada en ambiente anecoico x(t). Para generar la respuesta al impulso se utiliza un software de simulación acústica; para la convolución se utiliza un procesador digital de señal (DSP).

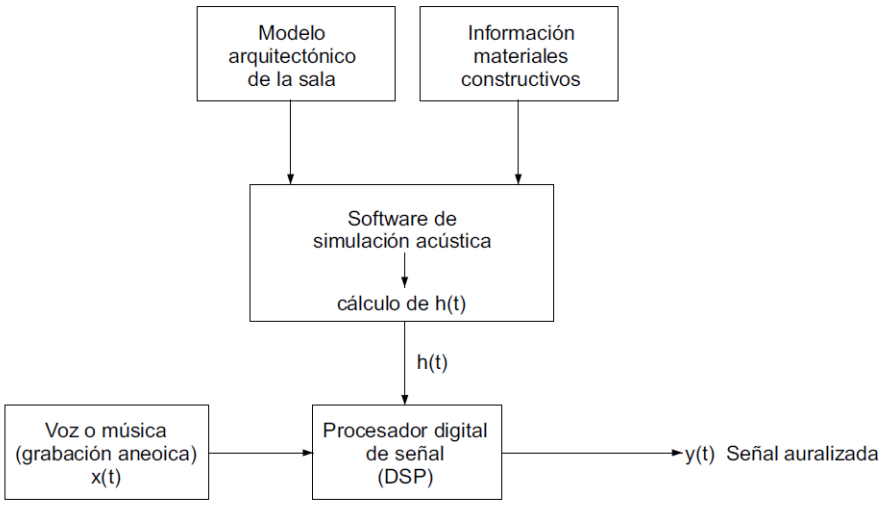

Figura 2: Esquema de la auralización monoaural por convolución.

– Auralización binaural por convolución: en este caso la señal auralizada tiene dos componentes (canal derecho y canal izquierdo), por lo que la señal de respuesta al impulso también tendrá las mismas dos componentes.

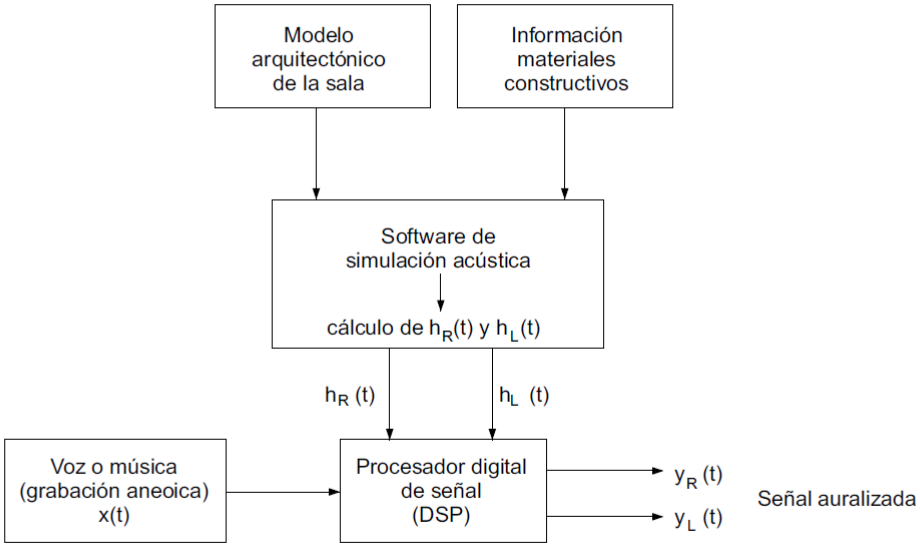

Figura 3: Esquema de la auralización binaural por convolución.

– Auralización mediante unidad de reverberación digital: en este método la respuesta al impulso se calcula de igual manera que en los casos anteriores, es decir con un software de simulación acústica.
A partir de dichas respuestas, el programa calcula una serie de parámetros acústicos representativos en el punto de estudio. Los cálculos obtenidos son utilizados para programar los parámetros internos de una unidad de reverberación digital.
Una vez programada se le aplica a la entrada la señal de excitación x(t), obteniendo a la salida la señal auralizada. Esta auralización también puede ser monoaural o binaural.

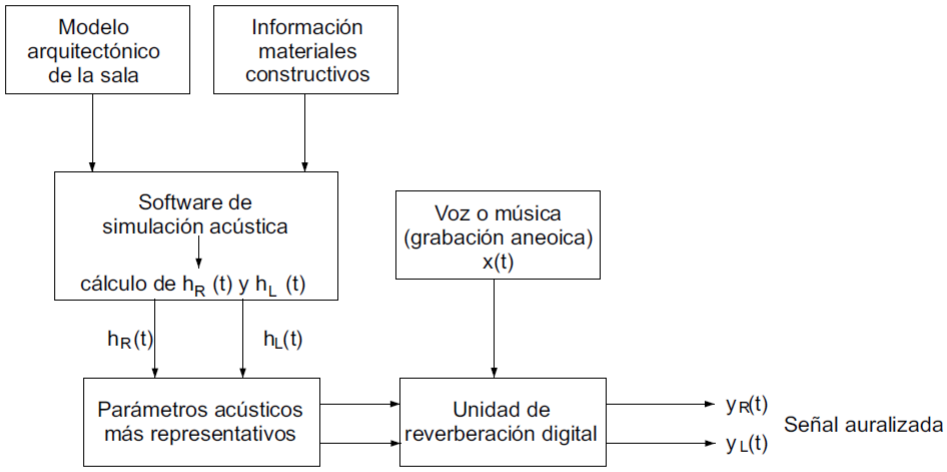

Figura 4: Esquema de la auralización mediante unidad de reverberación digital

1.4 Técnica de simulación *Finite difference time domain (FDTD).*

1.4.1 Explicación teórica del método.

La técnica para la simulación *Finite difference time domain* o *Método de las diferencias finitas,* fue propuesta por *Yee* en 1966 con el propósito de estudiar fenómenos de *scattering* para ondas electromagnéticas.
Su principal ventaja radica en que es una técnica de fácil comprensión y de implementación, cubriendo sin embargo, un gran ancho de banda con predicciones exactas de *scattering.* Esto hace que cada usuario pueda adaptar su código con suma facilidad.
Existen otros métodos para la predicción de *scattering* pero son mucho más complejos que el método de las diferencias finitas en lo que se refiere a código.
Uno de los puntos débiles del método *FDTD,* es que todos los puntos tienen que estar incluidos en el diseño, lo que hace que aumente el coste en lo que a código se refiere. Para subsanar esto, se usa *Near Field to Far Field Transformations (NFFFT)* basado en un teorema de equivalencia de contorno.
Más adelante se adaptó el método para la acústica basándose en las ecuaciones de conservación del momento y de continuidad:

$$\frac{\partial p}{\partial t} + \frac{\left(\vec{\nabla}\boldsymbol{u}\right)}{\mathrm{K}_e} = 0$$

$$\vec{\nabla} p + \rho_0 \frac{\partial \boldsymbol{u}}{\partial t} = 0$$

Donde $\boldsymbol{p}$ es la presión sonora, $\boldsymbol{u}$ es el vector velocidad y $\boldsymbol{\rho}$ es la densidad del medio.

1.4.2 Simplificación para dos dimensiones.

Simplificando para el método en dos dimensiones, las anteriores ecuaciones se pueden escribir como sigue:

$$\frac{\partial p}{\partial t} + \frac{1}{\mathrm{K}_e}\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y}\right) = 0$$

$$\frac{\partial p}{\partial x} + \rho_0 \frac{\partial u_x}{\partial t} = 0$$

$$\frac{\partial p}{\partial y} + \rho_0 \frac{\partial u_y}{\partial t} = 0$$

Llevando los cálculos numéricos a un diseño de dos dimensiones, se pueden representar la velocidad de las partículas y la presión sonora de la siguiente manera:

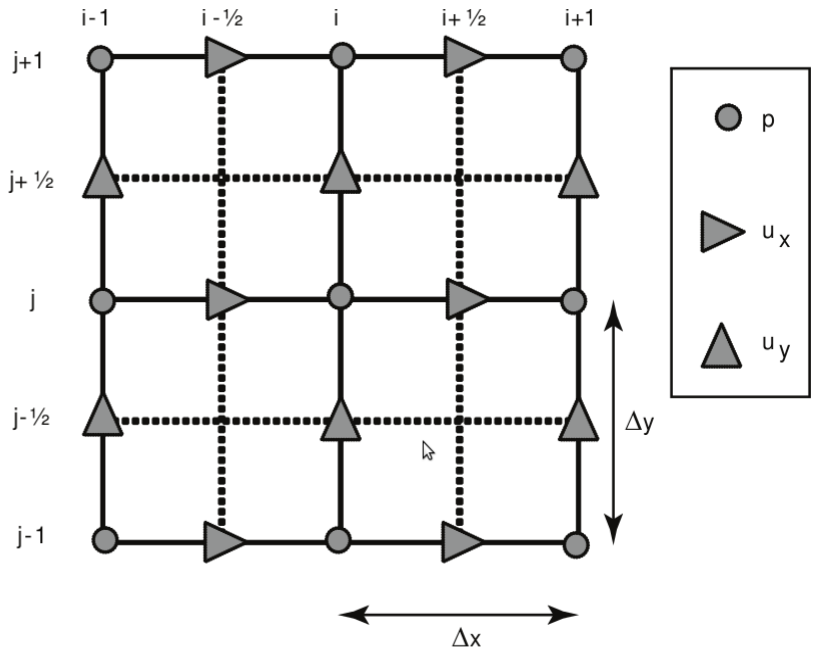

Figura 5: Representación en dos dimensiones de la velocidad de las partículas y presión sonora.

Presión y velocidad vienen dadas por las siguientes ecuaciones:

$$p_{i,j}^{n+\frac{1}{2}} = p(i\,\Delta x, j\,\Delta y, (n+\tfrac{1}{2})\;\;\Delta t)$$

$$ux_{i+\frac{1}{2},j}^{n} = u_x((i+\tfrac{1}{2})\,\Delta x, j\,\Delta y, n\,\Delta t)$$

$$uy_{i,j+\frac{1}{2}}^{n} = u_y(i\,\Delta x, (j+\tfrac{1}{2})\,\Delta y, n\,\Delta t)$$

Donde Δx y Δy son los intervalos espaciales, x e y la dirección de la partícula y, Δt es el tiempo entre cálculos.

1.4.3 Aproximación por diferencias finitas.

Teniendo en cuenta que las derivadas de espacio y tiempo de la presión, y la velocidad de la partícula pueden aproximarse por ecuaciones de diferencias finitas; la derivada de la presión con respecto ***x*** queda de la siguiente manera:

$$\left.\frac{\partial p}{\partial x}\right|_{x=x_0} \approx \frac{p\left(x_0+\frac{\Delta x}{2}\right)-p\left(x_0-\frac{\Delta x}{2}\right)}{\Delta x}+O\left[(\Delta x)^2\right]$$

Usando esta aproximación, se puede concluir que las ecuaciones anteriores de presión y velocidad se pueden escribir así:

$$p_{i,j}^{n+\frac{1}{2}} = p_{i,j}^{n-\frac{1}{2}} - \frac{1}{\mathrm{K}_e}\Delta t\left(\frac{ux_{i+\frac{1}{2},j}^{n}-ux_{i-\frac{1}{2},j}^{n}}{\Delta x}+\frac{uy_{i,j+\frac{1}{2}}^{n}-uy_{i,j-\frac{1}{2}}^{n}}{\Delta y}\right)$$

$$ux_{i+\frac{1}{2},j}^{n+1} = ux_{i+\frac{1}{2},j}^{n} - \frac{\Delta t}{\rho_0}\left(\frac{p_{i+1,j}^{n+\frac{1}{2}}-p_{i,j}^{n+\frac{1}{2}}}{\Delta x}\right)$$

$$uy_{i,j+\frac{1}{2}}^{n+1} = uy_{i,j+\frac{1}{2}}^{n} - \frac{\Delta t}{\rho_0}\left(\frac{p_{i,j+1}^{n+\frac{1}{2}}-p_{i,j}^{n+\frac{1}{2}}}{\Delta y}\right)$$

Por lo tanto tenemos que, todas las velocidades de las partículas son calculadas y guardadas en memoria para un punto en particular de tiempo usando el valor de presión calculado anteriormente. De la misma manera, todas las presiones son calculadas usando el valor de velocidad obtenido anteriormente. Este ciclo se repite tantas veces como sea necesario.

Para asegurar una convergencia numérica, el tiempo entre cálculos debe ser lo suficientemente pequeño como para poder describir la propagación de la onda. Los límites de la relación entre el espacio y el tiempo entre cálculos viene dada por la siguiente ecuación, que para dos dimensiones se define así:

$$s = c\Delta t\sqrt{\left(\frac{1}{\Delta x}\right)^2+\left(\frac{1}{\Delta y}\right)^2} \le 1$$

Existen varias formas de introducir fuentes en los diseños, la excitación es introducida como una condición inicial en el diseño y se pueden añadir varias fuentes:

$$p^{n+\frac{1}{2}}_{i_{source},j_{source}} = g(t)$$

Una de las maneras para introducir una fuente en el diseño es, por ejemplo, introducir una señal impulso de esta manera:

$$W(t) = -\sqrt{2}\,\pi\, f_{cent}\left[\left(\sqrt{2}\,\pi\, f_{cent}\, t\right)^2 - 1\right]\exp\left[-\frac{1}{2}\left(\sqrt{2}\,\pi\, f_{cent}\, t\right)^2\right]$$

Para evitar errores de discretización, es necesario incluir para cada dimensión factores de atenuación. Esto es necesario para que no surjan fallos en los límites del área de integración.

$$\frac{\partial p_x}{\partial t} + \gamma_x p_x + \frac{1}{\mathrm{K}_e}\left(\frac{\partial u_x}{\partial x}\right) = 0$$

$$\frac{\partial p_y}{\partial t} + \gamma_y p_y + \frac{1}{\mathrm{K}_e}\left(\frac{\partial u_y}{\partial y}\right) = 0$$

Por lo tanto las ecuaciones de presión y velocidad quedan de la siguiente manera:

$$\frac{\partial p}{\partial x} + \rho\left(\frac{\partial u_x}{\partial t} + \gamma_x u_x\right) = 0$$

$$\frac{\partial p}{\partial y} + \rho\left(\frac{\partial u_y}{\partial t} + \gamma_y u_y\right) = 0$$

El factor de atenuación es cero dentro del área de integración y va incrementando conforme se va acercando a los límites de dicha área. Este incremento viene dado por la ecuación de la técnica de *Perfectly Matched Layers (PMLs):*

$$\gamma_x = \gamma_{x\max}\left|\frac{x - x_0}{x_{\max} - x_0}\right|^n$$

Donde $\boldsymbol{x_0}$ es el punto inicial del *PMLs*, $\boldsymbol{x_{max}}$ es el último punto del *PMLs* y $\boldsymbol{n}$ es un número entre dos y tres según convenga el estudio.

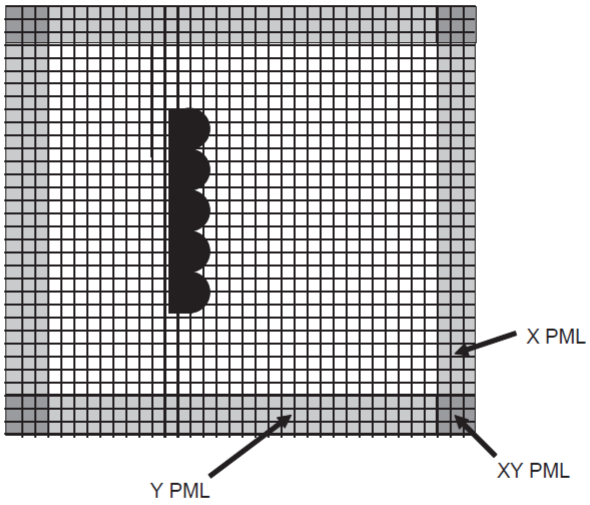

Figura 6: Distribución del factor de atenuación por el área de integración.

2. Justificación del proyecto.

2.1 Objetivo principal del proyecto.

Para la elaboración de este proyecto, en un principio se marcaron unos objetivos de los cuales cabe destacar el objetivo principal: *"diseñar una interfaz gráfica para la simulación en acústica de salas. Se trata de crear una herramienta útil, sencilla, eficaz, potente y, por qué no, didáctica para poder simular de manera visual y auditiva el comportamiento del sonido en recintos y en elementos acústicos usados habitualmente"*. (texto extraído de la Memoria previa).
Llegados a este punto y teniendo en cuenta que la primera versión del *Vi_Ac* (la 1.0) es una versión válida que cumple con el objetivo principal, se debe contestar a una pregunta: ¿por qué desarrollar un nuevo proyecto si ya existe uno de similares características y que cumple con los objetivos?.
La respuesta no es complicada pero tampoco breve, los puntos siguientes de este apartado segundo contestan a la pregunta formulada para así justificar la realización de este proyecto.

2.2 Inconvenientes y fallos del *Vi_Ac 1.0.*

Hay que decir que el programa original no tiene fallos graves ni de programación ni de funcionamiento, ni tampoco hay cosas a medio hacer que impiden que el programa no haga lo que debe de hacer. El *Vi_Ac 1.0* está completo y todas sus funciones son correctas.
Sin embargo, también hay que decir que cuando se lleva tiempo trabajando con la interfaz, se llega a la conclusión que pequeños detalles de programación hacen que aparezcan inconvenientes a la hora de trabajar. Estos inconvenientes son los siguientes:

- Funcionalidad con todas las versiones de **MATLAB**: para trabajar con la interfaz se dispone de los archivos **.p* de tipo *MATLAB P-code* que son ejecutados desde **MATLAB** para arrancar la aplicación. Como es sabido estos archivos están compilados y codificados por el propio **MATLAB.**
  El problema viene a la hora de ejecutar dichos archivos en una versión de **MATLAB** distinta de la que se compilaron. Cuando sucede esto, el programa responde así:

  ??? Corrupt P-file

- Problemas originados por el cambio de resolución: cuando se abre un nuevo objeto de tipo *figure, axes* o *uicontrol,* aparecen en el código sentencias como éstas:

```
h=uicontrol('style','pushbutton');
set(h,'position',[250 1 40 20])
```

  El problema es que al determinar la posición y el tamaño del objeto no se tiene en cuenta la resolución actual de la pantalla, por lo que al hacer un cambio de resolución con el programa ejecutándose, se producen efectos no deseados.

- Bloqueos al cerrar algunas de las ventanas: como en todos los programas, a lo largo de su ejecución aparecen ventanas demandando datos de entrada al usuario. En el *Vi_Ac 1.0* muchas de estas ventanas no se pueden cerrar ya que esto origina fallos de funcionamiento (mensajes de error y cierre repentino del programa).

- Errores al intentar corregir diseños: en el apartado de diseño de recintos, existe la función *borrar/remove wall* mediante la cual podemos borrar lo dibujado con anterioridad. Muchas veces esta función no realiza su tarea de manera correcta.

