Universidad de Valladolid

Escuela Técnica Superior
de Ingenieros de Telecomunicación

Proyecto Fin de Carrera
Ingeniero en Electrónica

Ampliación de la interfaz gráfica IISGUI para simulador de Implantación Iónica

Autor:
Jairo Alonso Ortiz
Tutor:
Jesús M. Hernández Mangas

Valladolid, 26 de Septiembre de 2008

TÍTULO:	**Ampliación de la interfaz gráfica IISGUI para simulador de Implantación Iónica**
AUTOR:	**Jairo Alonso Ortiz**
TUTOR:	**Jesús M. Hernández Mangas**
DEPARTAMENTO:	**Electricidad y Electrónica**

Miembros del tribunal

PRESIDENTE:	Martín Jaraiz Maldonado
VOCAL:	Jesús M. Hernández Mangas
SECRETARIO:	Iván Santos Tejido
FECHA DE LECTURA:	26 de Septiembre de 2008
CALIFICACIÓN:	

Palabras Clave

Implantación Iónica, Interfaz, GUI, Gráfica, Átomo, Simulador, Lenguaje de Programación, ActiveX, Ampliación, Mejora, 2D, 3D, OpenGL

Resumen del proyecto

Para el desarrollo de este proyecto era necesaria una persona con conocimientos informáticos para poder realizar los análisis y desarrollos necesarios para realizar las mejoras; y que esa misma persona tuviese los conocimientos adecuados de electrónica para poder comprender el funcionamiento del programa y sus necesidades.

El proyecto que se ha realizado contempla la mejora de ciertas partes de una aplicación ya existente y usada en la Universidad de Valladolid y la adición de nuevos componentes para la mejora de la misma.

El proyecto ha consistido en la realización de una ampliación de las funcionalidades de la aplicación añadiéndole la posibilidad de la visualización de datos en tres dimensiones, tanto en formato texto, como mediante la presentación de gráficas.

Además se han realizado numerosas mejoras de partes de la aplicación que se consideraban deficientes. Entre las mejoras más importantes están los arreglos realizados a la parte de visualización en dos dimensiones y las posibilidad de configurar la lista de simulaciones a gusto del usuario de la aplicación. Además se ha desarrollado una nueva visualización para el apartado de añadir nuevos elementos para la simulación.

Abstract

A knowledge computing person was required to develop this project in order to make the analysis and developments to improve it; and this same person must to have the accurate knowledge of electronics to understand the operation of the program and its necessities.

This project contains the improvements of certain parts of an existing application that is used in the University of Valladolid. It contains also the addition of new component to improve it.

The project has composed by the enlargement of the functionalities of the application by adding the possibility to display the 3-D data both in text format and graphic.

In addition several unsatisfactory parts of the applications were improved. Among the main improvement were the corrections made to the 2-D visualization and the possibility to customize the list of simulations to user's preferences. A new visualization in the new elements section was developed.

A todos mis amigos

Agradecimientos

No hay nada que se pueda decir que no haya dicho ya a toda la gente que me ha apoyado y me ha dado ánimos para terminar este proyecto.

Sin su apoyo casi constante muchas cosas no podrían haber salido de la manera en que lo han hecho. Muchas largas conversaciones de dudas y horas de comentarios al final dan su fruto.

Por último, quisiera agradecer a todo el mundo que de una forma u otra ha estado implicado en el desarrollo de este trabajo, mostrándome su paciencia y dándome su apoyo.

Gracias a todos y a los demás también.

Prologo

No siempre el llegar a la meta el primero es lo más importante. Muchas veces con llegar es suficiente, y la gente que te conoce lo valora y lo agradece, porque ha depositado en ti muchas esperanzas y sueños.

Pero esto no es el final ni la meta de nada, apenas acabamos de tomar la salida. Ahora nos queda mucho tiempo para desarrollarnos y demostrar que nuestro camino ha sido el correcto y no hemos perdido el tiempo.

Unos iremos en un sentido y otros en el contrario, y puede que no volvamos a ver a la persona que teníamos a nuestro lado en la línea de salida, pero seguro que nos acordamos de ella muchas veces; e incluso puede que alguna vez futura podamos volver a coincidir y unamos nuestros esfuerzos de nuevo por el bien de ambos.

Ahora que hemos hecho el calentamiento estamos preparados para la carrera.

El autor

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Origen

Este proyecto surge de una necesidad mutua, por una parte el profesor responsable con ganas y tiempo de enseñar a un alumno y ver como evoluciona uno de sus programas; del otro el alumno necesitado de poder realizar un trabajo que demuestre que está preparado para dar el salto y convertirse en ingeniero.