- Problemas específicos de funcionamiento: no tiene muchos problemas de este tipo ni tampoco son graves, se han tenido en cuenta porque son muy fáciles de solucionar. Por poner un ejemplo, a la hora de añadir un nuevo material e introducir sus coeficientes de absorción, el programa no tiene en cuenta si esos datos son correctos o no; es decir que se puede introducir un valor de absorción mayor que 1con el consiguiente trastorno en los datos de simulación.

2.3 Solución de problemas.

Una vez localizados los errores y/o inconvenientes, el siguiente paso es dar una solución. De los errores de programación la solución es trivial, habrá que añadir las sentencias adecuadas para el correcto funcionamiento del programa; y del problema de funcionalidad con otras versiones de **MATLAB** la solución pasa por crear un archivo ejecutable (**.exe* para *Microsoft Windows*) para trabajar independientemente de **MATLAB**.
Evidentemente no es el grueso del desarrollo de este proyecto, quizá sólo sea un veinte por cien del total pero, si se pretende partir de una base para crear algo más complejo, esta base debe ser sólida en cuanto a funcionamiento y ausencia de errores.
Todo lo realizado para subsanar todos estos problemas y para la elaboración de la nueva interfaz se especifica en el punto 3.1 de este documento.

2.4 Innovaciones del nuevo programa.

Es en este punto donde se da respuesta a la pregunta que se hacía al principio del punto *2. Justificación del proyecto.* Es el grueso del proyecto, el ochenta por cien restante que quedada después de corregir los errores de base. Es la creación de una interfaz gráfica para simulación en acústica de salas. Es la creación del *Virtual Room Acoustics 2* (a partir de ahora *Vi_Ac 2*).
Lógicamente no vale cualquier mejora que se pueda proponer, hay que tener claro el enfoque que se le quiere dar a la herramienta que se va a crear, hay que pensar para que va a servir y, sobretodo,quien la va a utilizar.
En los objetivos se propone la creación de una herramienta didáctica, que enseñe, que muestre la propagación del sonido de una manera intuitiva, que haga ver cosas que,seguramente el usuario ya sepa,pero que refuerce esos conocimientos adquiridos en la teoría.
Las nuevas mejoras o funciones que se añaden sirven básicamente para poder hacer más cosas que con el *Vi_Ac 1.0*, para poder avanzar un paso más, para poder investigar y experimentar un poco más, para tener más datos sobre las simulaciones,etc... En definitiva para saber más.

Teniendo en cuenta para que va a servir la interfaz, se proponen estos cambios y mejoras de funcionamiento:

- Nueva interfaz gráfica: la interfaz del *Vi_Ac 1.0* es correcta pero algunas ventanas no aparecen de forma ordenada y en ocasiones resulta incómodo.

- Introducir unidades en los gráficos: los gráficos que aparecen durante la simulación no tienen unidades y puede llevar a interpretaciones erróneas.
- Posibilidad de guardar proyecto para futuros retoques: un diseño guardado con el *Vi_Ac 1.0* es un diseño que no admite ninguna modificación.

- Introducir sonómetros: lo que se pretende con esta mejora es introducir indicadores en los receptores que nos muestren durante la simulación el nivel de señal recibido.

- Generación de informes con características del recinto: al acabar la simulación, el programa no deja datos sobre la simulación que se acaba de realizar para poder estudiarla o compararla con otras.

- Texto de ayuda: aunque sea un texto básico pero es necesario para cualquier programa por si surge alguna duda por parte del usuario.

- Poder simular cualquier archivo de audio: para así obtener una simulación y una respuesta con pequeños archivos de audio como puedan ser golpes de tambor, etc,...

- Poder poner audiencia en los recintos: la audiencia forma un papel esencial en la **Acústica de Salas,** por eso resulta interesante poder agregar público en los diseños para simular de forma más exacta la respuesta de una sala.

- Varios oyentes y varias fuentes: la herramienta de simulación sólo permite la simulación con un solo oyente.

- Mejoras en el apartado "Tiempo de simulación": el tiempo de simulación es el tiempo durante el cual el programa está generando la simulación. Durante ese tiempo, la fuente está activa por lo tanto no hay un tiempo de decaimiento o *sustain.* Controlando también el tiempo que la fuente está activa se logran dichos efectos.

- Creación de recintos a partir de coordenadas: para así poder diseñar objetos con total precisión.

- Posibilidad de obtener un video de la simulación: es interesante poder generar archivos de video de las simulaciones para poder almacenarlos y estudiarlos con detenimiento (poder detener el video en cualquier momento, avanzar rápido, retroceder, etc...).

- Posibilidad de obtener un archivo de audio de cada uno de los oyentes: de esta manera se puede oír el efecto de la sala sobre el sonido de la fuente.

- Nuevos materiales para el diseño de recintos: es decir, ampliar la base de datos por defecto con materiales usados normalmente en la construcción de recintos.

- Análisis en frecuencia: para poder ver la respuesta en frecuencia de la sala mediante un gráfico.

- Introducir banco de sonidos: sonidos de corta duración (como palmadas) que ayuden para el estudio de la respuesta del diseño simulado.

- Creación de un archivo ejecutable sobre plataformas ***Microsoft Windows***: consiste en la creación de un paquete autoextraible y autoejecutable que contenga el programa en sí y, además, **MATLAB Compiler Runtime (MCR)** con todas las bibliotecas necesarias para el funcionamiento del programa.

- Configuración del programa: creación de una serie de opciones para que el usuario pueda, de alguna manera, personalizar las simulaciones.

3. Resultados.

3.1 Desarrollo del nuevo programa.

El proceso de creación de cualquier herramienta informática pasa por varias etapas de desarrollo que hay que seguir de manera estricta y ordenada para conseguir resultados óptimos. El caso de *Vi_Ac 2* no es una excepción, ha ido evolucionando poco a poco desde el punto de partida y se ha ido añadiendo código hasta completar los objetivos propuestos pasando por diversas alternativas que han surgido, forzosamente, ante la aparición de problemas y/o errores.

3.1.1 Primeros pasos.

Si lo que se pretende es crear un programa que tiene como base otro programa, lo primero que hay que hacer es asegurarse que el programa "base" funcione perfectamente.
Lo primero que nos encontramos es que *Vi_Ac 1.0* no convence en muchos de sus aspectos, muchas de sus acciones pasan inadvertidas para el usuario y, en otras ocasiones, el usuario no es consciente de que si se están realizando de manera correcta dichas acciones. Por todo esto se deciden realizar los siguientes cambios en el código del programa:

- En ***materials:*** se añade un aviso por si no hay elementos en la lista de materiales que borrar, antes de borrar un material de la lista, el programa confirma la acción al usuario y de esta manera se puede eliminar el aviso que advierte de que un material ha sido borrado.

- En ***restore_materials*** aparece un cuadro de diálogo que confirma el reseteo de la lista de materiales.

- En las funciones: ***design, simulation, new_materials*** y ***materials,*** se comprueba si la lista de materiales (llamada ***materials_list.mat***) existe; ya que estas funciones no pueden trabajar sin dicho archivo.

Otra de las cosas que no convencen de *Vi_Ac 1.0* es que es bastante inestable, el código original no tiene en cuenta los posibles errores de manejo del usuario y ante cualquier adversidad (ya sea a la hora de introducir datos de manera errónea o a la hora de cerrar ventanas para volver al punto anterior)el programa responde con el cierre total de la aplicación y, **MATLAB** alertando sobre los errores.

También llama la atención el hecho de que la interfaz no verifica la información que se le introduce, por ejemplo, si para la realización de una tarea en concreto el programa ha de recibir un objeto de tipo *double* y recibe un objeto de tipo *string,* éste no advertirá de tal incoherencia y a la hora de procesar los datos de entrada para la realización de dicha tarea se producirán, lógicamente, errores graves.

Para subsanar este tipo de inconvenientes se procede a lo siguiente:

- En ***materials:*** la ventana de la función ya no se cierra hasta que lo hace manualmente el usuario, para ello se añade un *pushbutton.*

- En ***new_materials:*** al cancelar esta función ya no se cierra todo el programa, los datos introducidos son revisados para que estén dentro del rango lógico (también se comprueba que sean del tipo correcto) y se comprueba que no se introduzca un nombre para un material que ya exista.

- En ***simulation:*** se añade código para que al cerrar cualquiera de sus ventanas no produzca errores y cierres inesperados del programa. Como se ha hecho en *new_materials,* ahora el programa comprueba que los datos de entrada sean los correctos, corrigiendo automáticamente valores erróneos de entrada así como sustituyendo datos de entrada que pueda dejar el usuario en blanco por datos correctos.

Una vez se han solucionado los problemas que en un primer momento se advirtieron, se pasa a la parte estética de la interfaz y se modifica la apariencia de algunas de sus ventanas:

- En ***new_materials:*** se modifican algunas ventanas para la mejora del aspecto visual.

- En ***design:*** se añade una rejilla (*grid on*) para poder diseñar con mayor precisión y, además se simplifica la interfaz de dicha función para mayor facilidad de manejo.

- En todas las funciones se repasa el código de inicio para que las ventanas que van apareciendo en el programa aparezcan centradas y, además, tengan en cuenta la resolución actual de la pantalla para adaptarse correctamente a ésta.

3.1.2 Datos de salida e información de elementos del programa.

Llegados a este punto, lo que se le va a exigir a la nueva interfaz, es que nos proporcione la máxima información posible en cuanto a lo sucedido en la parte de simulaciones; es decir, generar una información que pueda ser interpretada posteriormente por el usuario (gráficas, video y audio) y que le ayuden a comprender lo que ha ocurrido durante la simulación de un recinto cualquiera.
También resulta interesante mostrar más datos acerca de los materiales de construcción que se utilizan en las simulaciones, por eso parte del código se ha dedicado para mostrar gráficas del coeficiente de absorción respecto de la frecuencia.
Para el propósito de generar más datos al usuario se ha optado por:

- Añadir gráficas del coeficiente de absorción respecto de la frecuencia en la función ***new_materials.***

- Crear una subfunción que permite ver las características de los materiales en la función ***materials.***(coeficiente de absorción, gráficas,etc...)

- Añadir unidades en gráficos que antes no tenían (***simulation***).(amplitud y tiempo).

- Generar un archivo de video **.AVI* de la simulación para poder reproducirlo más adelante sin tener que volver a arrancar la función. El archivo se guarda en el directorio donde estemos trabajando.

- Generar dos archivos de audio **.WAV* con la señal impulso y con la señal respuesta. De esta manera se pueden comparar las dos señales reproduciéndolas en cualquier reproductor multimedia. Las características de los archivos son: 16 bits de margen dinámico y frecuencia de muestreo variable (dependiendo de la resolución y tamaño del diseño). Los archivos se guardan en el directorio donde estemos trabajando.

- Introducir medidores de nivel en *dB* para todos los oyentes en las simulaciones. Funcionan a tiempo real y nos dan una idea clara del nivel de señal en un determinado momento (*-inf* para ausencia de señal y *0 dB* para nivel máximo).

- Generar un informe con las características más significativas de cada proceso de simulación. Estas características son: gráficas con las señales impulso y respuesta, tanto en el dominio del tiempo como en el de la frecuencia (para cada oyente se genera una gráfica de tiempo y frecuencia distintas). El informe se guarda en un archivo **.pdf* en el directorio donde estemos trabajando.

3.1.3 Nuevas funciones y herramientas.

Finalmente para completar la interfaz se han desarrollado unas nuevas funciones que harán que el programa sea más cómodo y fácil de usar y que tenga un enfoque más claro hacia la **Acústica de Salas.**

3.1.3.1 Función *Opciones.*

Se ha creado desde cero una nueva función llamada ***opciones*** mediante la cual se pueden configurar varios elementos del programa con la intención de que éste sea más cómodo y rápido de manejar.
La función arranca desde la ventana principal mediante un *pushbutton* y permite configurar los siguientes aspectos:

- **Archivos de audio**: permite la posibilidad de crear o no los archivos de audio que se generan después de cada simulación. De esta manera el programa puede ir más rápido y, además se evita el riesgo de borrado de otros archivos anteriormente generados.

- **Archivo de video**: idénticas características que con la opción de los archivos de audio, pero en este caso con el archivo de video.

- **Gráficas**: durante la simulación aparecen a tiempo real, las gráficas de impulso y respuesta y el video de la simulación que el programa va generando. Este hecho hace que el tiempo que tarda el programa en hacer la simulación aumente. Con esta opción se pueden desactivar dichas gráficas para que el tiempo de ejecución disminuya.

La configuración se guarda en un vector llamado *provis* que se encuentra dentro de un archivo tipo **.mat* llamado *archivopciones.mat.*

3.1.3.2 Archivos **.WAV* como señal impulso.

Hasta el momento solamente se podían estudiar dos tipos de casos: tono puro e impulso. Con la incorporación del nuevo código, *Vi_Ac 2* permite ,además, hacer estudios con cualquier archivo en formato de ondas **.WAV* . El programa es capaz de importar un archivo de audio (a elección del usuario), adaptarlo para el tiempo de la simulación, resamplear al archivo e incorporarlo como señal impulso.

Por norma general, nos encontramos que los archivos de audio en formato *wav* tienen una frecuencia de muestreo de 44.100 Hz, muy por encima de nuestra frecuencia de trabajo (si trabajamos con resoluciones bajas y/o medias), por lo que se hace inevitable resamplear la señal de entrada.
El programa calcula la frecuencia de muestreo dependiendo del vector *maxttt* que a su vez depende del vector *maxtt* y de las variables *nx*, *ny* y *answer{1}* (que dependen del tamaño de la matriz de diseño,de la resolución y del tiempo de simulación introducido por el usuario).
Cuanto mayor sea *maxttt* para un mismo tiempo de simulación (es decir, cuanta más resolución exista en el diseño), más samples por segundo (BitRate) habrán y mayor será la frecuencia de muestreo con la que se pueda trabajar.

Ejemplo de cálculo del tiempo de la simulación:

```
maxtt=round(sqrt(nx.^2+ny.^2)/sqrt(2)/50);
maxttt=ceil(str2num(answer{1})/1000/(maxtt*dt));
t=((1:maxtt*maxttt)*dt);
```

Ejemplo para el cálculo de la nueva frecuencia de muestreo:

```
[nad,samples]=size(t);%número de samples totales
[nada,sizew]=size(w);
 as=as/1000;
fs=samples/as%samples por segundo BitRate
fs=round(samples/as)
```

Una vez obtenida la nueva frecuencia de muestreo se procede a resamplear la señal, dependiendo de la frecuencia del archivo de audio y de nuestra frecuencia de trabajo se pueden dar tres casos:

- **Diezmado**: la frecuencia de entrada es mayor que la de trabajo, por lo tanto la señal es resampleada y pierde componentes espectrales por el fenómeno de *Aliasing* ya que ***fs>=fmax*2.***

- **Sobremuestreo**: es el caso contrario al anterior, por lo que la señal de entrada es sobremuestreada y no pierde componentes espectrales.

- **Frecuencia entrada igual a frecuencia de trabajo**: en este caso las dos frecuencias coinciden y no es necesario resamplear.

3.1.3.3. Función *design2.*

A partir de la función ***design*****,** que es la encargada de ejecutar las herramientas para el diseño de recintos, se crea la función ***design2.*** La diferencia con la primera es que ésta permite abrir proyectos existentes, volver a diseñar sobre ellos nuevos elementos y guardarlos ya sea para ejecutar la simulación o para volver a diseñar en otro momento. Por lo que ya no es necesario finalizar un proyecto si no se desea, ya que se puede volver a abrir en otro momento.
Los datos de entrada son el archivo ***.mat* que contenga el proyecto que se desea abrir, el programa coge los valores *a*,*b,c* y *res* (que se ubican dentro de dicho archivo) y genera por pantalla el diseño para poder hacer modificaciones.
Para esta operación el programa genera un archivo temporal *planta.tif* que al finalizar se elimina automáticamente.

Ejemplo parte del código de la función:

```
d(:,:,1)=a;
d(:,:,2)=b;
d(:,:,3)=c;
grosor=res;
```

3.1.3.4 Diseños mediante coordenadas.

Se añade a las funciones ***design*** y ***design2*** una nueva herramienta a la hora de dibujar diseños. Se trata de la posibilidad de dibujar rectas mediante sus coordenadas. Si se elige esa opción, el programa abre una nueva ventana donde se le pide al usuario las coordenadas de la nueva recta; estas coordenadas corresponden a los valores de inicio de la recta en el eje X y en el eje Y, y a los valores del otro extremo de la recta en los dos ejes respectivamente.
Al finalizar es necesario que el elemento diseñado sea rellenado de color amarillo para su posterior simulación, para ello se ha creado una subfunción que se encarga de esta tarea y es llamada por mediación de un *pushbutton* que su título es *Rellenar.*

3.1.3.5 Tiempo de fuente activa.

En la práctica, cuando lo que se pretende realizar es un estudio sobre el tiempo de reverberación de un recinto cualquiera, se utiliza una fuente de audio que genera un tono o ruido ponderado durante un determinado tiempo y después se detiene; y es en ese instante cuando empieza el estudio de la reverberación de la sala en cuestión. Si la fuente no se detuviese no se podría realizar dicho cálculo ya que siempre habría sonido directo.
Al realizar un programa sobre simulación en **Acústica de Salas** se plantea la misma cuestión, por este motivo para el desarrollo de este programa se ha pensado un código que permita separar el tiempo de la simulación, es decir su duración, con el tiempo en el que la fuente excitadora del impulso esté activa. De esta manera el programa permite realizar estudios en los cuales se puede decidir el tiempo en que habrá sonido directo y sonido reflejado, y el tiempo que habrá sólo sonido reflejado.

Al arrancar la función ***simulation,*** uno de los datos de entrada es: *"Tiempo de fuente activa"*, que es el que determina la duración del estado activo de la fuente. Para el cálculo del vector de tiempo de fuente activa el programa utiliza un código idéntico al que utiliza para el tiempo de duración de la simulación:

```
maxttact=round(sqrt(nx.^2+ny.^2)/sqrt(2)/50);
maxtttact=ceil(str2num(answer{2})/1000/(maxtt*dt));
```

3.1.3.6 Simulación con varias fuentes.

En la versión anterior de *Vi_Ac* (1.0) uno de los parámetros que no era ajustable era el número de fuentes para la simulación. Se puede pensar que para un programa de estas características no es necesario que el número de fuentes sea mayor que uno pero, si se piensa de que manera puede beneficiar al usuario que este número sea mayor, resulta interesante el plantear el aumento del número de fuentes.

En esta nueva versión (2), es posible escoger de una a tres fuentes excitadoras de impulso, con esta medida se pueden comprobar entre otras cosas:

- En que medida aumenta el nivel de señal en el oyente dependiendo si el número de fuentes es mayor o menor.
- Que colocando dos o más fuentes de manera que se produzca un desfase entre ellas, se produce una atenuación (o incluso eliminación) de un rango concreto de frecuencias. (Muy útil cuando se piensa en configuraciones de altavoces en linea o *line array*).

3.1.3.7 Función *ayuda.*

Esta función no es más que una nueva ventana que se abre al pulsar en la pantalla principal el botón *Ayuda,* y que muestra un *popupmenu* con las diferentes funciones o herramientas del programa. Al elegir del *popupmenu* cualquiera de las opciones que ofrece, aparece detallado todas las funciones que ofrece el programa con un sencillo texto explicativo para que así, cualquier usuario pueda comenzar a utilizar el programa sin ninguna ayuda más.
Para programar esta función se ha utilizado la herramienta **GUIDE (Graphical User Interface Development Environment)** que nos ofrece **MATLAB,** y que sirve precisamente para crear interfaces gráficas de usuario (*GUI's*).
La ventaja de usar **GUIDE** para el desarrollo de *GUI's* es que permite su creación de una manera más visual e intuitiva. También es más práctico ya que a la hora de hacer modificaciones, con el archivo que genera **.fig* se puede abrir de nuevo el editor y cambios.

3.1.3.8 Función *viac2.*

Es la función que sustituye a la ya obsoleta ***start,*** realiza exactamente lo mismo: abre la ventana principal del programa y llama por medio de *pushbutton* a las demás funciones.
La única diferencia con la antigua versión es que, ésta, ha sido diseñada con **GUIDE** por las ventajas que conlleva ya comentadas anteriormente.

3.1.3.9 Banco de sonidos.

Como material extra y de ayuda*,* se incluye un fichero con diferentes sonidos distribuidos en archivos de formato de ondas (**.wav*) para utilizar en las simulaciones. Estos archivos son:

- bombo.wav
- caja.wav
- palmada.wav
- pandereta.wav
- tom.wav

Han sido creados mediante una estación de audio digital y son muy útiles a la hora de escuchar como suena el efecto de reverberación (*reverb*) de una sala sobre un sonido concreto.
Por ejemplo: se efectúa una simulación con *Vi_Ac 2* y se generan dos archivos de audio: la señal impulso y la señal respuesta;se introducen estas dos señales en una estación de audio digital y se le

aplica a la señal respuesta un fundido de salida a modo de puerta de ruido. Se sonorizan ambas señales para que la señal impulso suene por arriba de la respuesta y el efecto producido es un efecto de *reverb* que ha sido generado por *Vi_Ac 2* según hallan sidos sus datos de entrada (diseño, materiales, tiempo,etc...).

3.1.4 Generar archivo ejecutable.

Uno de los objetivos que se planteaba para este proyecto era que la interfaz fuese compatible con todas las versiones de **MATLAB**. Para ir más allá se pensó que fuese independiente de **MATLAB,** para lograr esto se ha creado un archivo ejecutable que contiene la interfaz; así se puede ejecutar el programa en cualquier ordenador tenga o no **MATLAB** instalado.
Para crear el ejecutable se ha procedido como sigue:

- En *Command Window* se llama al compilador *deploytool* (*Deployment Tool*).
- Se genera un proyecto **.prj*.
- Se añade al proyecto la función principal (***viac2***).
- Se añaden al proyecto las demás funciones y archivos secundarios. (ayuda.m, ayuda.fig, design2.m,etc...).
- Se marca la opción de incluir **MCR (MATLAB COMPONENT RUNTIME),** en la compilación.
- Generación del archivo ejecutable ***viac2.exe.***
- Generación del archivo autoextraible ***viac2_pkg.exe.*** (***viac2.exe*** y ***MCR***).

Ejecutando el archivo ***viac2_pkg.exe*** en cualquier ordenador, se instala :

- **MATLAB COMPONENT RUNTIME**.
- Las librerías necesarias para la la ejecución de **VIRTUAL ROOM ACOUSTICS 2.**
- Un archivo ejecutable que arranca la aplicación (**Vi_Ac 2**).

3.2 Resumen de resultados.

De todos los resultados obtenidos, cabe destacar los más relevantes o significativos. Son aquellos que han supuesto que ***Vi_Ac 2*** sea ahora más potente y disponga de nuevas y mejores herramientas para el estudio de la propagación del sonido. En cierta medida todos los resultados tienen su importancia ya que, como se ha dicho anteriormente, el programa ha ido evolucionando desde su inicio hasta convertirse en una herramienta completa, pasando por diferentes estados o etapas. Dentro de esa evolución, cada resultado obtenido no ha sido más que una continuación del anterior.

He aquí una lista de los resultados más importantes:

- Estabilidad en el funcionamiento del programa.
- Disposición de mayor información al usuario.
- Nuevos archivos de salida para el mejor estudio de las simulaciones.
- Nuevas opciones a la hora de realizar simulaciones.
- Nuevas herramientas para el diseño.
- Mayor control sobre las fuentes.

4. Conclusión.

Llegados a este punto, queda valorar el desarrollo del proyecto y su estado actual después de todas las modificaciones realizadas.
El programa ha pasado por todas las fases de desarrollo que se habían previsto hasta llegar a su finalización. En cada una de esas etapas de desarrollo habían previstos una serie de objetivos que, en gran medida se han cumplido haciendo que el resultado final sea completamente satisfactorio.
Desde un punto de vista global y observando todas las características del programa y su funcionalidad, queda por tanto justificada la realización de este proyecto.
Si bien es cierto, que en alguno de los casos, en la implementación de algunas de las funciones, el camino seguido no ha sido el esperado debido a la magnitud de algunas de las propuestas pensadas en un principio; entre muchos aspectos por las limitaciones del programa.

Dicho todo esto, el resultado final es un programa con el que se pueden realizar diseños en dos dimensiones de recintos escogiendo los materiales que lo componen, puede observarse y oírse el sonido generado en dicho recinto, puede verse como afecta en la frecuencia la arquitectura del recinto, pueden obtenerse informes para su posterior estudio, etc... En definitiva se puede hacer una idea de cómo afectara los elementos de una sala o recinto para música, en la evolución del sonido. Y lo más importante, queda una herramienta completa, didáctica y cuya evolución no tiene porque detenerse aquí.
Además de todo esto, el programa realiza tales acciones de una manera estable e inteligente en cierta manera (ya que comprueba los datos de entrada). Es, también flexible y configurable, a la vez de tal manera que se puede adaptar a muchos más usuarios.

Básicamente el programa está limitado por dos cuestiones fundamentales:

- Diseño.
- Uso de memoria.

Como se ha explicado y como se puede comprobar, el programa trabaja a la hora de realizar diseños y ejecutar simulaciones, en sólo dos dimensiones. Esto parece un problema pero cuando se piensa en el objetivo de la aplicación, que es ser una herramienta didáctica, pasa de ser un problema a ser una limitación. Limitación porque, además de no tener en cuenta que el sonido se propaga de manera omnidireccional, existen parámetros que no se pueden calcular como por ejemplo el tiempo de reverberación.
Aún así, resulta útil que el programa funcione con sólo dos dimensiones ya que de esta manera se puede visualizar la evolución del sonido mediante la simulación. Por lo tanto lejos de ser un problema, es una limitación que en algunos casos nos beneficia.
Se puede plantear para el futuro desarrollo de la interfaz un nuevo código que permita trabajar al programa en tres dimensiones.

La otra cuestión que limita el uso del programa y la que tiene mayor importancia es la del uso de la memoria, ya que se necesita mucha memoria virtual para simular con una frecuencia de muestreo aceptable.
Por ejemplo, si se desea hacer una simulación con una baja resolución (más rápido), se da que el vector de tiempo nos proporciona una frecuencia de muestreo de unos 11.700 Hz; lo que quiere decir que la máxima frecuencia que se puede simular sin que aparezca *Aliasing,* es la mitad de la frecuencia de muestreo, es decir 5.850 Hz.
Para remediar esta falta de ancho de banda, se puede recurrir al aumento de la resolución a la hora de realizar diseños, lo que hace que las simulaciones sean muy costosas en lo referente al tiempo de ejecución y hace que el ordenador se ralentice ya que la aplicación ocupa la mayor parte de la

memoria. Por lo que las opciones son dos: trabajar con una frecuencia de muestreo baja o realizar simulaciones más pesadas (unos 6 segundos por cada milisegundo de simulación).
Por otra parte si se decide realizar diseños con una gran resolución y de un gran tamaño, es muy probable que la memoria virtual sea insuficiente y **MATLAB** responde de la manera siguiente:

error "Out of memory"

Como propuesta para futuras lineas de trabajo, ya que es evidente que la evolución del programa puede continuar, se pueden plantear algunas modificaciones que, de seguro, mejorarán su rendimiento.
Algunas de estas propuestas pueden pasar por el desarrollo de un nuevo motor de diseño y simulación. Es decir, no cabe duda de que para estudiar de manera más precisa cómo afectan los elementos arquitectónicos a la evolución del sonido, es lógico pensar en un modelo en tres dimensiones que permita realizar dicho estudio de una forma más completa.
También, y relacionado con lo descrito en el párrafo anterior, el programa sería mucho más potente con unas nuevas herramientas para la realización de diseños, ya que si se pretende realizar simulaciones en tres dimensiones, también habrá que pensar en la creación de dichos diseños conforme al tipo de simulación.
Por último y no por ello menos importante, se deja planteado el solucionar el uso excesivo de recursos de memoria por parte del programa para poder realizar estudios en un rango de frecuencias superior, sin que esto afecte de manera tan drástica a los tiempos de ejecución ni limite el uso del programa a equipos mucho más potentes.

Como resumen de este proyecto puede servir este esquema en el que se destacan los puntos más importantes. A la versión anterior de *Vi_Ac*, se ha aportado lo siguiente:

- **MAYOR ESTABILIDAD.**
- **MÁS Y MEJORES FUNCIONES.**
    - DISEÑO.
        - COORDENADAS.
        - ABRIR PROYECTOS.
    - MATERIALES.
        - GRÁFICAS ALFA.
    - AYUDA.
    - SIMULACIÓN.
        - FUENTES.
        - TIEMPOS.
        - ARCHIVOS.
        - SONÓMETROS.
- **MÁS DATOS DE SALIDA.**
    - VIDEO.
    - AUDIO.
    - INFORME.
- **MAYOR POSIBILIDAD DE CONFIGURACIÓN.**

5. Bibliografía.

A continuación se detalla la bibliografía consultada para la realización del presente proyecto. Está dividida en cuatro partes claramente diferenciadas: manuales y libros sobre **MATLAB,** manuales y libros sobre audio en general, documentos de la ***Escuela Politécnica Superior de Gandía*** y páginas web y archivos de ayuda de **MATLAB**.

BIBLIOGRAFÍA SOBRE **MATLAB** :

[1] "A guide to MATLAB for Beginners and Experienced Users." Cambridge.

[2] "Digital Signal Processing Using MATLAB V.4." Ingle/Proakis.

[3] "Learning MATLAB 7." The MathWorks.

[4] "Aprenda Matlab 7.0 como si estuviera en primero." Javier García de Jalón, José Ignacio Rodriguez, Jesús Vidal. Universidad Politécnica de Madrid.

[5] "Graphics and GUIs with MATLAB". Marchand, Patrick CRC Press 1997.

[6] "Introducción a MATLAB y a la creación de interfaces gráficas". Carrera Amuriza, Ana Rosa – Universidad del País Vasco – 2004.

OTRA BIBLIOGRAFÍA Y MANUALES:

[7] "The art of digital audio". Watkinson, John. - Focal Press -1991.

[8] "Adobe Audition. Guía del Usuario".

[9] "Libro Básico Nuendo versión 1.5". Steinberg.

[10] "Reason Drum Kits. Refill Manual". Propellerhead Software.

[11] "Acoustic Absorbers and Diffusers. Theory, design and aplication. Second edition". Trevor J. Cox, Peter D'Antonio. -Taylor & Francis Group-.

MATERIAL DE LA ESCUELA POLITÉCNICA SUPERIOR DE GANDÍA:

[12] "Diseño de una interfaz gráfica para la simulación mediante Diferencias Finitas" [Recurso electrónico-CD-ROM] - Sainz-Pardo Marti, Sara - Universidad Politécnica de Valencia – 2007.

[13] Documentación de las áreas de:

- Programación.
- Programación avanzada.
- Diseño Acústico de Recintos.
- Acústica Arquitectónica.
- Audio Digital.
- Análisis de Sistemas Discretos.
- Prácticas con MATLAB.

OTRAS FUENTES:

[14] http://www.mathworks.com/support/tech-notes...

[15] http://www.mathworks.com/matlabcentral/fileexcahnge

[16] Doc help de MATLAB 7.10.0.499.

6. Anexos.

6.1 Máxima memoria virtual usada por MATLAB.

Si se ejecuta en *Command Window* la instrucción: *feature('memstats'),* aparecerá lo siguiente (dependiendo claro está del ordenador donde se ejecute):

Physical Memory (RAM):
In Use: 861 MB (35d59000)
Free: 2200 MB (898b2000)
Total: 3062 MB (bf60b000)
Page File (Swap space):
In Use: 633 MB (27902000)
Free: 6365 MB (18dd34000)
Total: 6998 MB (1b5636000)
Virtual Memory (Address Space):
In Use: 626 MB (2729c000)
Free: 1421 MB (58d44000)
Total: 2047 MB (7ffe0000)
Largest Contiguous Free Blocks:
1. [at 22810000] 1087 MB (43f30000)
2. [at 68036000] 83 MB (532a000)
3. [at 6d366000] 70 MB (466a000)
4. [at 746fc000] 28 MB (1c44000)
5. [at 66799000] 24 MB (1867000)
6. [at 71ab2000] 20 MB (14ce000)
7. [at 72fa6000] 18 MB (126a000)
8. [at 7eec1000] 8 MB (82f000)
9. [at 7f7f0000] 6 MB (69f000)
10.[at 7ab7b000] 6 MB (605000)
======= ==========
1353 MB (5497a000)

Con esta información y dependiendo de la memoria RAM que disponga nuestro ordenador, se puede averiguar cual es el máximo de memoria de la que dispone **MATLAB** para la ejecución de la aplicación. Este máximo viene dado por :

Largest Contiguous Free Blocks:

1. *[at 22810000] 1087 MB (43f30000)*

Es decir, máximo de memoria virtual 1087 MB, ya que **MATLAB** necesita bloques enteros de memoria al no poder fragmentar los archivos en ejecución.

6.2 Capturas de pantalla.

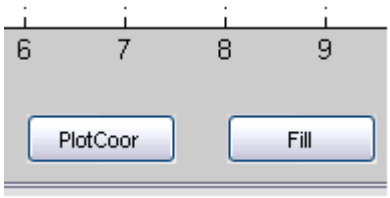

Figura 7: detalle de los botones para diseñar mediante coordenadas.

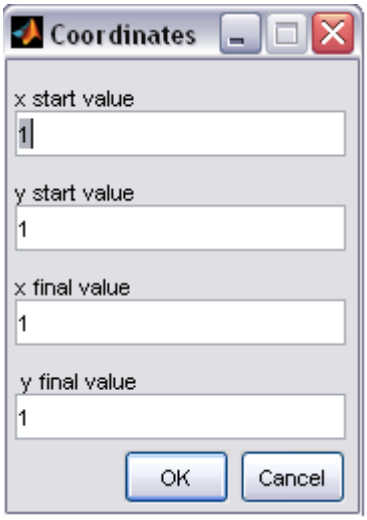

Figura 8: detalle de la interfaz para introducir coordenadas en el diseño.

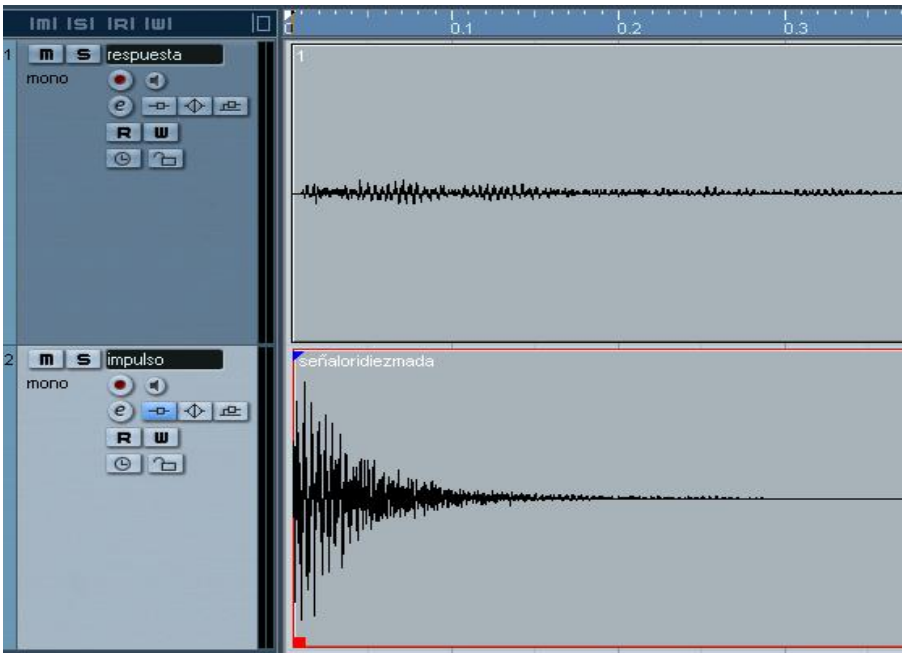

Figura 9: detalle de señales IMPULSO y RESPUESTA para el estudio de la reverberación en una estación de audio digital.

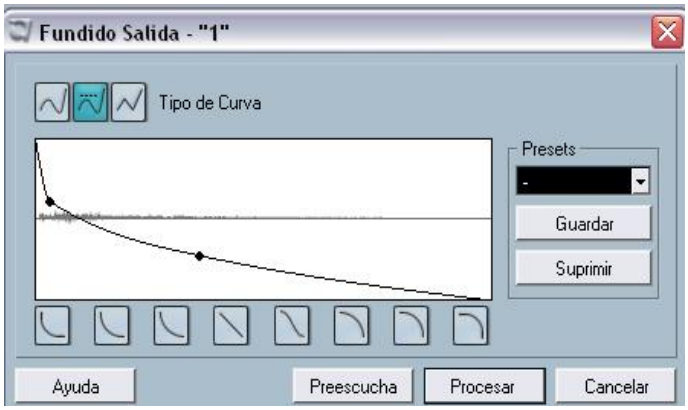

Figura 10: detalle del FUNDIDO DE SALIDA en la señal RESPUESTA para el estudio de la reverberación en una estación de audio digital.

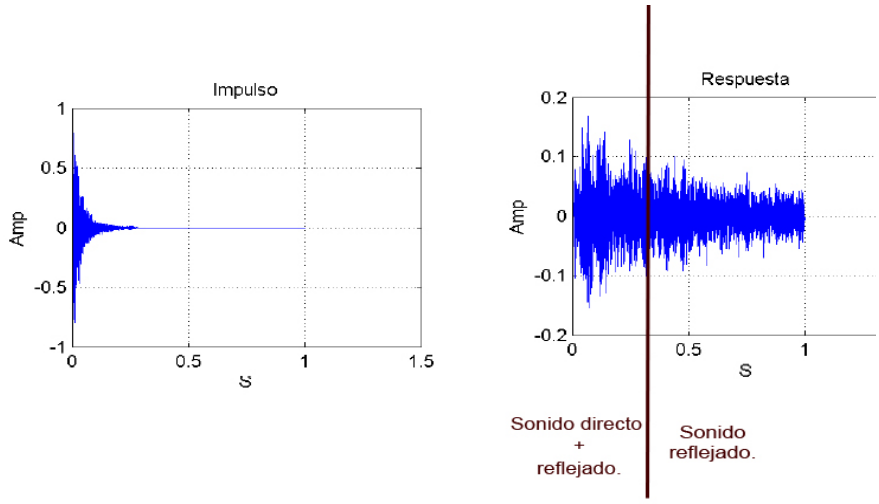

Figura 11: gráficas de señales IMPULSO y RESPUESTA para explicar la importancia sobre el control del tiempo de funcionamiento de la señal IMPULSO.