La razón de la elección de este proyecto por parte del alumno y la idea de proponerlo por parte del profesor surge debido a que el alumno proviene de Ingeniería Informática, por lo que le resultaría más sencillo unir los conocimientos de Informática y de Electrónica y poder plasmarlos en un proyecto que consista en mejorar un programa informático relacionado con la electrónica.

1.2. Definición del proyecto

Este proyecto se define como una ampliación y mejora de las funcionalidades que ofrece un programa de simulación ya existente y funcional, al cual se le desarrolló una interfaz gráfica como otro proyecto.

El mayor peso del proyecto recae sobre la realización de un módulo que se encargue de representar los resultados obtenidos en la simulación mediante gráficas en tres dimensiones, esta parte conforma el apartado de ampliación.

La mejora de partes ya existentes contempla muchos apartados y matices a renovar, y, en conjunto, pueden llegar incluso a superar la carga de trabajo y esfuerzo de la parte anterior.

1.3. Previsión de Contenidos

La idea inicial del proyecto es la de mejorar distintos aspectos de la interfaz gráfica del programa *"Ion Implant Simulator GUI"*.

Figura 1.1: Logotipo del programa Ion Implant Simulator GUI

La parte más importante del proyecto se centra sobre darle al programa la posibilidad de representar gráficas en tres dimensiones de los resultados obtenidos en las simulaciones, así como la visualización de los resultados en modo texto. Otras partes menores incluyen la corrección de errores, sobre todo del apartado de visualización en dos dimensiones, o la mejora de ciertas vistas o menús.

Los contenidos a realizar serán:

- **Ampliación del programa:** Esta parte corresponde a la más importante del proyecto, y consiste en desarrollar y programar, con un lenguaje estructurado orientado a objetos, un objeto externo al programa principal que se encargue de realizar la visualización de datos en tres dimensiones.
- **Mejora del programa:** Esta parte consiste en corregir los errores y deficiencias encontradas en la aplicación durante el tiempo que lleva usándose. También será un apartado de programación.
- **Actualización de la ayuda:** Tendremos que actualizar la ayuda del programa con los cambios introducidos. Esta parte no requiere programación, se utilizará un programa especifico de creación de ayudas.
- **Mejora de la instalación:** Una vez que tengamos todos los apartados anteriores listos generaremos una distribución de la aplicación que simplifique el proceso de instalación para los usuarios finales.

1.4. Objetivos Iniciales

Los objetivos iniciales o necesidades planteadas sobre el aplicativo no parecían en un primer acercamiento complicados de conseguir. El estudio más detallado de cada objetivo de manera individual descubrió que algunas partes se resolverían más fácil de lo planteado, otras no serían tan sencillas como parecían y que algunas no se podrían ni siquiera llegar a realizar.

La necesidad de proveer a la aplicación de un módulo que se encargase de la representación de gráficas en tres dimensiones es lo más importante. El simulador genera los datos que servirán de entrada para esta visualización, pero los datos y la funcionalidad se desaprovecha ya que no esta disponible esta posibilidad de presentación en la aplicación. Este módulo se desarrollará como un objeto externo a la aplicación de manera que se pueda usar en cualquier otro programa si se desea.

Otra parte importante es añadirle al programa la posibilidad de que admitir parámetros nuevos para mejorar las posibilidades del simulador que trabaja por debajo de la interfaz. Esto surge como necesidad para las posibles evoluciones del simulador y que no esté condicionado a la evolución de una interfaz creada por separado.

El resto de tareas a realizar son de mejora y corrección de errores existentes y localizados de la aplicación.

Capítulo 2

Aplicación Origen

2.1. Estado de la Aplicación

Hemos de tener presente en todo momento que la aplicación que se va a mejorar esta funcionando correctamente y está siendo usada por algunas personas para la realización de pruebas de simulación. Nosotros solo nos vamos a limitar a mejorarla en ciertos aspectos y a dotarla de nuevas funcionabilidades.

Algunos de los apartados con los que cuenta la aplicación son deficientes o presentan errores que nos vamos a encargar de mejorar y solucionar. Además de estos fallos también se van a cambiar algunos aspectos por petición expresa. Son apartados que funcionan pero que no se les saca todo el partido posible debido a su deficiente interfaz o las dificultades de acceso.

En los siguientes apartados se van a comentar los apartados que se van a modificar y las nuevas funcionalidades que se van a aportar a la aplicación durante este proyecto, indicando cual es el estado actual de los distintos apartados e indicando cuales son los que se van a modificar o ampliar.

2.2. Nuevas Funcionalidades

El programa ya tiene mucha funcionalidad, pero hay alguna parte que le falta para ser completo. Entre estos apartados se encuentran la posibilidad de mostrar gráficas en tres dimensiones y evitar la dependencia del simulador con la interfaz, en lo que a desarrollos o evoluciones de éste se refiere.

La ampliación del programa es la parte que más tiempo va a llevar y la que va a requerir de mayor esfuerzo, si contamos cada una de las mejoras que vamos a realizar por separado y no como conjunto.

2.2.1. Visualización en tres dimensiones

El programa presenta la posibilidad de mostrar las gráficas de ciertos ficheros en dos dimensiones, pero hay otros que son en tres dimensiones y no pueden ser mostrados por la falta de esta funcionalidad.

La visualización de ficheros en tres dimensiones será resuelta mediante la construcción de un objeto externo destinado a esta labor. El objeto será muy similar al existente para la representación de gráficas en dos dimensiones para que el uso de distintos tipos de gráficas no dificulte el uso del programa.

2.2.2. Nuevos Parámetros

EL programa *"Ion Implant Simulator GUI"* solo presenta una interfaz gráfica para la introducción y presentación de datos de un simulador que trabaja por debajo. No podemos condicionar las mejoras del simulador con las evoluciones que siga la interfaz. Por este motivo se desea que se cree la posibilidad de recoger nuevos parámetros de entrada para el simulador para las posibles evoluciones es éste.

Para realizar esta parte se añadirá una nueva hoja de propiedades a la simulación donde se podrán generar estos nuevos parámetros de entrada para el simulador. Se van a permitir definir cuatro tipos de parámetros como se explica en el apartado 3.2

2.3. Correcciones

Con el inicio del proyecto se indicaron unas partes ya identificadas de la aplicación que había que mejorar y corregir. Posiblemente durante el desarrollo se encuentren otros errores o deficiencias que también necesiten ser resueltas. Los puntos inicialmente marcados son:

1. Idiomas de la aplicación.

2. Corrección de ciertos apartados de visualización de las gráficas en dos dimensiones.

3. Posibilidad de tener ejecuciones en **"batch"**en lugar de con conexión directa.
4. Rehacer el dialogo de **"Control de Versiones"**.
5. Añadirle más campos a la lista de simulaciones.
6. Mejorar la tabla de elementos de la aplicación.
7. Investigar la posibilidad de cambiar los atajos de teclado.
8. Poner los ficheros de configuración en modo texto.
9. Quitar las rutas absolutas de la aplicación.
10. Rehacer la ayuda en castellano e inglés.
11. Generar un nuevo instalable

2.3.1. Visualización en dos dimensiones

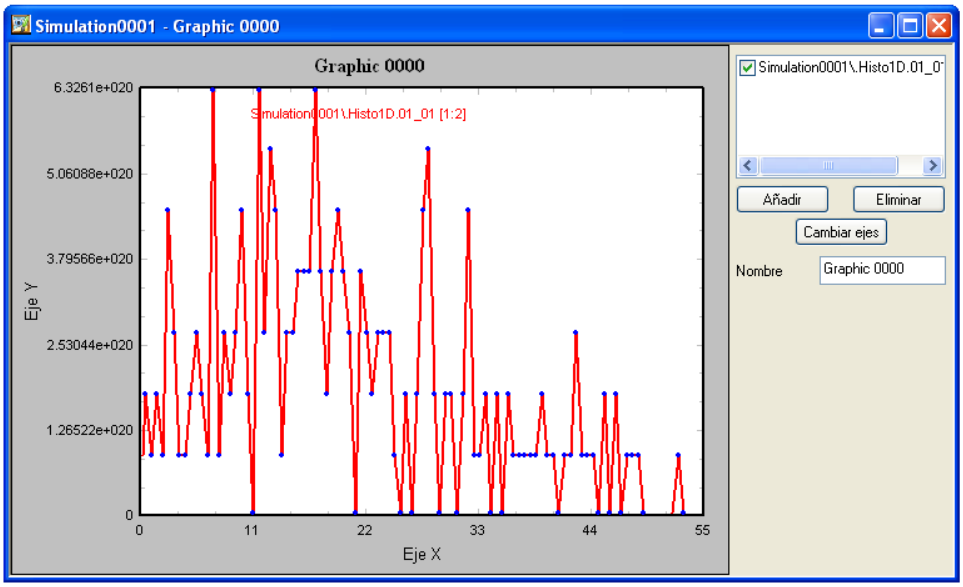

Figura 2.1: Objeto que dibuja las gráficas en dos dimensiones

El objeto **NTGraph.ocx** se encarga de la visualización de las gráficas en dos dimensiones. Este objeto presenta muchas funcionalidades y realiza correctamente su labor