6.3 Diseño del código.

A continuación se muestra un diseño que contiene todas las funciones y subfunciones de la aplicación con la intención de mostrar a simple vista su estructura. No se ha utilizado ningún método estándar, simplemente se ha usado un protocolo que se explica a continuación:

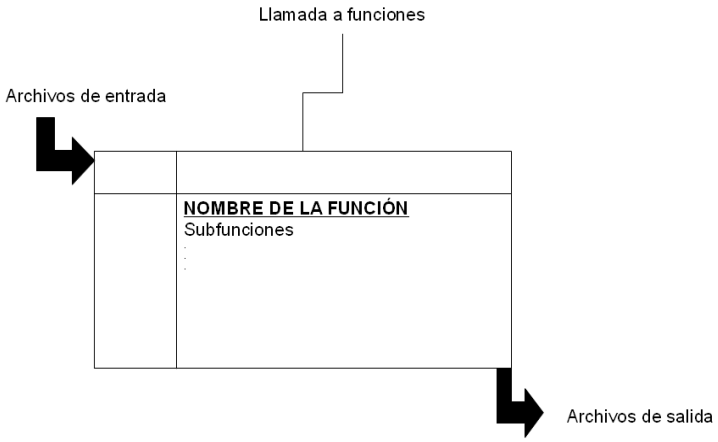

Figura 12: detalle de la metodología en la estructura del diseño.

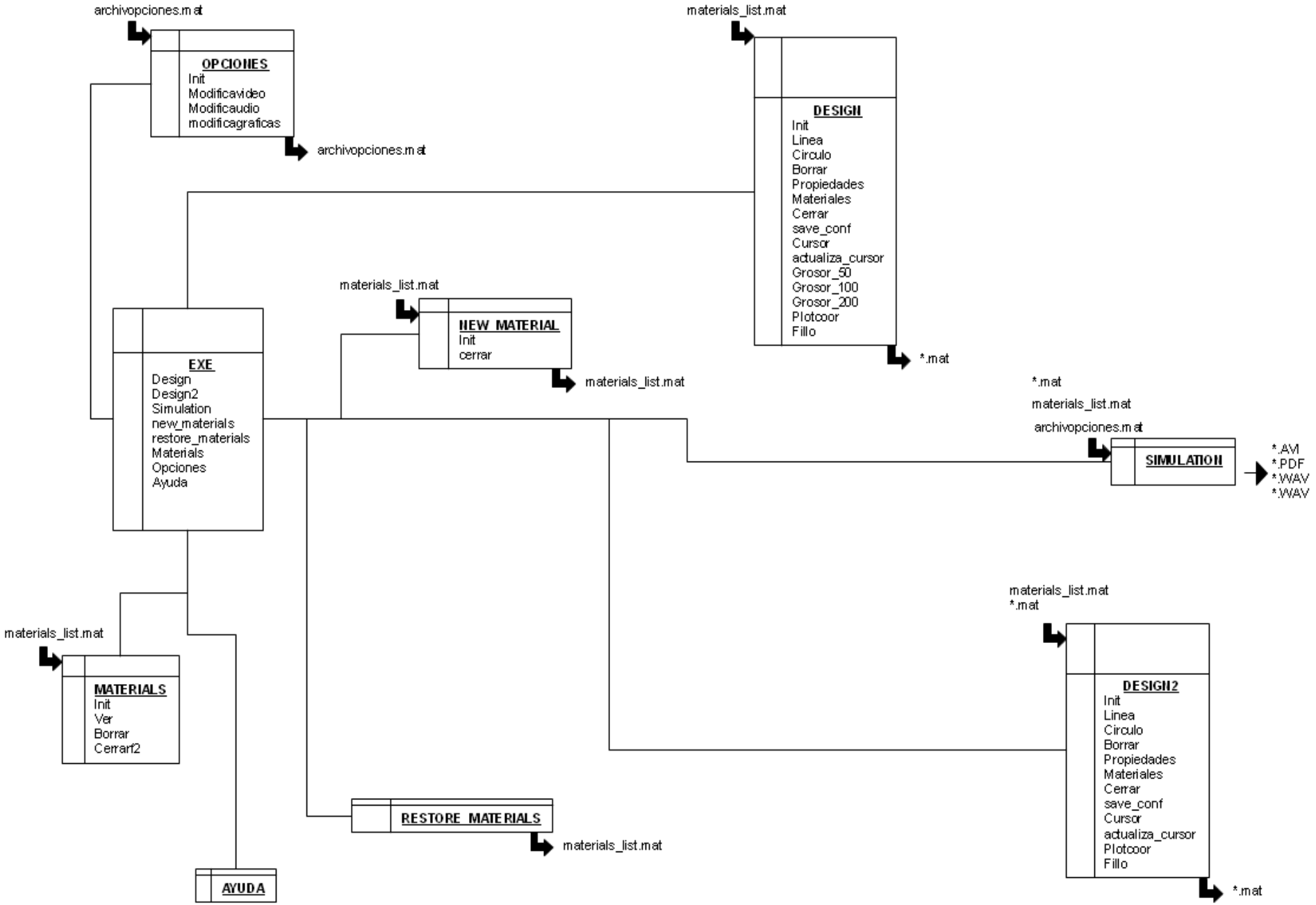

Figura 13: diseño del programa.

6.4 Código.

FUNCIÓN AYUDA.

```
%==========================================================================
%VIRTUAL ROOM ACOUSTICS 2
%Eugenio Garcés Leante
%==========================================================================

%==========================================================================
%FUNCIÓN AYUDA
%==========================================================================

%==========================================================================
%Muestra una ventana con comentarios de las diferentes funciones del
%programa para ayudar en su manejo
%==========================================================================

function varargout = Ayuda(varargin)

% AYUDA M-file for Ayuda.fig
%      AYUDA, by itself, creates a new AYUDA or raises the existing
%      singleton*.
%
%      H = AYUDA returns the handle to a new AYUDA or the handle to
%      the existing singleton*.
%
%      AYUDA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in AYUDA.M with the given input arguments.
%
%      AYUDA('Property','Value',...) creates a new AYUDA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Ayuda_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Ayuda_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Ayuda

% Last Modified by GUIDE v2.5 20-Feb-2011 17:55:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Ayuda_OpeningFcn, ...
                   'gui_OutputFcn',  @Ayuda_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Ayuda is made visible.
function Ayuda_OpeningFcn(hObject, eventdata, handles, varargin)

%==========================================================================
%Centrar ventana
%==========================================================================

scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/16);
yr=scrsz(4) - pos_act(4);
yp=round(yr/52);

set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
set(gcf,'name','Help');
axes(handles.axes1)
background = imread('diseño.jpg');
axis off;
imshow(background);

% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Ayuda (see VARARGIN)

% Choose default command line output for Ayuda
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Ayuda wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Ayuda_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)

getval=get(hObject,'Value');

if getval==1
axes(handles.axes1);
background = imread('diseño.jpg');
axis off;
imshow(background);

elseif getval==2
    axes(handles.axes1);
background = imread('sim.jpg');
axis off;
imshow(background);
```

```
elseif getval==3
  axes(handles.axes1);
background = imread('nuevo mat.jpg');
axis off;
imshow(background);

elseif getval==4
  axes(handles.axes1);
background = imread('resetear mat.jpg');
axis off;
imshow(background);

elseif getval==5
  axes(handles.axes1);
background = imread('ver.jpg');
axis off;
imshow(background);

elseif getval==6
  axes(handles.axes1);
background = imread('op.jpg');
axis off;
imshow(background);

else
    axes(handles.axes1);
background = imread('diseño.jpg');
axis off;
imshow(background);
end
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

FUNCIÓN DESIGN 2.

```
%=========================================================================
%VIRTUAL ROOM ACOUSTICS
%Eugenio Garcés Leante
%=========================================================================

%=========================================================================
%FUNCIÓN design2
%=========================================================================
%Función básicamente como "design",con la única diferencia de que ahora
%partimos de un diseño anterior y no de cero.
%El dato de entrada es el diseño con el cual queremos seguir trabajando.
%=========================================================================

function [] = design2(arg1,arg2)

 %=========================================================================
%Si no existe materials_list.mat el programa no puede trabajar
%=========================================================================

if(length(dir('materials_list.mat'))<1)
    errordlg('"materials_list" not found','¡¡WARNING!!','on');

    return

end
load ('materials_list.mat');
if length (materials_list)==0,


errordlg('"Materials_list" is empty','¡¡WARNING!!','on');
return
end

%=========================================================================
%CREA UN NUEVO CASE
%=========================================================================

if ~nargin,
 design2('init');
 return
end

%=========================================================================
%CREA UN NUEVO CASE Y
%crea un nuevo case y aplica las propiedades dadas en arg1
%=========================================================================

if nargin == 1,
 if isstruct(arg1),
   update(arg1)
  return
 end
end
```

```
%=========================================================================
%actualiza el case actual con las propiedades de arg2
%=========================================================================

if nargin == 2,
 if isstruct(arg2) && strcmpi(arg1,'update'),
   update(arg2)
  return
 end
end

%=========================================================================
%selecciona funcion (depending on arg1)
%=========================================================================

switch lower(arg1),

    case 'init'
        init
    case 'linea'
        linea
    case 'circulo'
        circulo
    case 'borrar'
        borrar
    case 'propiedades'
        propiedades
        delete(gca)
    case 'materiales'
        materiales
    case 'cerrar'
        cerrar
    case 'save_conf'
        save_conf
    case 'cursor'
        cursor
    case 'actualiza_cursor'
        actualiza_cursor
        case 'plotcoor'
        plotcoor
    case 'fillo'
        fillo


end

%=========================================================================
%Función principal
%=========================================================================

function [] = init



    name_file=0
name_file=uigetfile('sc_*.mat','Design file name','sc_my_design.mat');

if name_file==0
return
end
load (name_file)

grosor=res;
save('grosor.mat','grosor')
```

```
%==========================================================================
%crea la ventana principal
%==========================================================================

hfig=figure;
set(hfig,'tag','fig')
set(hfig,'interruptible','off')
set(hfig,'colormap',jet(256))
set(hfig,'renderer','zbuffer')
set(hfig,'menubar','none')
set(hfig,'numbertitle','off')
set(hfig,'name','Draw the room shape')

set(hfig,'resize','off')
set(hfig,'units','pixels')
set(hfig,'DefaultUIControlFontName','default')
set(hfig,'DefaultUIControlFontSize',8)
set(hfig,'DefaultUIControlFontAngle','normal')
set(hfig,'DefaultUIControlFontWeight','light')
set(hfig,'DefaultUIControlForeGroundColor','k')
set(hfig,'DefaultUIControlInterruptible','off')
set(hfig,'DefaultUIControlBusyAction','queue')
set(hfig,'DefaultUIControlbackgroundcolor',[0.878 0.875 0.89])
set(hfig,'color',[0.8 0.8 0.8])

scrsz = get(0, 'ScreenSize');
pos_act=get(hfig,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/3.5);
yr=scrsz(4) - pos_act(4);
yp=round(yr/7);

set(hfig,'Position',[xp-50 yp-10 895 708])

%==========================================================================
%crea el axes (panel de trabajo)
%==========================================================================

h=axes;




set(h,'units','pixels','position',[100 78 499 499])
set(h,'tag','axes')
set(h,'box','on')
set(h,'xtick',[])
set(h,'ytick',[])
set(h,'nextplot','replace')%add
set(h,'hittest','on')
set(h,'ALim',[0 10])


set(h,'Xlim',[0 10])
set(h,'Ylim',[0 10])



ax = get(hfig,'CurrentAxes');
if isempty(ax)
  ax = axes('Parent',hfig);
  end
ex=ejes(2);
ey=ejes(4);
  set(ax,'Xlim',[0 ex])
set(ax,'Ylim',[0 ey])
```

```
r=plot(ejes(2),ejes(4));
set(r,'tag','rejilla')
set(r,'markersize',4)
set(r,'hittest','off')
axis equal

a=rot90(a);
b=rot90(b);
c=rot90(c);
d(:,:,1)=a;
d(:,:,2)=b;
d(:,:,3)=c;

imwrite(d,'planta.tif','tif');


for i=1:ejes(2)+1
    x(i)=i-1;
end

for i=1:ejes(4)+1
    y(i)=i-1;
end



ab=imread('planta.tif');
delete planta.tif;
abc=image(x,y,ab);


grid on

%=========================================================================
%crea los botones
%=========================================================================

h=uicontrol('style','pushbutton');

set(h,'position',[200 10 75 25])
set(h,'string','__')
set(h,'tag','linea')
set(h,'tooltipstring','Plot lines')
set(h,'callback','design2(''linea'')')

h=uicontrol('style','pushbutton');
set(h,'position',[300 10 75 25])
set(h,'string','O')
set(h,'tooltipstring','Plot a circle')
set(h,'callback','design2(''circulo'')')

h=uicontrol('style','pushbutton');
set(h,'position',[400 10 75 25])
set(h,'string','PlotCoor')
set(h,'tooltipstring','Plot with coordinates ')
set(h,'callback','design(''plotcoor'')')

h=uicontrol('style','pushbutton');
set(h,'position',[500 10 75 25])
set(h,'string','Fill')
set(h,'tooltipstring','Fill(PlotCoor)')
set(h,'callback','design(''fillo'')')

h=uicontrol('style','pushbutton');
```

```
set(h,'position',[600 10 75 25])
set(h,'string','Save')
set(h,'FontSize',8.0)
set(h,'tooltipstring','Save')
set(h,'callback','design2(''save_conf'')')

h=uicontrol('style','togglebutton');
set(h,'position',[100 10 75 25])
set(h,'string','Coordinates')
set(h,'tag','coordenates')
set(h,'value',0)
set(h,'FontSize',8.0)
set(h,'tooltipstring','Show cursor position')
set(h,'callback','design2(''cursor'')')


set(ax,'xlim',[0 ejes(2)],'ylim',[0 ejes(4)]);

load ('materials_list.mat')
for ii=1:length(materials_list),
    for jj=640:-40:40,

        h=uicontrol('style','text');
        set(h,'position',[625 jj 154 21])
        set(h,'string',materials_list(ii).nombrematerial)
        set(h,'backgroundcolor',[0.878 0.875 0.89])

        c=uicontrol('style','text');
        set(c,'position',[800 jj 52 21])
        set(c,'backgroundcolor',materials_list(ii).color)

        ii=ii+1;
        if (ii>length(materials_list))
            return
        end
    end

end




%



return

%=========================================================================
%DIBUJAR CON COORDENADAS
%=========================================================================

function [] = plotcoor


        if(length(dir('figura.mat'))<1)

    m=1;

        else  load('figura.mat');
            m=length(datosx)+1;

end
```

```
prompt={'x start value','y start value','x final value','y final value'};
titulo = 'Coordinates';
num_lineas = 1;
def = {'1','1','1','1'};
answer = inputdlg(prompt,titulo,num_lineas,def);
if isempty(answer),

 return
end

ix=str2double(answer{1});
iy=str2double(answer{2});
fx=str2double(answer{3});
fy=str2double(answer{4});
wall=line([ix fx],[iy fy],'Color','k');
hold on;
set(wall,'LineWidth',5)%ojo
set(wall,'hittest','on')
set(wall,'tag','wall')
          menu=uicontextmenu;
          set(wall,'uicontextmenu',menu)
          menu1=uimenu(menu,'label','delete','callback','design(''borrar'')');
          menu2=uimenu(menu,'label','properties','userdata',wall,'callback','des
ign(''propiedades'')','separator','on');
datosx(m)=ix;
datosy(m)=iy;



m=m+1;
datosx(m)=fx;
datosy(m)=fy;

m=m+1;
if(length(dir('figura.mat'))<1)
   save('figura.mat');
end

save('figura.mat','datosx','datosy','-append');

   return%fucntion plot coor

%=========================================================================
%FUNCIÓN RELLENAR
%=========================================================================

        function []=fillo

            load('figura.mat');


             menu2=uicontextmenu;
            axis manual;
           relleno= fill(datosx,datosy,'y');
            set(relleno,'uicontextmenu',menu2)
             menu1=uimenu(menu2,'label','delete','callback','design(''borrar'')
');
            hold on;


            delete('figura.mat');
            return
```

```
%============================================================
%FUNCIÓN LINEA
%============================================================================

function [] = linea

% Dibuja una linea entre dos puntos. Al hacer la geometría, el recinto se
% cerrará presionando la tecla espacio, y el interior se rellenará de
% amarillo.

button=1;
nwall=-1;
i=1;
while button==1,
     [xp,yp,button]=ginput(1);
     if button==3,
         return;
     end
     x(i)=xp;
     y(i)=yp;
     if button==32,
         x(i)=x(1);
         y(i)=y(1);
     end
     if(nwall >= 0),
         wall=plot(x,y,'k');
         num=[x;y];
         set(wall,'LineWidth',5)%ojo
         set(wall,'hittest','on')
         set(wall,'x',[x(i-1) x(i)])
         set(wall,'y',[y(i-1) y(i)])
         set(wall,'tag','wall')
         menu=uicontextmenu;
         set(wall,'uicontextmenu',menu)
         menu1=uimenu(menu,'label','delete','callback','design2(''borrar'')');
         menu2=uimenu(menu,'label','properties','userdata',wall,'callback','des
ign2(''propiedades'')','separator','on');
         if button==32,
             menu2=uicontextmenu;
             relleno=fill(x(:),y(:),'y');
             set(relleno,'uicontextmenu',menu2)
             menu1=uimenu(menu2,'label','delete','callback','design2(''borrar''
)');
         end
     end
     i=i+1;
     nwall=nwall+1;
     hold on;
end

return

 %============================================================================
 %FUNCION CIRCULO
 %============================================================================

 function [] = circulo

% Con el ratón se marca el radio del circulo, al hacer el segundo click
% aparecerá el circulo y éste será de color negro.

    [x1,y1]=ginput(1); %coge las coordenadas del primer click
    [x2,y2]=ginput(1); %coge las coordenadas del primer click

    ud.punto=[x1 x2;y1 y2];
    x = ud.punto(1,1:2);
```

```
    y = ud.punto(2,1:2);
    r = sqrt(diff(x)^2 + diff(y)^2);%halla el radio

    npts = 1000;
    res = 2*pi/(npts-1);
    theta = 0:res:2*pi;

    xout = r*cos(theta) + x(1);
    yout = r*sin(theta) + y(1);
    hold on;

    c=plot(xout,yout,'k');
    set(c,'visible','on')
    set(c,'LineWidth',3,'color','k')
    set(c,'tag','circulo')

    menu=uicontextmenu;
    set(c,'uicontextmenu',menu)
    menu1=uimenu(menu,'label','delete','callback','design2(''borrar'')');
    menu2=uimenu(menu,'label','properties','userdata',c,'callback','design2(''p
ropiedades'')','separator','on');

    menu2=uicontextmenu;
    relleno=fill(xout,yout,'k');
    set(relleno,'uicontextmenu',menu2)
    menu1=uimenu(menu2,'label','delete','callback','design2(''borrar'')');

return

%=====================================================================
%FUNCION CURSOR
%=========================================================================

function [] = cursor

% Obtiene las funciones de la figura actual, y determina si el boton
% 'coordenadas' está activado, para así aplicar su función o no.

if get(gcbo,'value'),
    set(gcf,'Pointer','crosshair')
    set(gcf, 'windowbuttonmotionfcn', 'design2(''actualiza_cursor'')')
else
    Title = get(gca, 'Title');
    delete(Title)
    set(gcf,'windowbuttonmotionfcn','')
    set(gcf,'pointer','arrow')
end

return

%=====================================================================
%FUNCION ACTUALIZACURSOR
%=========================================================================

function [] = actualiza_cursor

%Muestra por pantalla las coordenadas donde está situado el cursor.