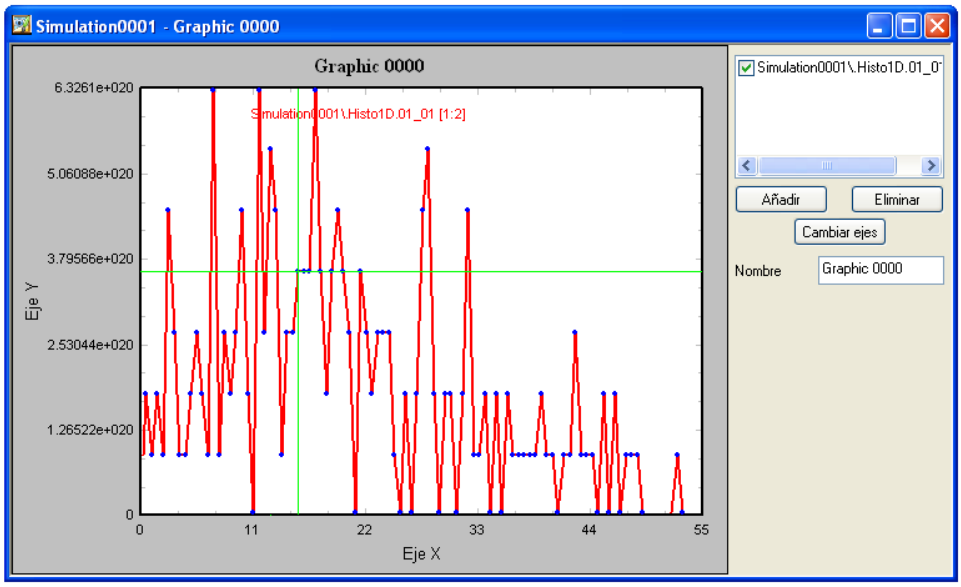

Figura 2.2: Fallo del objeto al mostrar cursores. Ratón en X=22; Y=3.5e20

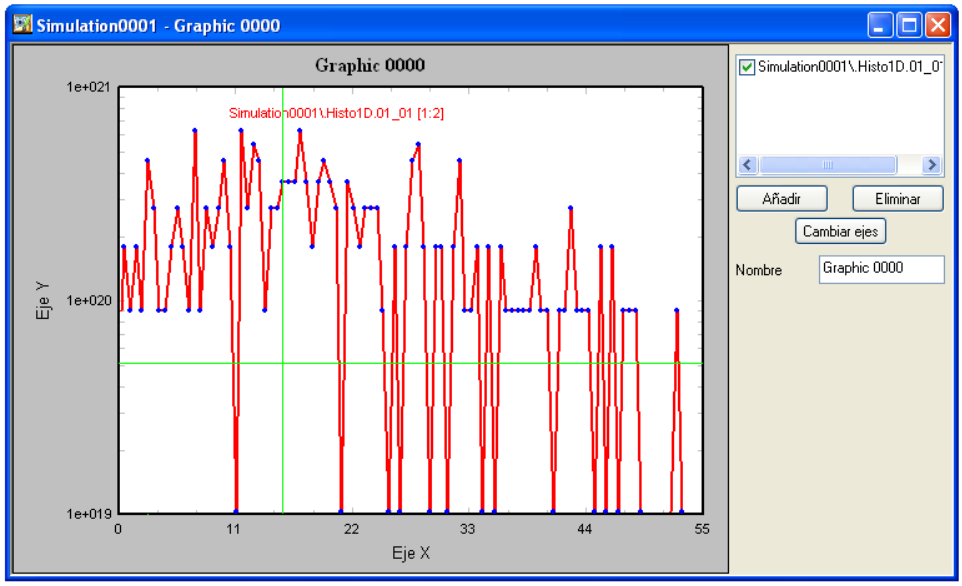

Figura 2.3: Fallo del objeto al mostrar cursores en modo logarítmico

siempre que se trabaje con los datos en lineal. En cuanto se usa el objeto con los datos en logarítmico se puede observar que presenta numerosos fallos.

El problema que presenta el objeto es que no distingue cuando están los ejes en lineal y cuando en logarítmico, realizando los cálculos de posición siempre como si los ejes estuviesen en lineal, esto hace que los marcadores, leyendas, cursores, etc., no se coloquen en los lugares correctos cuando se dibujan los datos en escala logarítmica.

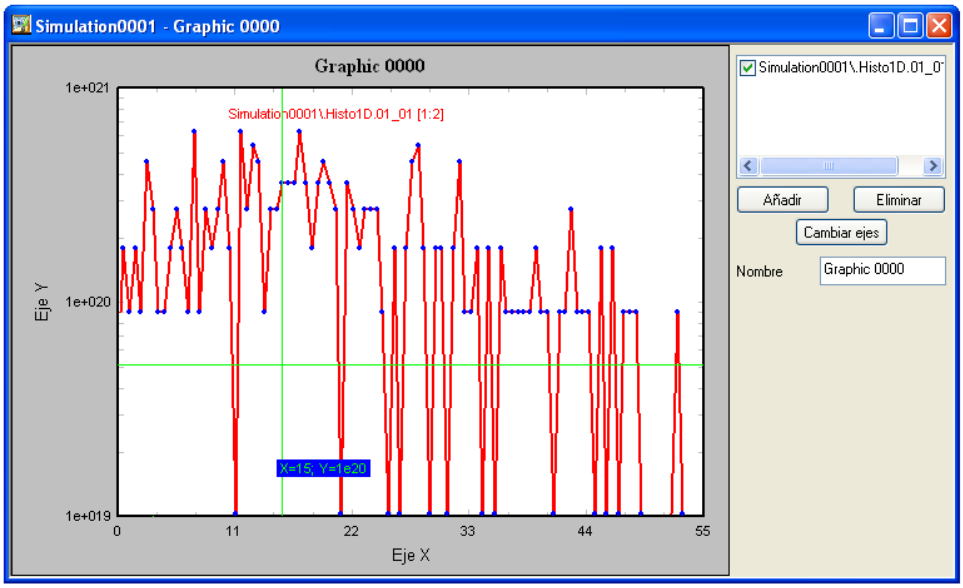

Figura 2.4: Fallo del objeto al mostrar las anotaciones en modo logarítmico

Además de las correcciones anteriores también se desea que el objeto dibuje la leyenda de las gráficas que están siendo dibujadas. Actualmente pinta el nombre, pero no se puede elegir su posición. Se desea que exista una leyenda única que muestre los datos de todas las gráficas que se estén pintado, el objeto permite pintar más de una gráfica en cada momento, y que ésta tenga ciertas características como poder ser mostrada u ocultada, así como aparecer con fondo de color y recuadrada.

El objeto también provee de ciertas funciones asociadas a un menú de contexto. Estas funciones son las de guardar la gráfica como bmp en un fichero, copiarla al portapapeles o imprimirla. Se quiere que estas opciones también estén presentes en las hojas de propiedades de la aplicación.

El objeto que se usa para visualizar las gráficas en dos dimensiones se basa en un objeto

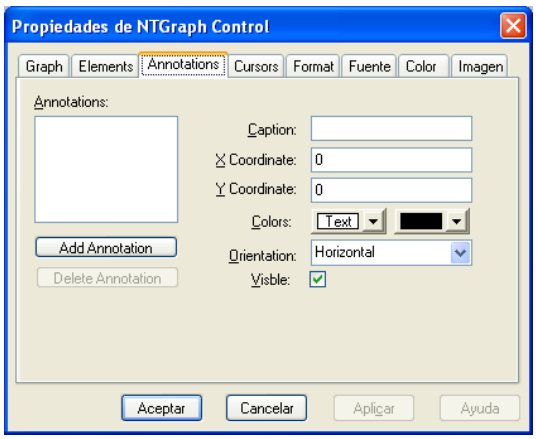

Figura 2.5: Página de propiedades antes de añadir las opciones de la leyenda

existente sacado de internet [20]. El objeto está íntegramente desarrollado en inglés y las páginas de propiedades se muestran siempre en dicho idioma.

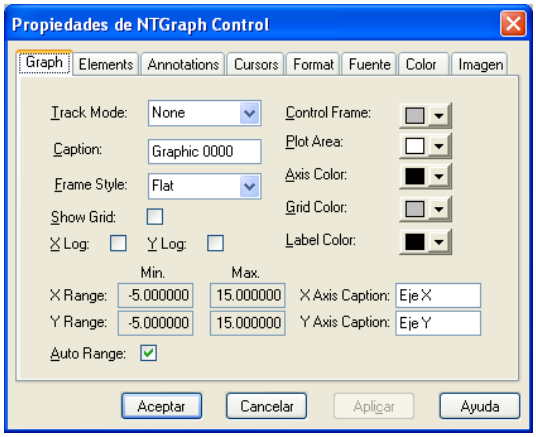

Figura 2.6: El objeto muestra todas la hojas de propiedades únicamente en inglés

2.3.2. Ejecuciones Batch

La aplicación permite realizar las simulaciones de las aplicaciones en local o en remoto a gusto del usuario. Se pretende que además de estas opciones se incorpore la posibilidad de que las simulaciones se puedan realizar en modo batch, esto es, que nos podamos desconectar de la máquina sobre la que se ha lanzado la simulación para recoger posteriormente los resultados cuando nos volvamos a conectar a la máquina.