% el cursor es una cruz
set(gcf,'Pointer','crosshair');

% posición del cursor
pt = get(gca, 'CurrentPoint');
xInd = pt(1, 1);
yInd = pt(1, 2);
```

```
% comprueba sin está dentro de los límites del axes
xLim = get(gca, 'XLim');
yLim = get(gca, 'YLim');
if xInd < xLim(1) | xInd > xLim(2)
    title('Out of X limit ');
    return;
end
if yInd < yLim(1) | yInd > yLim(2)
    title('Out of Y limit ');
    return;
else

title(['X = ' num2str(xInd) ', Y = ' num2str(yInd)]);

end

return

%================================================================
%FUNCIÓN BORRAR
%================================================================================

function [] = borrar

%borra la linea seleccionada.

delete(gco)

return

%================================================================
%FUNCION PROPIEDADES
%================================================================================

 function [] = propiedades

%crea la ventana de propiedades
fig=figure;
set(fig,'tag','fig2')
set(fig,'interruptible','off')
set(fig,'colormap',jet(256))
set(fig,'renderer','zbuffer')
set(fig,'menubar','none')
set(fig,'numbertitle','off')
set(fig,'name','Materials properties')
set(fig,'resize','off')
set(fig,'DefaultUIControlFontName','default')
set(fig,'DefaultUIControlFontSize',8)
set(fig,'DefaultUIControlFontAngle','normal')
set(fig,'DefaultUIControlFontWeight','light')
set(fig,'DefaultUIControlForeGroundColor','k')
set(fig,'DefaultUIControlInterruptible','off')
set(fig,'DefaultUIControlBusyAction','queue')
set(fig,'pointer','arrow')
set(fig,'color',[0.8 0.8 0.8])
delete(gca)

%================================================================================
%crea los botones y las listas
%================================================================================

h=uicontrol('Style','text');
set(h,'Position',[32 352 119 18])
set(h,'String','Select material')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])
```

```
h=uicontrol('style','popupmenu');
set(h,'position',[35 319 254 18])
set(h,'value',1)
load ('materials_list.mat')
bienhecho=[];
for ii=1:length(materials_list);
    bienhecho=[bienhecho materials_list(ii).nombrematerial ' | '];
    ii=ii+1;
end
set(h,'string',bienhecho)
set(h,'FontSize',8.0)
set(h,'tooltipstring','Materials')
set(h,'tag','materials_list')
set(h,'callback','design2(''materiales'')')

h=uicontrol('Style','text');
set(h,'Position',[242 352 309 15])
set(h,'String','Absorption coefficient(alfa):')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 319 52 15])
set(h,'String','125 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 269 52 15])
set(h,'String','250 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 219 52 15])
set(h,'String','500 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 169 52 15])
set(h,'String','1000 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 119 52 15])
set(h,'String','2000 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 69 52 15])
set(h,'String','4000 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[368 319 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_125')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])
```

```
h=uicontrol('Style','text');
set(h,'Position',[368 269 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_250')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[368 219 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_500')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[368 169 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_1000')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[368 119 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_2000')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[368 69 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_4000')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('style','pushbutton');
set(h,'position',[189 72 95 23])
set(h,'value',0)
set(h,'string','Accept')
set(h,'FontSize',8.0)
set(h,'tooltipstring','Exit')
set(h,'callback','design2(''cerrar'')')

return

%=========================================================================
%FUNCION MATERIALES
%=========================================================================

function [] = materiales

% según sea el material que interese la linea se verá de un color
% diferente.
load ('materials_list.mat')
ha=gco(1); % INDICA LA PANTALLA, Y DEPENDERÁ DE LAS PANTALLAS QUE HAYAN
ABIERTAS
hb=findobj('tag','materials_list','parent',gcbf);
hc=findobj('tag','coeficiente_125','parent',gcbf);
hd=findobj('tag','coeficiente_250','parent',gcbf);
he=findobj('tag','coeficiente_500','parent',gcbf);
hf=findobj('tag','coeficiente_1000','parent',gcbf);
```

```
hg=findobj('tag','coeficiente_2000','parent',gcbf);
hh=findobj('tag','coeficiente_4000','parent',gcbf);
seleccion = get(hb,'Value');
for ii=1:length(materials_list)
switch seleccion;
    case ii
        set(ha,'color',materials_list(ii).color)
        set(ha,'LineWidth',5) %OJO
        set(hc,'string',materials_list(ii).alfa(1))
        set(hd,'string',materials_list(ii).alfa(2))
        set(he,'string',materials_list(ii).alfa(3))
        set(hf,'string',materials_list(ii).alfa(4))
        set(hg,'string',materials_list(ii).alfa(5))
        set(hh,'string',materials_list(ii).alfa(6))
        ii=ii+1;
end
end

return

%=========================================================================
%FUNCION CERRAR
%=========================================================================

function [] = cerrar

% Cierra la pantalla, en este caso se ejecutará cuando presionamos 'aceptar'
%en la pantalla de propiedades.

    close;

return

%=========================================================================
%FUNCION SAVECONF
%=========================================================================

function [] = save_conf

 pregunta=questdlg ('Do you want to save?','WARNING','Yes','no','no');
ans=char;
if strcmp (pregunta, 'no')
    return;
end

grid off;
name=uiputfile('sc_*.mat','Design file name');

if sum(name(1:3)=='sc_')==3;
    name=name(4:end);
end


set(gca,'xtick',[])
set(gca,'ytick',[])
load grosor.mat;

ejes=axis;
res=grosor;
dimension_mayor=max(ejes(2)-ejes(1),ejes(4)-ejes(3));
disp(['resolucion= ' num2str(round(res/15*dimension_mayor))])
print(gcf,'planta','-dtiff', ['-r' num2str(round(res/15*dimension_mayor))],'-
noui') %Nuevo
        espera=waitbar(.25,'Please wait');
dib0=imread('planta.tif');
```

```
delete grosor.mat
a=double(dib0(:,:,1)); b=double(dib0(:,:,2)); c=double(dib0(:,:,3));
size(a)
size(b)
size(c)
        waitbar(.5,espera,'Please wait')
ymin=1+floor(find(a==0,1,'first')/size(a,1));
xmin=round((find(a==0,1,'first')/size(a,1)-(ymin-1)).*size(a,1));
xmin=xmin+1;ymin=ymin+1;
ymax=find(a(xmin,:)==0,1,'last')-1;
xmax=find(a(:,ymin)==0,1,'last')-1;
a=a(xmax:-1:xmin,ymin:ymax)';
b=b(xmax:-1:xmin,ymin:ymax)';
c=c(xmax:-1:xmin,ymin:ymax)';
size(a)
size(b)
size(c)
save(['sc_' name],'a','b','c','ejes','res')
delete planta.tif
    waitbar(.75,espera,'Please wait')
close(espera)
close

return
```

FUNCIÓN DESIGN.

```
%=========================================================================
%VIRTUAL ROOM ACOUSTICS
%Eugenio Garcés Leante
%=========================================================================

%=========================================================================
%FUNCIÓN DESIGN
%=========================================================================
%La función "design" o "new project" abre una ventana para crear desde cero
%un nuevo diseño.
%Los datos de entrada son:resolución(dependiendo de las frecuencias a
%simular) y tamaño del diseño(afectará a la resolución).
%=========================================================================

function [] = design(arg1,arg2)




%=========================================================================
%Si no existe materials_list.mat el programa no puede trabajar
%=========================================================================

if(length(dir('materials_list.mat'))<1)
    errordlg('"materials_list" not found','WARNING!','on');

    return

end
load ('materials_list.mat');
if length (materials_list)==0,


errordlg('"Materials_list" is empty','WARNING!','on');
return
end


%=========================================================================
%CREA UN NUEVO CASE
%=========================================================================

if ~nargin,
 design('init');
 return
end
%=========================================================================
%crea un nuevo case y aplica las propiedades dadas en arg1
%=========================================================================

if nargin == 1,
 if isstruct(arg1),
   update(arg1)
  return
 end
end

%=========================================================================
%actualiza el case actual con las propiedades de arg2
%=========================================================================

if nargin == 2,
 if isstruct(arg2) && strcmpi(arg1,'update'),
```

```
   update(arg2)
  return
 end
end

%=========================================================================
%selecciona funcion (depending on arg1)
%=========================================================================

switch lower(arg1),

    case 'init'
        init
    case 'linea'
        linea
    case 'circulo'
        circulo
    case 'borrar'
        borrar
    case 'propiedades'
        propiedades
        delete(gca)
    case 'materiales'
        materiales
    case 'cerrar'
        cerrar
    case 'save_conf'
        save_conf
    case 'cursor'
        cursor
    case 'actualiza_cursor'
        actualiza_cursor
    case 'grosor_50'
        grosor_50
    case 'grosor_100'
        grosor_100
    case 'grosor_200'
        grosor_200
    case 'plotcoor'
        plotcoor
    case 'fillo'
        fillo

end

%=========================================================================
%FUNCIÓN PRINCIPAL
%=========================================================================

function [] = init
   % crea la ventana (2) de resolución
% mediante un cuadro de diálogo se pregunta al usuario cuánta va a ser la
% resolución que se requiere, dependiendo del intervalo de frecuencias en el
% que se vaya a simular

hfig=figure;
set(hfig,'units','pixels')
set(hfig,'position',[320 300 400 250])
set(hfig,'interruptible','off')
set(hfig,'colormap',jet(256))
set(hfig,'renderer','zbuffer')
set(hfig,'menubar','none')
set(hfig,'numbertitle','off')
set(hfig,'resize','off')
set(hfig,'name','Resolution')
set(hfig,'color',[0.8 0.8 0.8])
```

```
t=uicontrol('style','text');
set(t,'position',[61 200 290 15])
set(t,'string','Which is the maximum frequency range to be simulated?')
set(t,'backgroundcolor',[0.878 0.875 0.89])

p1=uicontrol('style','checkbox');
set(p1,'position',[100 150 136 25])
set(p1,'string','Low freq. (fast)')
set(p1,'backgroundcolor',[0.878 0.875 0.89])
set(p1,'callback','design(''grosor_50'')')
set(p1,'Value',0);

p2=uicontrol('style','checkbox');
set(p2,'position',[100 100 136 25])
set(p2,'string','Medium freq.')
set(p2,'backgroundcolor',[0.878 0.875 0.89])
set(p2,'callback','design(''grosor_100'')')

p3=uicontrol('style','checkbox');
set(p3,'position',[100 50 136 25])
set(p3,'string','High freq.(slow)')
set(p3,'backgroundcolor',[0.878 0.875 0.89])
set(p3,'callback','design(''grosor_200'')')

h=uicontrol('style','pushbutton');
set(h,'position',[270 30 95 23])
set(h,'value',0)
set(h,'string','Ok')
set(h,'FontSize',8.0)
set(h,'tag','Ok')
set(h,'backgroundcolor',[0.878 0.875 0.89])
set(h,'callback','design(''cerrar'')')



uiwait(hfig)

%=========================================================================
%Por si no se elige ninguna frecuencia.
%=========================================================================

if(length(dir('grosor.mat'))<1)
return;
end

%=========================================================================
%crea la ventana principal
%=========================================================================

hfig=figure;
set(hfig,'tag','fig')
set(hfig,'interruptible','off')
set(hfig,'colormap',jet(256))
set(hfig,'renderer','zbuffer')
set(hfig,'menubar','none')
set(hfig,'numbertitle','off')
set(hfig,'name','Draw the room shape')

set(hfig,'resize','off')
set(hfig,'units','pixels')
set(hfig,'DefaultUIControlFontName','default')
set(hfig,'DefaultUIControlFontSize',8)
set(hfig,'DefaultUIControlFontAngle','normal')
set(hfig,'DefaultUIControlFontWeight','light')
set(hfig,'DefaultUIControlForeGroundColor','k')
```

```
set(hfig,'DefaultUIControlInterruptible','off')
set(hfig,'DefaultUIControlBusyAction','queue')
set(hfig,'DefaultUIControlbackgroundcolor',[0.878 0.875 0.89])
set(hfig,'color',[0.8 0.8 0.8])

scrsz = get(0, 'ScreenSize');
pos_act=get(hfig,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/3.5);
yr=scrsz(4) - pos_act(4);
yp=round(yr/7);

set(hfig,'Position',[xp-50 yp-10 895 708])

%================================================================================
%crea el axes (panel de trabajo)
%================================================================================

h=axes;
set(h,'units','pixels','position',[100 78 499 499])
set(h,'tag','axes')
set(h,'box','on')
set(h,'xtick',[])
set(h,'ytick',[])
set(h,'nextplot','replace')
set(h,'hittest','on')
grid on

%================================================================================
% pantalla de inicio (QUIERO QUE SEA LA SEGUNDA PANTALLA DE INICIO)
%================================================================================

prompt={'X size','Y size'};
titulo = 'Size';
num_lineas = 1;
def = {'10','10'};

answer = inputdlg(prompt,titulo,num_lineas,def);

if isempty(answer)
 close (hfig);
 return

end




xdi=str2double(answer{1});
ydi=str2double(answer{2});


if isnan (xdi)

    close (hfig);
    return
end%isnan

if isnan (ydi)

    close (hfig);
    return
end%isnan
```

```
ax = get(hfig,'CurrentAxes');
if isempty(ax)
   ax = axes('Parent',hfig);
end
set(ax,'Xlim',[0 xdi])
set(ax,'Ylim',[0 ydi])


r=plot(xdi,ydi);
set(r,'tag','rejilla')
set(r,'markersize',4)
set(r,'hittest','off')
grid on
axis equal

%=========================================================================
%crea los botones
%=========================================================================

h=uicontrol('style','pushbutton');
set(h,'position',[200 10 75 25])
set(h,'string','__')
set(h,'tag','linea')
set(h,'tooltipstring','Plot lines')
set(h,'callback','design(''linea'')')

h=uicontrol('style','pushbutton');
set(h,'position',[300 10 75 25])
set(h,'string','O')
set(h,'tooltipstring','Plot a circle')
set(h,'callback','design(''circulo'')')

h=uicontrol('style','pushbutton');
set(h,'position',[400 10 75 25])
set(h,'string','PlotCoor')
set(h,'tooltipstring','Plot with coordinates')
set(h,'callback','design(''plotcoor'')')

h=uicontrol('style','pushbutton');
set(h,'position',[500 10 75 25])
set(h,'string','Fill')
set(h,'tooltipstring','Fill(PlotCoor)')
set(h,'callback','design(''fillo'')')

h=uicontrol('style','pushbutton');
set(h,'position',[600 10 75 25])
set(h,'string','Save')
set(h,'FontSize',8.0)
set(h,'tooltipstring','Save')
set(h,'callback','design(''save_conf'')')

h=uicontrol('style','togglebutton');
set(h,'position',[100 10 75 25])
set(h,'string','Coordinates')
set(h,'tag','coordenates')
set(h,'value',0)
set(h,'FontSize',8.0)
set(h,'tooltipstring','Show cursor position')
set(h,'callback','design(''cursor'')')


set(ax,'xlim',[0 xdi],'ylim',[0 ydi]);

load ('materials_list.mat')
for ii=1:length(materials_list),
```

```
    for jj=640:-40:40,

        hmat=uicontrol('style','text');
        set(hmat,'position',[625 jj 154 21])
        set(hmat,'string',materials_list(ii).nombrematerial)
        set(hmat,'backgroundcolor',[0.878 0.875 0.89])

        cmat=uicontrol('style','text');
        set(cmat,'position',[800 jj 52 21])
        set(cmat,'backgroundcolor',materials_list(ii).color)

       ii=ii+1;
        if (ii>length(materials_list))
            return
        end
    end

end




return

%=================================================================
%DIBUJAR CON COORDENADAS.
%=========================================================================

    function [] = plotcoor


        if(length(dir('figura.mat'))<1)

    m=1;

        else  load('figura.mat');
            m=length(datosx)+1;

end



prompt={'x start value','y start value','x final value',' y final value'};
titulo = 'Coordinates';
num_lineas = 1;
def = {'1','1','1','1'};
answer = inputdlg(prompt,titulo,num_lineas,def);
if isempty(answer),
 return

end

ix=str2double(answer{1});
iy=str2double(answer{2});
fx=str2double(answer{3});
fy=str2double(answer{4});
wall=line([ix fx],[iy fy],'Color','k');
hold on;
set(wall,'LineWidth',5)%ojo
set(wall,'hittest','on')
set(wall,'tag','wall')
         menu=uicontextmenu;
         set(wall,'uicontextmenu',menu)
```

```
         menu1=uimenu(menu,'label','delete','callback','design(''borrar'')');
         menu2=uimenu(menu,'label','properties','userdata',wall,'callback','des
ign(''propiedades'')','separator','on');
datosx(m)=ix;
datosy(m)=iy;



m=m+1;
datosx(m)=fx;
datosy(m)=fy;

m=m+1;
if(length(dir('figura.mat'))<1)
   save('figura.mat');
end

save('figura.mat','datosx','datosy','-append');

   return%fucntion plot coor

%=========================================================================
%FUNCION RELLENAR
%=========================================================================

        function []=fillo

            load('figura.mat');


             menu2=uicontextmenu;
            axis manual;
           relleno= fill(datosx,datosy,'y');
            set(relleno,'uicontextmenu',menu2)
             menu1=uimenu(menu2,'label','delete','callback','design(''borrar'')
');
            hold on;


            delete('figura.mat');
            return

%=========================================================================
%FUNCION LINEA
%=========================================================================

function [] = linea

% Dibuja una linea entre dos puntos. Al hacer la geometría, el recinto se
% cerrará presionando la tecla espacio, y el interior se rellenará de
% amarillo.

button=1;
nwall=-1;
i=1;
while button==1,%es pulsar sobre el dibujo
     [xp,yp,button]=ginput(1)
     if button==3,%es boton derecho

         return;
     end
     x(i)=xp;
     y(i)=yp;
     if button==32,%pulsar espacio
         x(i)=x(1);
         y(i)=y(1);
```

```
     end
     if(nwall >= 0),
         wall=plot(x,y,'k');
         num=[x;y];
         set(wall,'LineWidth',5)%ojo
         set(wall,'hittest','on')
         set(wall,'x',[x(i-1) x(i)])
         set(wall,'y',[y(i-1) y(i)])
         set(wall,'tag','wall')
         menu=uicontextmenu;
         set(wall,'uicontextmenu',menu)
         menu1=uimenu(menu,'label','delete','callback','design(''borrar'')');
         menu2=uimenu(menu,'label','properties','userdata',wall,'callback','des
ign(''propiedades'')','separator','on');

         if button==32,
             menu2=uicontextmenu;
             relleno=fill(x(:),y(:),'y');
             set(relleno,'uicontextmenu',menu2)
             menu1=uimenu(menu2,'label','delete','callback','design(''borrar'')
');
         end
     end
     i=i+1;
     nwall=nwall+1;
     hold on;
end

return

 %=========================================================================
 %FUNCION CIRCULO
 %=========================================================================

 function [] = circulo

% Con el ratón se marca el radio del circulo, al hacer el segundo click
% aparecerá el circulo y éste será de color negro.