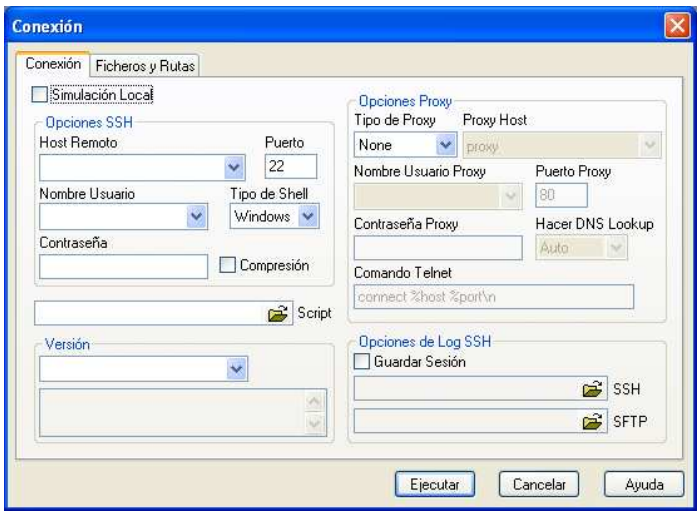

Figura 2.7: Opciones de Ejecución de la simulación

2.3.3. Control de versiones

El Control de versiones es una opción general del simulador. Aparece como una pestaña dentro de *Herramientas⇒Opciones...* y se desea que tenga un acceso más directo, desde el propio menú herramientas y desde la barra de tareas.

Figura 2.8: Ventana del Control de Versiones

Además de lo anterior, también se quiere que se seleccione el script que se va a usar en la simulación desde este apartado y no desde el apartado de simulaciones como se venía haciendo hasta ahora.

2.3.4. Lista de simulaciones

La lista de simulaciones presenta las simulaciones que están abiertas en cada momento. Además, muestra una serie de información de cada simulación. Cada columna puede ocultarse o mostrarse al gusto de usuario, pero no se pueden poner otras distintas.

Figura 2.9: Versión original de la Lista de Simulaciones

El objeto de la lista de simulaciones tiene muchas propiedades que se están desaprovechando por la limitación de que los campos mostrados siempre son los mismos, por ello se quiere que los campos que se muestren en la lista de simulaciones sean seleccionables de alguna forma dentro de la aplicación entre todo el conjunto de propiedades de las simulaciones.

2.3.5. Tabla de elementos

La tabla de elementos que presenta la aplicación está realizada con dos imágenes bmp, una de ellas es el positivo y la otra el negativo para marcar el elemento que se está seleccionando.

Este modo de proceder de la tabla de elementos impide que se presente en distintos idiomas, de forma que la tabla siempre aparece en el mismo idioma, indistintamente del idioma de la aplicación. El resto de funcionalidades que presenta la tabla de elementos es el que se desea.

2.3.6. Atajos de teclado

Todos los menús de la aplicación presentan en uno u otro apartado la posibilidad de utilizar una combinación de teclas para ejecutar dicha orden. Se pretende que estos atajos se puede cambiar a gusto del usuario, pudiendo introducir nuevos atajos para las funciones que más utiliza o redefiniendo los ya existentes por otros que sean más de su agrado.

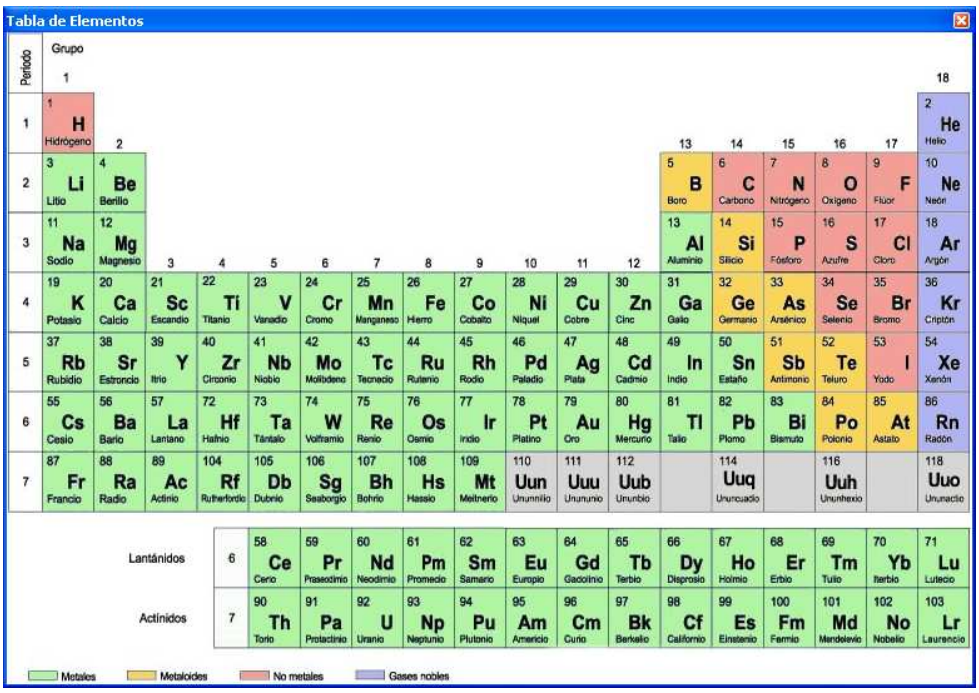

Figura 2.10: Versión inicial de la Tabla de Elementos

2.3.7. Ficheros de configuración en modo texto

La aplicación presenta gran cantidad de propiedades y características configurables que son guardadas en el registro de Windows y recuperadas desde su ubicación cada vez que se inicia la aplicación.

Para un mejor control de las distintas propiedades se quiere que todos los datos que necesite la aplicación para funcionar, como valores por defecto, variables globales, ...; se guarden en ficheros en formato de texto en alguna ruta de la aplicación para que puedan ser leídas sin problemas y modificadas, en su caso, sin necesitar la interfaz gráfica para hacerlo.

2.3.8. Rutas de la aplicación

La aplicación utiliza numerosos ficheros externos para contener distintos tipos de datos. La mayoría de las rutas de los ficheros son absolutas, lo que hace que el programa no funcione si no se utilizan las rutas correctas.

2.3.9. Creación de la Ayuda

La ayuda actual que presenta la aplicación es muy deficiente, estando solo realizada una versión que parece indistintamente cual sea el idioma elegido en la aplicación además de contener partes de la misma en castellano y parte en inglés.

Se pretende que la aplicación presente la ayuda en el idioma en el que se está ejecutando, mostrando toda la ayuda en el idioma de la aplicación, de forma que pueda ser utilizada y comprendida por todas las personas que la utilizan.

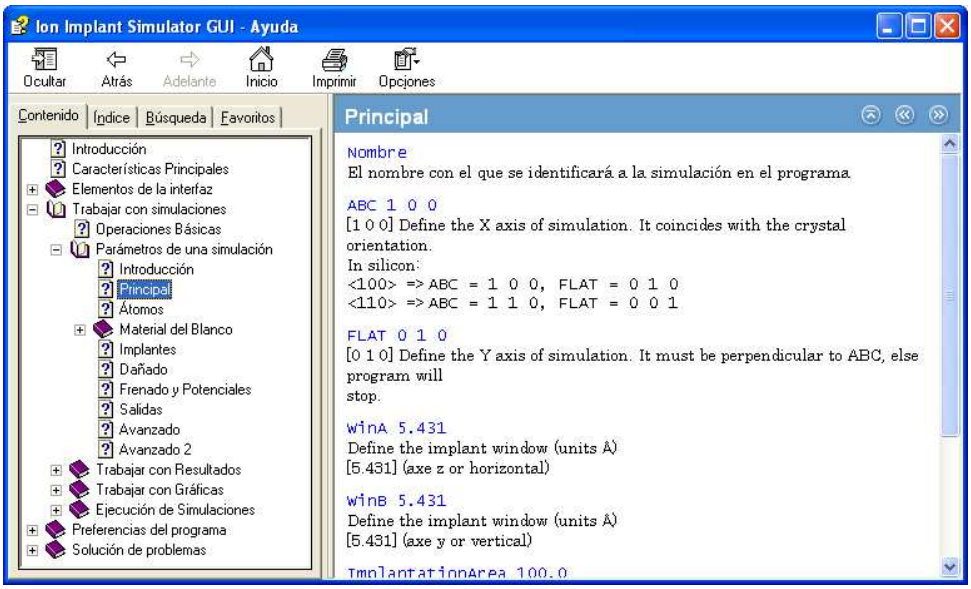

Figura 2.11: Imagen de la ayuda existente en la aplicación

2.3.10. Software Distribuible

Debido a la realización de modificaciones del programa es totalmente necesario la generación de un nuevo software distribuible que contenga todas las nuevas funcionalidades de la nueva versión.