    [x1,y1]=ginput(1); %coge las coordenadas del primer click
    [x2,y2]=ginput(1); %coge las coordenadas del primer click

    ud.punto=[x1 x2;y1 y2];
    x = ud.punto(1,1:2);
    y = ud.punto(2,1:2);
    r = sqrt(diff(x)^2 + diff(y)^2);%halla el radio

    npts = 1000;
    res = 2*pi/(npts-1);
    theta = 0:res:2*pi;

    xout = r*cos(theta) + x(1);
    yout = r*sin(theta) + y(1);
    hold on;

    c=plot(xout,yout,'k');
    set(c,'visible','on')
    set(c,'LineWidth',3,'color','k')
    set(c,'tag','circulo')

    menu=uicontextmenu;
    set(c,'uicontextmenu',menu)
    menu1=uimenu(menu,'label','delete','callback','design(''borrar'')');
    menu2=uimenu(menu,'label','properties','userdata',c,'callback','design(''pr
opiedades'')','separator','on');
```

```
    menu2=uicontextmenu;
    relleno=fill(xout,yout,'k');
    set(relleno,'uicontextmenu',menu2)
    menu1=uimenu(menu2,'label','delete','callback','design(''borrar'')');

return

%=====================================================================
%FUNCION CURSOR
%=====================================================================

function [] = cursor

% Obtiene las funciones de la figura actual, y determina si el boton
% 'coordenadas' está activado, para así aplicar su función o no.

if get(gcbo,'value'),
    set(gcf,'Pointer','crosshair')
    set(gcf, 'windowbuttonmotionfcn', 'design(''actualiza_cursor'')')
else
    Title = get(gca, 'Title');
    delete(Title)
    set(gcf,'windowbuttonmotionfcn','')
    set(gcf,'pointer','arrow')
end

return

%=====================================================================
%FUNCION ACTUALIZA CURSOR
%=====================================================================

function [] = actualiza_cursor

%Muestra por pantalla las coordenadas donde está situado el cursor.

% el cursor es una cruz
set(gcf,'Pointer','crosshair');

% posición del cursor
pt = get(gca, 'CurrentPoint');
xInd = pt(1, 1);
yInd = pt(1, 2);

% comprueba sin está dentro de los límites del axes
xLim = get(gca, 'XLim');
yLim = get(gca, 'YLim');
if xInd < xLim(1) | xInd > xLim(2)
    title('Out of X limit');
    return;
end
if yInd < yLim(1) | yInd > yLim(2)
    title('Out of Y limit');
    return;
else

title(['X = ' num2str(xInd) ', Y = ' num2str(yInd)]);

end

return
```

```
%=================================================================
%FUNCION BORRAR
%=================================================================================

function [] = borrar

%borra la linea seleccionada.

delete(gco)

return

%=================================================================
%FUNCION PROPIEDADES
%=================================================================================

 function [] = propiedades

%=================================================================================
%crea la ventana de propiedades
%=================================================================================

fig=figure;
set(fig,'tag','fig2')
set(fig,'interruptible','off')
set(fig,'colormap',jet(256))
set(fig,'renderer','zbuffer')
set(fig,'menubar','none')
set(fig,'numbertitle','off')
set(fig,'name','Materials properties')
set(fig,'resize','off')
set(fig,'DefaultUIControlFontName','default')
set(fig,'DefaultUIControlFontSize',8)
set(fig,'DefaultUIControlFontAngle','normal')
set(fig,'DefaultUIControlFontWeight','light')
set(fig,'DefaultUIControlForeGroundColor','k')
set(fig,'DefaultUIControlInterruptible','off')
set(fig,'DefaultUIControlBusyAction','queue')
set(fig,'pointer','arrow')
set(fig,'color',[0.8 0.8 0.8])
delete(gca)

%=================================================================================
%crea los botones y las listas
%=================================================================================

h=uicontrol('Style','text');
set(h,'Position',[32 352 119 15])
set(h,'String','Select material')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('style','popupmenu');
set(h,'position',[35 319 254 15])
set(h,'value',1)
load ('materials_list.mat')
bienhecho=[];
for ii=1:length(materials_list);
    bienhecho=[bienhecho materials_list(ii).nombrematerial ' | '];
    ii=ii+1;
end
set(h,'string',bienhecho)
set(h,'FontSize',8.0)
set(h,'tooltipstring','Materiales')
set(h,'tag','materials_list')
set(h,'callback','design(''materiales'')')
```

```
h=uicontrol('Style','text');
set(h,'Position',[242 352 309 15])
set(h,'String','Absorption coefficient(alfa):')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 319 52 15])
set(h,'String','125 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 269 52 15])
set(h,'String','250 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 219 52 15])
set(h,'String','500 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 169 52 15])
set(h,'String','1000 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 119 52 15])
set(h,'String','2000 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[322 69 52 15])
set(h,'String','4000 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[368 319 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_125')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[368 269 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_250')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[368 219 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_500')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])
```

```
h=uicontrol('Style','text');
set(h,'Position',[368 169 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_1000')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[368 119 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_2000')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('Style','text');
set(h,'Position',[368 69 52 15])
set(h,'String','')
set(h,'FontSize',8.0)
set(h,'tag','coefficient_4000')
set(h,'enable','inactive')
set(h,'backgroundcolor',[0.878 0.875 0.89])

h=uicontrol('style','pushbutton');
set(h,'position',[189 72 95 23])
set(h,'value',0)
set(h,'string','Accept')
set(h,'FontSize',8.0)
set(h,'tooltipstring','Exit')
set(h,'callback','design(''cerrar'')')

return

%=========================================================================
%FUNCION MATERIALES
%=========================================================================

function [] = materiales

% según sea el material que interese la linea se verá de un color
% diferente.
load ('materials_list.mat')
ha=gco(1); % INDICA LA PANTALLA, Y DEPENDERÁ DE LAS PANTALLAS QUE HAYAN
ABIERTAS
hb=findobj('tag','materials_list','parent',gcbf);
hc=findobj('tag','coeficiente_125','parent',gcbf);
hd=findobj('tag','coeficiente_250','parent',gcbf);
he=findobj('tag','coeficiente_500','parent',gcbf);
hf=findobj('tag','coeficiente_1000','parent',gcbf);
hg=findobj('tag','coeficiente_2000','parent',gcbf);
hh=findobj('tag','coeficiente_4000','parent',gcbf);
seleccion = get(hb,'Value');
for ii=1:length(materials_list)
switch seleccion;
    case ii
        set(ha,'color',materials_list(ii).color)
        set(ha,'LineWidth',5) %OJO
        set(hc,'string',materials_list(ii).alfa(1))
        set(hd,'string',materials_list(ii).alfa(2))
        set(he,'string',materials_list(ii).alfa(3))
        set(hf,'string',materials_list(ii).alfa(4))
        set(hg,'string',materials_list(ii).alfa(5))
        set(hh,'string',materials_list(ii).alfa(6))
        ii=ii+1;
```

```
end
end

return

%=========================================================================
%FUNCION CERRAR
%=========================================================================

function [] = cerrar

% Cierra la pantalla, en este caso se ejecutará cuando presionamos 'aceptar'
%en la pantalla de propiedades.

    close;

return

%=========================================================================
%GROSOR
%=========================================================================

%=========================================================================
function [] = grosor_50

    grosor=50;
    save('grosor.mat','grosor')

return
%=========================================================================
function [] = grosor_100

    grosor=100;
    save('grosor.mat','grosor')

return
%=========================================================================
function [] = grosor_200

    grosor=200;
    save('grosor.mat','grosor')

return

%=========================================================================
%FUNCION SAVECONF
%=========================================================================

function [] = save_conf

pregunta=questdlg ('Do you want to save?','WARNING','yes','no','no');
ans=char;
if strcmp (pregunta, 'no')
    return;
end

grid off;
name=uiputfile('sc_*.mat','Design file name');


if sum(name(1:3)=='sc_')==3;
    name=name(4:end);
end

load grosor.mat
res=grosor;
```

```
set(gca,'xtick',[])
set(gca,'ytick',[])


ejes=axis;
dimension_mayor=max(ejes(2)-ejes(1),ejes(4)-ejes(3));
disp(['resolucion= ' num2str(round(res/15*dimension_mayor))])
print(gcf,'planta','-dtiff', ['-r' num2str(round(res/15*dimension_mayor))],'-
noui') %Nuevo
        espera=waitbar(.25,'Please wait');
dib0=imread('planta.tif');

delete planta.tif
delete grosor.mat
a=double(dib0(:,:,1)); b=double(dib0(:,:,2)); c=double(dib0(:,:,3));
size(a)
size(b)
size(c)
        waitbar(.5,espera,'Please wait')
ymin=1+floor(find(a==0,1,'first')/size(a,1));
xmin=round((find(a==0,1,'first')/size(a,1)-(ymin-1)).*size(a,1));
xmin=xmin+1;ymin=ymin+1;
ymax=find(a(xmin,:)==0,1,'last')-1;
xmax=find(a(:,ymin)==0,1,'last')-1;
a=a(xmax:-1:xmin,ymin:ymax)';
b=b(xmax:-1:xmin,ymin:ymax)';
c=c(xmax:-1:xmin,ymin:ymax)';
size(a)
size(b)
size(c)
save(['sc_' name],'a','b','c','ejes','res')
    waitbar(.75,espera,'Please wait')
close(espera)
close

return
```

FUNCIÓN MATERIALS.

```
%=========================================================================
%VIRTUAL ROOM ACOUSTICS
%Eugenio Garcés Leante
%=========================================================================

%=========================================================================
%FUNCIÓN MATERIALS
%=========================================================================
%Visualiza y/o borra los elementos de la lista de materiales
%materials_list.mat
%=========================================================================

function [] = materials(arg1,arg2)

%=========================================================================
%Si no existe materials_list.mat el programa no puede trabajar
%=========================================================================

if(length(dir('materials_list.mat'))<1)
    errordlg('"materials_list" not found','WARNING!','on');

    return

end

%=========================================================================
%Comprueba si hay materiales en la lista.
%=========================================================================

load materials_list.mat

if length (materials_list)==0,

 errordlg('"materials_list" is empty','WARNING!','on');

return
end

%=========================================================================
%CREACIÓN DE LOS CASE
%=========================================================================

if ~nargin,
 materials('init');
 return
end

%=========================================================================
%Crea un nuevo case y aplica las propiedades dadas en arg1.
%=========================================================================

if nargin == 1,
 if isstruct(arg1),
   update(arg1)
  return
 end
end

%=========================================================================
%Actualiza el case actual con las propiedades de arg2
%=========================================================================

if nargin == 2,
```

```
 if isstruct(arg2) && strcmpi(arg1,'update'),
   update(arg2)
  return
 end
end

%=========================================================================
%Selecciona funcion
%=========================================================================

switch lower(arg1),
    case 'init'
        init
    case 'ver'
        ver
    case 'borrar'
        borrar
    case 'cerrarf2'
        cerrarf2

end

%=========================================================================
%FUNCIÓN INIT
%=========================================================================
%Crea la ventana y realiza las funciones
%=========================================================================

function [] = init

%=========================================================================
%Crea la ventana de propiedades
%=========================================================================
fig=figure;

scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);

set(fig,'Position',[xp*1.4 yp*2 pos_act(3)/2.2 pos_act(4)/2.3]);
set(fig,'tag','fig')
set(fig,'interruptible','off')
set(fig,'colormap',jet(256))
set(fig,'renderer','zbuffer')
set(fig,'menubar','none')
set(fig,'numbertitle','off')
set(fig,'name','Materials')
set(fig,'resize','off')
set(fig,'DefaultUIControlFontSize',8)
set(fig,'pointer','arrow')
set(fig,'color',[0.8 0.8 0.8])

h=uicontrol('Style','text');
set(h,'Position',[30 150 200 18])
set(h,'String','Material name')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.878 0.875 0.89])

%=========================================================================
%BOTONES
%=========================================================================
%Uno de los botones es del tipo popupmenu.De ahí que se utilice un "for"
%para recorrer la lista materials_list y mostrar los materiales en el
```

```
%botón.
%========================================================================

h=uicontrol('style','popupmenu');
set(h,'position',[2 125 254 18])
set(h,'value',1)
load ('materials_list.mat')
bienhecho=[];

for ii=1:(length(materials_list))-1;
    bienhecho=[bienhecho materials_list(ii).nombrematerial '|'];

    ii=ii+1;

end

a=length(materials_list);
bienhecho=[bienhecho materials_list(a).nombrematerial];


set(h,'string',bienhecho)
set(h,'FontSize',8.0)
set(h,'tooltipstring','Materials')
set(h,'tag','boton')


h=uicontrol('style','pushbutton');
set(h,'position',[25 10 80 50])
set(h,'string','Delete')
set(h,'FontSize',8.0)
set(h,'tooltipstring','Delete')
set(h,'callback','materials(''borrar'')')

h=uicontrol('style','pushbutton');
set(h,'position',[150 10 80 50])
set(h,'string','See')
set(h,'FontSize',8.0)
set(h,'tooltipstring','See')
set(h,'callback','materials(''ver'')')



return

%========================================================================
%FUNCIÓN BORRAR
%========================================================================
%Quita los elementos seleccionados de la lista materials_list
%========================================================================

function [] = borrar

%========================================================================
%Confirmación para borrar elemento
%========================================================================

ans=questdlg('Do you want to delete this material?','WARNING','Yes','No','No');
if strcmp(ans,'No')
return;
end

%========================================================================
%Borrado
%========================================================================

hb=findobj('tag','boton','parent',gcbf);
```

```
seleccion = get(hb,'Value');
load materials_list.mat

close;


for ii=1:length(materials_list),
    switch seleccion;
    case ii

        materials_list(ii)=[];%borra la selección
    end

end

materials_list=[materials_list];
save('materials_list.mat','materials_list');

%=========================================================================
%Mensaje al borrar todos los materiales de la lista.
%Si,después de borrar un material,todavía quedarán materiales;la función
%borrar vueleve a llamar a INIT por si se quieren borrar más materiales
%=========================================================================

if length(materials_list)==0

    errordlg('Empty list','WARNING!','on');
else
materials('init');
end

%=========================================================================
%FUNCIÓN VER
%=========================================================================
%Muestra una pantalla con las características del material seleccionado
%=========================================================================

function []= ver

hb1=findobj('tag','boton','parent',gcbf);
s1 = get(hb1,'Value');
load materials_list.mat




fig2=figure;

scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(fig2,'Position',[xp yp pos_act(3) pos_act(4)]);
set(fig2,'tag','fig')
set(fig2,'interruptible','off')
set(fig2,'colormap',jet(256))
set(fig2,'renderer','zbuffer')
set(fig2,'menubar','none')
set(fig2,'numbertitle','off')
set(fig2,'name','Materials database')
set(fig2,'resize','off')
set(fig2,'DefaultUIControlFontName','default')
set(fig2,'DefaultUIControlFontSize',8)
set(fig2,'DefaultUIControlFontAngle','normal')
```

```
set(fig2,'DefaultUIControlFontWeight','light')
set(fig2,'DefaultUIControlForeGroundColor','k')
set(fig2,'DefaultUIControlInterruptible','off')
set(fig2,'DefaultUIControlBusyAction','queue')
set(fig2,'DefaultUIControlbackgroundcolor',[0.878 0.875 0.89])
set(fig2,'pointer','arrow')
set(fig2,'color',[0.8 0.8 0.8])
delete(gca)

hh=uicontrol('Style','text');
set(hh,'Position',[32 352 119 18])
set(hh,'String','Material name')
set(hh,'FontSize',8.0)
set(hh,'backgroundcolor',[0.878 .875 0.89])

hh=uicontrol('style','text');
set(hh,'position',[35 319 252 18])
set(hh,'string',materials_list(s1).nombrematerial)
set(hh,'FontSize',8.0)
set(hh,'tag','nuevo_material')
set(hh,'backgroundcolor',[0.878 0.875 0.89])

hh=uicontrol('Style','text');
set(hh,'Position',[242 352 309 18])
set(hh,'String','Absorption coefficient')
set(hh,'FontSize',8.0)
set(hh,'backgroundcolor',[0.878 .875 0.89])

hh=uicontrol('Style','text');
set(hh,'Position',[322 319 52 16])
set(hh,'String','125 Hz:')
set(hh,'FontSize',8.0)
set(hh,'backgroundcolor',[0.878 .875 0.89])

hh=uicontrol('Style','text');
set(hh,'Position',[322 269 52 16])
set(hh,'String','250 Hz:')
set(hh,'FontSize',8.0)
set(hh,'backgroundcolor',[0.878 .875 0.89])

hh=uicontrol('Style','text');
set(hh,'Position',[322 219 52 16])
set(hh,'String','500 Hz:')
set(hh,'FontSize',8.0)
set(hh,'backgroundcolor',[0.878 .875 0.89])

hh=uicontrol('Style','text');
set(hh,'Position',[322 169 52 16])
set(hh,'String','1000 Hz:')
set(hh,'FontSize',8.0)
set(hh,'backgroundcolor',[0.878 .875 0.89])

hh=uicontrol('Style','text');
set(hh,'Position',[322 119 52 16])
set(hh,'String','2000 Hz:')
set(hh,'FontSize',8.0)
set(hh,'backgroundcolor',[0.878 .875 0.89])

hh=uicontrol('Style','text');
set(hh,'Position',[322 69 52 16])
set(hh,'String','4000 Hz:')
set(hh,'FontSize',8.0)
set(hh,'backgroundcolor',[0.878 .875 0.89])

n=1;
for jj=319:-50:69,
```

```
    hh=uicontrol('Style','text');
    set(hh,'Position',[368 jj 52 16])
    set(hh,'String',materials_list(s1).alfa(n))
    set(hh,'FontSize',8.0)
    set(hh,'enable','inactive')
    set(hh,'backgroundcolor',[0.878 .875 0.89])
    n=n+1;
end

hh=uicontrol('style','text');
set(hh,'position',[32 250 119 18])
set(hh,'string','Material colour')
set(hh,'backgroundcolor',[0.878 .875 0.89])

c=uicontrol('style','text');
set(c,'position',[170 250 52 21])
set(c,'backgroundcolor',materials_list(s1).color)

h=uicontrol('style','pushbutton');
set(h,'position',[450 15 100 50])
set(h,'value',1)
set(h,'string','Ok')
set(h,'FontSize',8.0)
set(h,'tooltipstring','Close')
set(h,'callback','materials(''cerrarf2'')')

%=========================================================================
%Gráfica del coeficiente de absorción Vs frecuencia
%=========================================================================

x=[125 250 500 1000 2000 4000];
coef1=materials_list(s1).alfa(1);
coef2=materials_list(s1).alfa(2);
coef3=materials_list(s1).alfa(3);
coef4=materials_list(s1).alfa(4);
coef5=materials_list(s1).alfa(5);
coef6=materials_list(s1).alfa(6);
y=[coef1 coef2 coef3 coef4 coef5 coef6];

 subplot (2,2,3)
 p= plot(x,y);
set(p,'Linewidth',3)
axis normal;


title('Absorption graph');

xlabel('Frequency');
ylabel('Alfa');



return%function end

function []=cerrarf2
close;
close;
materials('init')

return

return
%=========================================================================
```

FUNCIÓN NEW_MATERIALS.

```
%==========================================================================
%VIRTUAL ROOM ACOUSTICS
%Eugenio Garcés Leante
%==========================================================================

%==========================================================================
%FUNCIÓN NEW_MATERIALS
%==========================================================================
%Función para añadir nuevos materiales a la lista de materiales
%materials_list
%==========================================================================

function [] = new_materials(arg1,arg2)

%==========================================================================
%Si no existe materials_list.mat el programa no puede trabajar
%==========================================================================

if(length(dir('materials_list.mat'))<1)
    errordlg('"materials_list" not found','Warning!!','on');

    return

end

%==========================================================================
%CREACIÓN DE UN NUEVO CASE
%==========================================================================

if ~nargin,
 new_materials('init');
 return
end

%==========================================================================
%Crea un nuevo case y aplica las propiedades dadas en arg1
%==========================================================================

if nargin == 1,
 if isstruct(arg1),
   update(arg1)
  return
 end
end

%==========================================================================
%Actualiza el case actual con las propiedades de arg2
%==========================================================================

if nargin == 2,
 if isstruct(arg2) && strcmpi(arg1,'update'),
   update(arg2)
  return
 end
end

%==========================================================================
%Selecciona funcion (depending on arg1)
%==========================================================================

switch lower(arg1),
    case 'init'
        init
```

```
    case 'cerrar'
        cerrar

end

%========================================================================
%FUNCIÓN INIT
%========================================================================
%Crea la ventana y realiza las funciones para añadir un nuevo material
%========================================================================

function [] = init

%========================================================================
%Ventana con los INPUTDLG
%========================================================================

titulo = 'New materials';

prompt={'Name of material:','Absorption coefficient at 125 Hz:','Absorption
coefficient at 250 Hz:',...
        'Absorption coefficient at 500 Hz:','Absorption coefficient at 1000
Hz:',...
        'Absorption coefficient at 2000 Hz:','Absorption coefficient at 4000
Hz:'};

num_lineas = 1;
def = {'','','','','','',''};
options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';

answer = inputdlg(prompt,titulo,num_lineas,def,options);

%========================================================================
%Al hacer click sobre la opción CANCEL de la ventana de introducción de
%datos,se cierra la función NEW_MATERIALS.
%========================================================================

if isempty(answer),

 return
end

%========================================================================
%Se comprueba que la lista no sobrepase los quince materiales(todavía no se
%para que).
%Si no se le pone nombre al material,el programa pondrá uno por defecto.
%No se pueden repetir nombres.
%Si se deja vacío el casillero de coeficiente de absorción el programa
%pondrá por defecto un cero (de igual forma si lo que hemos introducido no
%es un número).
%Si el coeficiente de abs.que ponemos es mayor que 1,el programa pondrá un
%1).
%========================================================================

load materials_list.mat

ii=length(materials_list)+1;
if (ii<16)



    if isempty(answer{1})
        materials_list(ii).nombrematerial='material';
```

```
    else
    materials_list(ii).nombrematerial=(answer{1});
    end


    for i=1:length(materials_list)-1

            esigual=isequal(materials_list(ii).nombrematerial,materials_list(i)
.nombrematerial);

            if esigual==1

                errordlg('This name is already in use','WARNING','ok');
                return
            end%if

        end%for

    for ind=1:6

        a=str2double(answer{ind+1});

        if isempty(answer{ind+1})

            materials_list(ii).alfa(ind)=0;

        elseif a>1
            materials_list(ii).alfa(ind)=1;

        elseif isnan(a)
            materials_list(ii).alfa(ind)=0;
        else
            materials_list(ii).alfa(ind)=str2double(answer{ind+1})

        end%if
    end%for


    materials_list(ii).color=uisetcolor;
else
    errordlg('Too many materials in the database','¡¡Warning!!','on');

    return
end

materials_list = [materials_list];

%================================================================================
%Comprobación del uso de cada color
%================================================================================

for ii=1:length(materials_list);
    a(1:3,ii)=materials_list(ii).color;
    A(ii)=2^8*2^8*a(1,ii)+2^8*a(2,ii)+a(3,ii);
end
if length(find(diff(sort(A))==0))
    errordlg('The new material can''t be created because the color is already
in use','¡¡Warning!!','on');

    return
end
```

```
%=========================================================================
%Si todo ha ido correctamente,se guarda el nuevo elemento en la lista
%=========================================================================
save('materials_list.mat','materials_list');

%=========================================================================
%Crea una ventana que nos muestra el elemento que acabamos de añadir
%=========================================================================

fig=figure;
set(fig,'tag','fig')
set(fig,'interruptible','off')
set(fig,'colormap',jet(256))
set(fig,'renderer','zbuffer')
set(fig,'menubar','none')
set(fig,'numbertitle','off')
set(fig,'name','Materials database')
set(fig,'resize','off')
set(fig,'DefaultUIControlFontName','default')
set(fig,'DefaultUIControlFontSize',8)
set(fig,'DefaultUIControlFontAngle','normal')
set(fig,'DefaultUIControlFontWeight','light')
set(fig,'DefaultUIControlForeGroundColor','k')
set(fig,'DefaultUIControlInterruptible','off')
set(fig,'DefaultUIControlBusyAction','queue')
set(fig,'DefaultUIControlbackgroundcolor',[0.9961 0.7968 0.2226])
set(fig,'pointer','arrow')
set(fig,'color',[0.9765 0.6562 0.0976])
delete(gca)

h=uicontrol('Style','text');
set(h,'Position',[32 352 119 18])
set(h,'String','Material name')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.9765 0.6562 0.0976])

h=uicontrol('style','text');
set(h,'position',[35 319 252 18])
set(h,'string',materials_list(ii).nombrematerial)
set(h,'FontSize',8.0)
set(h,'tag','nuevo_material')
set(h,'backgroundcolor',[0.9961 0.7968 0.2226])

h=uicontrol('Style','text');
set(h,'Position',[242 352 309 18])
set(h,'String','Absorption coefficient:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.9765 0.6562 0.0976])

h=uicontrol('Style','text');
set(h,'Position',[322 319 52 16])
set(h,'String','125 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.9765 0.6562 0.0976])

h=uicontrol('Style','text');
set(h,'Position',[322 269 52 16])
set(h,'String','250 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.9765 0.6562 0.0976])

h=uicontrol('Style','text');
set(h,'Position',[322 219 52 16])
set(h,'String','500 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.9765 0.6562 0.0976])
```

```
h=uicontrol('Style','text');
set(h,'Position',[322 169 52 16])
set(h,'String','1000 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.9765 0.6562 0.0976])

h=uicontrol('Style','text');
set(h,'Position',[322 119 52 16])
set(h,'String','2000 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.9765 0.6562 0.0976])

h=uicontrol('Style','text');
set(h,'Position',[322 69 52 16])
set(h,'String','4000 Hz:')
set(h,'FontSize',8.0)
set(h,'backgroundcolor',[0.9765 0.6562 0.0976])

n=1;
for jj=319:-50:69,

    h=uicontrol('Style','text');
    set(h,'Position',[368 jj 52 16])
    set(h,'String',materials_list(ii).alfa(n))
    set(h,'FontSize',8.0)
    set(h,'enable','inactive')
    set(h,'backgroundcolor',[0.9961 0.7968 0.2226])
    n=n+1;
end

h=uicontrol('style','text');
set(h,'position',[32 250 119 18])
set(h,'string','Material colour:')
set(h,'backgroundcolor',[0.9765 0.6562 0.0976])

c=uicontrol('style','text');
set(c,'position',[170 250 52 21])
set(c,'backgroundcolor',materials_list(ii).color)

h=uicontrol('style','pushbutton');
set(h,'position',[450 15 100 50])
set(h,'value',1)
set(h,'string','Ok')
set(h,'FontSize',8.0)
set(h,'tooltipstring','exit from aplication')
set(h,'callback','new_materials(''cerrar'')')

%=========================================================================
%Gráfica del coeficiente de absorción Vs frecuencia
%=========================================================================

x=[125 250 500 1000 2000 4000];
coef1=materials_list(ii).alfa(1);
coef2=materials_list(ii).alfa(2);
coef3=materials_list(ii).alfa(3);
coef4=materials_list(ii).alfa(4);
coef5=materials_list(ii).alfa(5);
coef6=materials_list(ii).alfa(6);
y=[coef1 coef2 coef3 coef4 coef5 coef6];

subplot (2,2,3),plot(x,y,'linewidth',5)
axis([125 4000 0 1]);
title('Gráfica');
xlabel('Frecuencia');
ylabel('Coef.Abs');
```

```
return

% ========================================================================
%FUNCIÓN CERRAR
%========================================================================
function [] = cerrar

close;

return
```

FUNCIÓN OPCIONES.

```
%========================================================================
%VIRTUAL ROOM ACOUSTICS
%Eugenio Garcés Leante
%========================================================================

%========================================================================
%FUNCIÓN OPCIONES
%========================================================================
%Muestra una ventana
% con las opciones del programa y permite llamar a la función
%"MODIFICAOPCIONES",que es la que se encarga de cambiar y guardar dichas
%opciones.
%========================================================================
%========================================================================
%CREACIÓN DE LOS CASE
%========================================================================
function []= opciones (arg1,arg2)

%========================================================================
%La función necesita de un archivo de opciones,al principio de
%ejectutarse,comprueba que existe ese archivo;de no ser así lo crea
%automáticamente.

if (length(dir('optionsfile.mat'))<1)

    provis(1)=1;
    provis(2)=1;
    provis(3)=1;

    save('optionsfile.mat','provis');

end%if

if ~nargin,
 opciones('init');
 return
end

%========================================================================
%Crea un nuevo case y aplica las propiedades dadas en arg1.
%========================================================================

if nargin == 1,
 if isstruct(arg1),
   update(arg1)
  return
 end
end

%========================================================================
%Actualiza el case actual con las propiedades de arg2
%========================================================================

if nargin == 2,
 if isstruct(arg2) && strcmpi(arg1,'update'),
   update(arg2)
  return
 end
end
```

```
%=========================================================================
%Selecciona funcion
%=========================================================================

switch lower(arg1),
    case 'init'
        init
    case 'modificavideo'
        modificavideo
    case 'modificaaudio'
        modificaaudio
    case 'modificagraficas'
        modificagraficas


end

function []= init


load ('optionsfile.mat');
provis;
%=========================================================================
%Ventana principal.
%=========================================================================

f=figure;

scrsz = get(0, 'ScreenSize');
pos_act=get(f,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(gcf,'Position',[xp*1.4 yp*2 pos_act(3)/2.2 pos_act(4)/2.3]);
set(f,'units','pixels')

set(f,'interruptible','off')
set(f,'colormap',jet(256))
set(f,'renderer','zbuffer')
set(f,'menubar','none')
set(f,'numbertitle','off')
set(f,'name','Options')
set(f,'resize','off')
set(f,'color',[0.8 0.8 0.8])

%=========================================================================
%Botones
%=========================================================================

h=uicontrol ('style','radiobutton');
set(h,'position',[15 125  231 39])
set(h,'string','Generate video')
set(h,'tooltipstring','Generate video')
set(h,'Value',provis(1))
set(h,'callback','opciones(''modificavideo'')')

h=uicontrol ('style','radiobutton');
set(h,'position',[15 75  231 39])
set(h,'string','Generate audio')
set(h,'tooltipstring','Generate audio')
set(h,'Value',provis(2))
set(h,'callback','opciones(''modificaaudio'')')

h=uicontrol ('style','radiobutton');
set(h,'position',[15 25  231 39])
```

```
set(h,'string','Generate gráphs')
set(h,'tooltipstring','Generate graphs')
set(h,'Value',provis(3))
set(h,'callback','opciones(''modificagraficas'')')

return
%Llamada por la función "OPCIONES".Se encarga de guardar los valores de la
%configuración elegida por el usuario.Para ello se utiliza un vector con
%equis valores,cada valor corresponde a las dos posibilidades de cada
%opción.Cuando el valor es igual a 1,la opción está activada y cuando el
%valor es igual a 0,la opción está desactivada.
%El vector se llama "PROVIS",está guardado dentro el archivo
%"ARCHIVOPCIONES.MAT" y estos son los valores a los que hace referencia:
%   provis(1):posibilidad de generar un video con la representación de la
%             simulación.
%   provis(2):posibilidad de generar un archivo de audio de la
%             representación de la simulación.
%   provis(3):posibilidad de generar gráficas de tiempo de las señales

%=========================================================================


%=========================================================================
%MODIFICA VIDEO
%=========================================================================

function []=modificavideo

load optionsfile.mat;
a=provis(1);
b=not(a);
provis(1)=b;
save ('optionsfile.mat','provis');


return

%=========================================================================
%MODIFICA AUDIO
%=========================================================================

function []=modificaaudio

load optionsfile.mat;
a=provis(2);
b=not(a);
provis(2)=b;
save ('optionsfile.mat','provis');


return

%=========================================================================
%MODIFICA GRAFICAS
%=========================================================================

function []=modificagraficas

load optionsfile.mat;
a=provis(3);
b=not(a);
provis(3)=b;
save ('optionsfile.mat','provis');


retur
```

FUNCIÓN RESTORE_MATERIALS.

```
function [] = restore_materials

% Introduce los valores de los materiales por defecto

%=========================================================================
%CONFIRMACIÓN
%=========================================================================
pregunta=questdlg ('Do you want to restore
materials?','WARNING','yes','no','no');
ans=char;
if strcmp (pregunta, 'no')
    return;
end
materials_list=[];
materials_list = struct('nombrematerial',{'Unpaint concrete','Glass wool of
6cm','Wood','Very absorbent material','Fibre-plaster perforated panel','Nothing
absorbent material','Audience','Reinforced concrete','Paint concrete','Brick
with plaster'},'alfa',{[0.01 0.01 0.01 0.02 0.02 0.03] [0.09 0.39 0.61 0.74
0.83 0.87] [0.10 0.16 0.13 0.10 0.06 0.05] [1 1 1 1 1 1] [0.40 0.80 0.62 0.92
0.81 1] [0 0 0 0 0 0] [0.76 0.83 0.88 0.91 0.91 0.89 ] [0.02 0.02 0.02 0.03
0.04 0.04] [0.10 0.05 0.06 0.07 0.09 0.08] [0.02 0.02 0.02 0.03 0.04
0.04]},'color',{[0 0 0] [0 1 1] [1 0 1] [1 0 0] [0 0 1] [0 1 0] [0.5 0.5 0.5 ]
[1 0.5 0.3] [0 0.5 0] [0.8 0.2 0.7]});
save('materials_list.mat','materials_list');

return
```

FUNCIÓN SIMULATION.

```
%=========================================================================
%VIRTUAL ROOM ACOUSTICS
%Eugenio Garcés Leante
%=========================================================================

%=========================================================================
%FUNCIÓN SIMULATION
%=========================================================================
%Función principal del VI_AC.Se encarga de mostrar de manera gráfica la
%evolución del sonido en nuestro recinto y de proporcionar datos de la
%simulación.
%=========================================================================
%=========================================================================

%=========================================================================
function []=simulation

%=========================================================================
%La función necesita de un archivo de opciones,al principio de
%ejectutarse,comprueba que existe ese archivo;de no ser así lo crea
%automáticamente.

if (length(dir('optionsfile.mat'))<1)

    provis(1)=1;
    provis(2)=1;
    provis(3)=1;

    save('optionsfile.mat','provis');

end%if

%=========================================================================
%Si no existe materials_list.mat o esta vacia el programa no puede trabajar
%=========================================================================

if(length(dir('materials_list.mat'))<1)
    errordlg('"materials_list" not found','WARNING!','on');

    return

end
load ('materials_list.mat');
if length (materials_list)==0,

 errordlg('"Materials_list" is empty','WARNING!','on');
return
end

load('optionsfile.mat');

%=========================================================================
%Declaración del archivo .AVI si está en opcióan activada.
%=========================================================================

if provis(1)==1

nombre_video=0;
nombre_video=uiputfile('*.avi','Video file name','');

if nombre_video==0
    return
end
```

```
aviobj = avifile(nombre_video,'fps',5);
aviobj.Quality = 100;

end%if

%=========================================================================
%Abrir archivo sc_*.mat
%=========================================================================

name_file=0;
name_file=uigetfile('sc_*.mat','Design file name','sc_my_design.mat');

if name_file==0
return
end




%=========================================================================
%DATOS DE ENTRADA
%=========================================================================

prompt1={'Choose source type,[1]Impulse,[2]Pure tone,[3]Audio file'};
titulo = 'Source';
num_lines = 1;
%def = {'1'};
respuesta = inputdlg(prompt1,titulo,num_lines);



if isempty(respuesta)


    return
end

entrada=round(str2num(respuesta{1}));

if isnan(entrada)

    return
end

if entrada>3


    return
end

if entrada==0

    return
end

if isempty(entrada)


    return
end
```

```
%=========================================================================
%Matriz de diseño
%=========================================================================

load (name_file);
matriz=(256*256*a+256*b+(255-c));

mip=(matriz==(2^3^8-1))*1.000;
[nx,ny]=size(matriz);
dh=(ejes(2)-ejes(1))/nx;
rho=1.21;
c=341;
k=(c^2)*rho;
dt=dh/c/2 %media constants

%=========================================================================
%Tiempo de la simulación
%=========================================================================

if  entrada==1

prompt2={'Simulation time in mS','Frequency in Hz','Listeners number','Sources
number'};
dlg_title = 'Simulation properties';
num_lines = 1;
adef = {'50','1000','1','1'};
answer = inputdlg(prompt2,dlg_title,num_lines,adef);



as=str2double(answer{1});




if isempty(answer{1})
   return
end

if isnan(as)
    return
end



if isempty(answer{2})
   return
end
excitacion=str2double(answer{2});
if isnan(excitacion)
    return
end



numerolisteners=round(str2num(answer{3}));

if isempty (numerolisteners)
    numerolisteners=1;
end

if isnan(numerolisteners)
   numerolisteners=1;
end
```

```
if ((numerolisteners>3)|(numerolisteners<1))
numerolisteners=1;
end


numerosources=round(str2num(answer{4}));

if isempty (numerosources)
    numerosources=1;
end

if isnan(numerosources)
   numerosources=1;
end

if ((numerosources>3)|(numerosources<1))
numerosources=1;
end



end%if entrada==1

%=========================================================================
%=
if  entrada==2

prompt2={'Simulation time in mS','Active source time in mS','Frequency in
Hz','Listeners number','Sources number'};
dlg_title = 'Simulation properties';
num_lines = 1;
adef = {'50','50','1000','1','1'};
answer = inputdlg(prompt2,dlg_title,num_lines,adef);



as=str2double(answer{1});
tact=str2double(answer{2});




if isempty(answer{1})
   return
end

if isnan(as)
    return
end


if isempty(answer{2})
   return
end


if isnan(tact)
    return
end

if isempty(answer{3})
   return
end
excitacion=str2double(answer{3});
if isnan(excitacion)
    return
```

```
end



numerolisteners=round(str2num(answer{4}));

if isempty (numerolisteners)
    numerolisteners=1;
end

if isnan(numerolisteners)
   numerolisteners=1;
end

if ((numerolisteners>3)|(numerolisteners<1))
numerolisteners=1;
end

numerosources=round(str2num(answer{5}));

if isempty (numerosources)
    numerosources=1;
end

if isnan(numerosources)
   numerosources=1;
end

if ((numerosources>3)|(numerosources<1))
numerosources=1;
end

tact=tact/1000;



end%if entrada==2

%=========================================================================

if entrada==3


prompt3={'Simulation time in mS','Active source time in mS','Listeners
number','Sources number'};
titulo='Simulation properties';
num_lines = 1;
aadef = {'50','50','1','1'};
answer = inputdlg(prompt3,titulo,num_lines,aadef);

if isempty(answer(1))
   return
end
as=str2double(answer(1));
if isnan(as)
    return
end


if isempty(answer(2))
   return
end
tact=str2double(answer(2));
if isnan(tact)
    return
end
```

```
numerolisteners=round(str2num(answer{3}));

if isempty (numerolisteners)
    numerolisteners=1;
end

if isnan(numerolisteners)
   numerolisteners=1;
end

if ((numerolisteners>3)|(numerolisteners<1))
numerolisteners=1;

end

numerosources=round(str2num(answer{4}));

if isempty (numerosources)
    numerosources=1;
end

if isnan(numerosources)
   numerosources=1;
end

if ((numerosources>3)|(numerosources<1))
numerosources=1;
end

tact=tact/1000;


end%if entrada igyal a 3


%=========================================================================
%Cálculo de tiempo para simulación
%=========================================================================

maxtt=round(sqrt(nx.^2+ny.^2)/sqrt(2)/50);
maxttt=ceil(str2num(answer{1})/1000/(maxtt*dt));

%=========================================================================
%Cálculo de tiempo para fuente activa
%=========================================================================

maxttact=round(sqrt(nx.^2+ny.^2)/sqrt(2)/50);
maxtttact=ceil(str2num(answer{2})/1000/(maxtt*dt));

%=========================================================================


load materials_list
for ii=1:length(materials_list);colores(1:3,ii)=materials_list(ii).color;
end;

colores=colores*255;
for ii=1:length(materials_list);
    alfas=materials_list(ii).alfa;
    alfas(4);
    impedancias(ii)=(2*c*rho - c*rho*alfas(4) + 2*(c^2*rho^2 -
c^2*rho^2*alfas(4))^.5)/alfas(4);
end;

clear alfas;
%figure(2);pcolor(mip');shading flat;colormap gray;colorbar
```

```
%Busquedad de los puntos de contorno.
uxz=[];uyz=[];pxz=[];pyz=[];p0x=[] ;p0y=[] ;signox=[];signoy=[];Zx=[];Zy=[];
%Busquedad de los puntos de contorno de ux
[i,j]=find(diff(mip)==1);
uxz=[uxz (i+1+(j-1)*(nx+1))'];
pxz=[pxz (i+1+(j-1)*(nx))'];
p0x =[p0x  (i+(j-1)*(nx))'];
signox=[signox i'*0-1];

[i,j]=find(diff(mip)==-1);
uxz=[uxz (i+1+(j-1)*(nx+1))'];
pxz=[pxz (i+(j-1)*(nx))'];
p0x =[p0x  (i+1+(j-1)*(nx))'];
signox=[signox i'*0+1];

% figure;p=zeros(nx,ny);p(p0x)=1;p(pxz)=2;pcolor(p');shading flat;colormap(1-
gray);pause;close
% figure;ux=zeros(nx+1,ny);ux(uxz)=1;pcolor(ux');shading flat;colormap(1-
gray);pause;close


A=floor(matriz(p0x)/256^2);
B=floor((matriz(p0x)-A*256^2)/256);
C=255-(matriz(p0x)-A*256^2-B*256);
Zx=A*0+413;
for ii=1:length(materials_list)
    Zx(find((A==colores(1,ii)).*(B==colores(2,ii)).*(C==colores(3,ii))))=impeda
ncias(ii);
end

%Busquedad de los puntos de contorno de uy
[i,j]=find(diff(mip')'==1);
uyz=[uyz   (i+(j+1)*nx)'];
pyz=[pyz   (i+(j+1)*nx)'];
p0y= [p0y    (i+(j-1)*nx)'];
signoy=[signoy i'*0-1];

[i,j]=find(diff(mip')'==-1);
uyz=[uyz   (i+(j)*nx)'];
pyz=[pyz   (i+(j-1)*nx)'];
p0y= [p0y    (i+(j)*nx)'];
signoy=[signoy i'*0+1];
% figure;p=zeros(nx,ny);p(p0y)=1;p(pyz)=2;pcolor(p');shading flat;colormap(1-
gray);pause;close
% figure;uy=zeros(nx,ny+1);uy(uyz)=1;pcolor(uy');shading flat;colormap(1-
gray);pause;close

A=floor(matriz(p0y)/256^2);
B=floor((matriz(p0y)-A*256^2)/256);
C=255-(matriz(p0y)-A*256^2-B*256);
Zy=A*0+413;
for ii=1:length(materials_list)
    Zy(find((A==colores(1,ii)).*(B==colores(2,ii)).*(C==colores(3,ii))))=impeda
ncias(ii);
end
p0=[p0x p0y];clear p0x p0y


%Fin Busquedad de los puntos de contorno.
p=zeros(nx,ny);ux=zeros(nx+1,ny);uy=zeros(nx,ny+1);


clear a b i matriz
%clc
%disp(['Maximun frequency correctly simulated ' num2str(round(.1/dt*1000)/1000)
' Hz']);
```

```
centralfrequency=500;sourcex=round(nx/2);sourcey=round(ny/2);

%=========================================================================
%SOURCES
%=========================================================================




%=========================================================================
%Tono
%=========================================================================

if entrada==2

    t=((1:maxtt*maxttt)*dt);
    tact=((1:maxttact*maxtttact)*dt);



    if length(tact)>length(t)

        tact=t;

    end%%if tact


    [nad,samples]=size(t);%número de samples totales
    w=(1-exp(-tact*excitacion/2)).*sin(2*pi*tact*excitacion);
    [nada,sizew]=size(w);
    as=as/1000;
    fs=samples/as%samples por segundo BitRate
    fs=round(samples/as);
    nbits=16;

    if provis(2)==1

    wavwrite(w,fs,nbits,'originaltone.wav');

    end%if provis

    for i=(sizew+1):samples

        w(i)=0;

    end


end

%=========================================================================
%Impulso
%=========================================================================

if entrada==1

    a=excitacion/(sqrt(pi)/2)*4;
    t=((1:maxtt*maxttt)/(1/dt)-8/a);
    w=-(exp(-a^2*(t.^2)/2).*(a^2*(t.^2)-1));
    [nad,samples]=size(t);%número de samples totales
    as=as/1000;
    fs=samples/as%samples por segundo BitRate
    fs=round(samples/as);
    nbits=16;
```

```
    if provis(2)==1

    wavwrite(w,fs,nbits,'originalimpulse.wav');
    end%if provis

    clear a

end

%=========================================================================
%Señal
%=========================================================================

if entrada==3



name_file=0;
name_file=uigetfile('*.wav','Select audio file','');

if name_file==0
return
end



   t=((1:maxtt*maxttt)*dt);
   [nad,samples]=size(t);%samples totales
   as=as/1000;%tiempo en milisegundos
   fs=round(samples/as);%samples por segundo bit rate
   [y,Fo,bits] = wavread(name_file);%leo la señal
   y=y(:,1);%cogemos un sólo canal
   [sizey,n]=size(y);%m=samples,n=canales(1).

 tarchivo=sizey/Fo;

  if tact>tarchivo
      tact=tarchivo;
   end

   nuevotam=tact*Fo;

   for i=1:nuevotam

       if    i<=sizey

           yo(i)=y(i);

       else yo(i)=0;
       end%if

   end%for de tamaño
   y=yo;
   [n,sizey]=size(y)%m=samples,n=canales(1).

   fac=round(Fo/fs);



%=========================================================================

    if fac>1%diezmado
   for i=1:(samples)

       if (fac*i)<=sizey
```

```
            w(i)=y(fac*i);

        else  w(i)=0;


        end


    end



 nbits=16;
 if provis(2)==1

wavwrite(w,fs,nbits,'lowresampledfile.wav');%señal original resampleada
 end%if provis
   end

%=========================================================================

    if fac<1%sobremuestreo

        newfac=round(fs/Fo);

       for i=1:sizey


            for j=1:newfac

                w(j+newfac*(i-1))=y(i);

            end%for j
        end%for
      nbits=16;

      if provis(2)==1
    wavwrite(w,fs,nbits,'highresampledfile.wav');%señal original resampleada
      end%if provis

      [e,sizw]=size(w);

      for i=(sizw)+1:samples

          w(i)=0;
      end




end%if fac<1

%=========================================================================
if fac==1
    for i=1:(samples)

        if (i)<=sizey

            w(i)=y(i);

        else  w(i)=0;


        end
```

```
   end

 nbits=16;

 if provis(2)==1

wavwrite(w,fs,nbits,'equalresampledfile.wav');%señal original resampleada

 end%if provis

end


end

%=========================================================================
%h1=figure.SIMULATION
%=========================================================================

h1=figure;

scrsz = get(0, 'ScreenSize');
set(h1,'units','pixels');
set(h1,'position',[0 0 scrsz(3)-305 scrsz(4)  ]);
set(h1,'name','Simulation','menubar','none','color',[0.878 0.875 0.89])
pp=p;
pp(round(sourcex),round(sourcey))=1;
pp=min(max(pp.^2,1d-5),1);pp(p0)=1d-6;
pcolor(dh*(1:nx),dh*(1:ny),10*log10(pp'));
shading flat,axis equal;
colorbar;
title(['time= 0 miliseconds']);

drawnow
drawnow

for ii=1:numerosources
    figuraerror=msgbox(['Place the source'  num2str(ii) 'at design']);
    uiwait(figuraerror);
    [sourcex(ii),sourcey(ii)]=ginput(1);
    sourcex(ii)=round(sourcex(ii)/dh);
    sourcey(ii)=round(sourcey(ii)/dh);
end

for ii=1:numerolisteners
    figuraerror=msgbox(['Place the listener' num2str(ii) ' at design']);
    uiwait(figuraerror);
    [listenerx(ii),listenery(ii)]=ginput(1);
    listenerx(ii)=round(listenerx(ii)/dh);
    listenery(ii)=round(listenery(ii)/dh);
end



IR=zeros(length(t),numerolisteners);


%=========================================================================
%h2=figure.EXCITACIÓN Y RESPUESTA
%=========================================================================

h2=figure;
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
```

```
xp=round(xr/2);
yr=scrsz(4) - pos_act(4);
yp=round(yr/2);
set(h2,'Position',[scrsz(3)-300 0 300 scrsz(4)  ]);
set(h2,'units','pixels');
set(h2,'name','Gráphs','menubar','none','color',[0.878 0.875 0.89])



%=========================================================================
%PINTANDO
%=========================================================================

for ttt=1:maxttt;
    for tt=1:maxtt;
        contador=tt+maxtt*(ttt-1);
        %preassure calculation
        p=p-k*dt/dh*(diff(ux)+diff(uy')');
        p(p0)=0;
        %excitacionç
        for i=1:numerosources
        p(sourcex(i),sourcey(i))=w(contador);
        end

        %velocity calculation
        ux(2:nx,:)=ux(2:nx,:)-dt/rho/dh*diff(p);
        uy(:,2:ny)=uy(:,2:ny)-dt/rho/dh*diff(p')';
        %boundary conditions integration area
        ux(1,:)=-p(1,:)/rho/c/1; ux(end,:)=p(end,:)/rho/c/1;   uy(:,1)=-
p(:,1)/rho/c/1;  uy(:,end)=p(:,end)/rho/c/1;
        %boundary conditions
        ux(uxz)=signox.*p(pxz)./Zx;
        uy(uyz)=signoy.*p(pyz)./Zy;

        for ii=1:numerolisteners
            IR(contador,ii)=p(listenerx(ii),listenery(ii));



        end%ii=1:numerolistweners

    end%for tt=1:maxtt

warning off;
pp=p;
pp(round(sourcex),round(sourcey))=1;
pp=min(max(pp.^2,1d-5),1);
pp(p0)=1d-6;
for ii=1:numerolisteners
    pp(listenerx(ii),listenery(ii))=1;
end%ii=1:numerolisteners







figure(h1);

if provis(3)==1

pcolor(dh*(1:nx),dh*(1:ny),10*log10(pp'));
shading flat,axis equal;
colorbar;
title(['time=' num2str(round((tt+maxtt*(ttt-1))*1000*dt)) ' miliseconds']);
```

```
drawnow

end%if provis(3)==1

title(['time=' num2str(round((tt+maxtt*(ttt-1))*1000*dt)) ' miliseconds']);

drawnow

if provis(1)==1

   frame=getframe(h1);%Creando el video
   aviobj=addframe(aviobj,frame);%Creando el video

end%if provis(1)==1

warning on

%=========================================================================
%EXCITACIÓN Y RESPUESTA
%=========================================================================

if provis(3)==1


figure(h2);

subplot(2,1,1),plot(t,w,t(contador),w(contador),'or')

a=10*log(abs(w(contador)));
title({round(a);'dB'},'color','b');


xlabel({'Impulse';'Amp vs S'});

figure(h2);

subplot(2,1,2),plot(t,IR,t(contador),IR(contador),'or')

for ii=1:numerolisteners
b(ii)=10*log(abs(IR(contador,ii)));

end
title({round(b);'dB'},'color','b');
xlabel({'Response';'Amp vs S'});







end%if provis(3)==1

end%%for ttt

%=========================================================================
%FIN PINTANDO
%=========================================================================


final=msgbox('Simulation finished');
uiwait(final)

close(h1);close(h2);
```

```
if provis(1)==1

aviobj = close(aviobj);

clear aviobj;

end%if
%=========================================================================
%Respuesta en archivo .WAV
%=========================================================================

if provis(2)==1

    for ii=1:numerolisteners
   aser= num2str(ii);
 wavwrite(IR(:,ii),fs,aser);
    end%for
end%provis

%=========================================================================

%=========================================================================
%Obtención informe
%=========================================================================

ans=questdlg('Do you want to generate a simulation
report?','Information','Yes','No','Yes');

if strcmp(ans,'Yes')

hfig=figure;
scrsz = get(0, 'ScreenSize');
set(hfig,'tag','fig')
set(hfig,'interruptible','off')
set(hfig,'colormap',jet(256))
set(hfig,'renderer','zbuffer')
set(hfig,'menubar','none')
set(hfig,'numbertitle','off')
set(hfig,'name','Simulation report')
set(hfig,'position',scrsz)
set(hfig,'resize','off')
set(hfig,'units','pixels')
set(hfig,'DefaultUIControlFontName','default')
set(hfig,'DefaultUIControlFontSize',8)
set(hfig,'DefaultUIControlFontAngle','normal')
set(hfig,'DefaultUIControlFontWeight','light')
set(hfig,'DefaultUIControlForeGroundColor','k')
set(hfig,'DefaultUIControlInterruptible','off')
set(hfig,'DefaultUIControlBusyAction','queue')
set(hfig,'DefaultUIControlbackgroundcolor',[0.878 0.875 0.89])
set(hfig,'color',[0.878 0.875 0.89])

%=========================================================================
%Impulso dominio del tiempo
%=========================================================================

subplot(numerolisteners+1,2,1),plot(t,w)

title('Impulse')
xlabel('S')
ylabel('Amp')
grid on
```

```
%=========================================================================
%Impulso dominio de la frcuencia
%=========================================================================


subplot(numerolisteners+1,2,2)


Y = fft(w,length(w));
Pyy = Y.* conj(Y)/length(w);
f = fs*(0:length(w)/2)/length(w);
loglog(f,(Pyy(1:(length(w)/2)+1)))
title('Impulse (Freq)')
xlabel('HZ')
ylabel('Amp')

grid on
%=========================================================================
%Respuesta dominio del tiempo
%=========================================================================
j=3;
for i=1:numerolisteners

subplot(numerolisteners+1,2,j),plot(t,IR(:,i));

title('Response')


xlabel('S')
ylabel('Amp')
grid on
j=j+2;
end%for i=1...
%=========================================================================
%Respuesta dominio de la frecuencia
%=========================================================================
j=4;
for i=1:numerolisteners

subplot(numerolisteners+1,2,j)
Y1 = fft(IR(:,i),length(IR(:,i)));
Pyy = Y1.* conj(Y1)/length(IR(:,i));
f = fs*(0:length(IR(:,i))/2)/length(IR(:,i));
loglog(f,(Pyy(1:(length(IR(:,i))/2)+1)))
title('Response(Freq)')
xlabel('Hz')
ylabel('Amp')
grid on

j=j+2;
end%for i=1...

%=========================================================================
%Informe en pdf
%=========================================================================
saveas(hfig,'filename','pdf')

end%if strcmp(ans,'Si')



return%init

%=========================================================================
%=========================================================================
```

FUNCIÓN VIAC_2.

```
%==========================================================================
%VIRTUAL ROOM ACOUSTICS
%Eugenio Garcés Leante
%==========================================================================

%==========================================================================
%FUNCIÓN VIAC2
%==========================================================================

%==========================================================================
%Abre la pantalla principal del programa
%==========================================================================

function varargout = viac2(varargin)

% VIAC2 M-file for viac2.fig
%      VIAC2, by itself, creates a new VIAC2 or raises the existing
%      singleton*.
%
%      H = VIAC2 returns the handle to a new VIAC2 or the handle to
%      the existing singleton*.
%
%      VIAC2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in VIAC2.M with the given input arguments.
%
%      VIAC2('Property','Value',...) creates a new VIAC2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before viac2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to viac2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help viac2

% Last Modified by GUIDE v2.5 12-Mar-2011 11:25:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @viac2_OpeningFcn, ...
                   'gui_OutputFcn',  @viac2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before viac2 is made visible.
function viac2_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
scrsz = get(0, 'ScreenSize');
pos_act=get(gcf,'Position');
xr=scrsz(3) - pos_act(3);
xp=round(xr/16);
yr=scrsz(4) - pos_act(4);
yp=round(yr/52);

set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);
set(gcf,'name','Virtual Room Acoustics 2')


axes(handles.axes1)
background = imread('logoUPV.jpg');
axis off;
imshow(background);
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to viac2 (see VARARGIN)

% Choose default command line output for viac2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes viac2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = viac2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
materials
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
restore_materials
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
Ayuda
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
opciones
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
design2
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
design;
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
new_materials
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
simulation
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on mouse press over figure background, over a disabled or
% --- inactive control, or over an axes background.
function figure1_WindowButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```