Se sugiere que el proceso de instalación debe ser lo más sencillo posible para no causar dudas en los usuario.

Capítulo 3

Desarrollo del Proyecto: Ampliaciones

Dentro del capítulo de Ampliaciones trataremos el modo de creación de las nuevas funcionalidades con las que se ha proveído a la aplicación, haciendo referencias a distintas partes de código para que se pueda ver como se han resuelto algunas partes comprometidas del desarrollo.

3.1. Visualización en Tres Dimensiones

La parte más importante dentro de las ampliaciones que se van a realizar sobre la aplicación es la realización de un objeto externo que se encargue de dibujar gráficas en tres dimensiones.

3.1.1. Introducción

La necesidad de crear un objeto para mostrar gráficas en tres dimensiones surge debido a que la aplicación ya realiza la presentación de resultados en dos dimensiones, pero ciertos datos no pueden ser presentados con esta vista. La base del módulo pertenece a una librería publica de internet [21]. El módulo utilizado de base esta desarrollado utilizando el lenguaje **C++** bajo *Visual Studio 6.0.* Esta base se estudió, migró a *Visual Studio 2003 .NET* y modificó como se explican en los siguientes apartados.

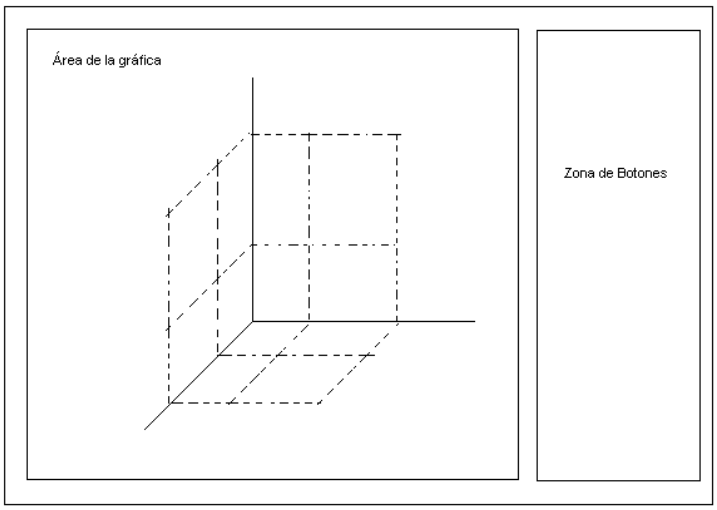

Figura 3.1: Boceto inicial del apartado Gráficas 3D

3.1.2. Análisis de la Solución

Se quiere realizar un objeto similar al existente para gráficas en dos dimensiones pero para gráficas en tres dimensiones. La solución debe aportar muchas posibilidades de las existentes en el objeto ya existente para las gráficas en dos dimensiones.

El módulo tiene que ser capaz de leer un fichero proveniente de la salida del simulador compuesto por un array tridimensional de datos. Este array tiene un tamaño de 50x50x50, lo que en realidad significa que son 50 capas de datos de tamaño 50x50. Las posibilidades de representación tienen que incluir la representación de las capas de forma individual o la suma de todas ellas (Acumulado). También se debe poder realizar la representación de datos en las tres dimensiones del array, es decir, tomar como referencia de datos cualquiera de los tres ejes.

Además se tienen que poder representar los datos en lineal y en logarítmico, debido a que al ser datos provenientes de una implantación iónica estaremos hablando de que todos los valores serán siempre positivos y, la mayoría de las veces, de ordenes de magnitud muy elevados.

Como otras características del módulo se tienen que poder realizar cambios del ángulo de visión de la gráfica así como cambios de colores o de valores de los rótulos de los ejes, y de las etiquetas de estos últimos y el propio título de la gráfica.

3.1.3. Realización del Objeto

La solución de realizar el objeto como un módulo se tomó para que el objeto una vez desarrollado pueda ser utilizado conjuntamente con otros programas que se están desarrollando en el departamento de electrónica. Además, la solución base ya era una librería de enlace dinámico (.dll). Se estudió la solución de esta librería y la viabilidad de realizar otro objeto, incluso usando otro lenguaje, pero se desechó la posibilidad.

De la librería inicial tomada como referencia a la versión final hay muchos cambios, de hecho los programas de ejemplo existentes para la librería ya no funcionan debido a la cantidad de cambios introducidos. Actualmente el módulo se llama *NTGraph3D.dll* y funciona como un objeto ActiveX.

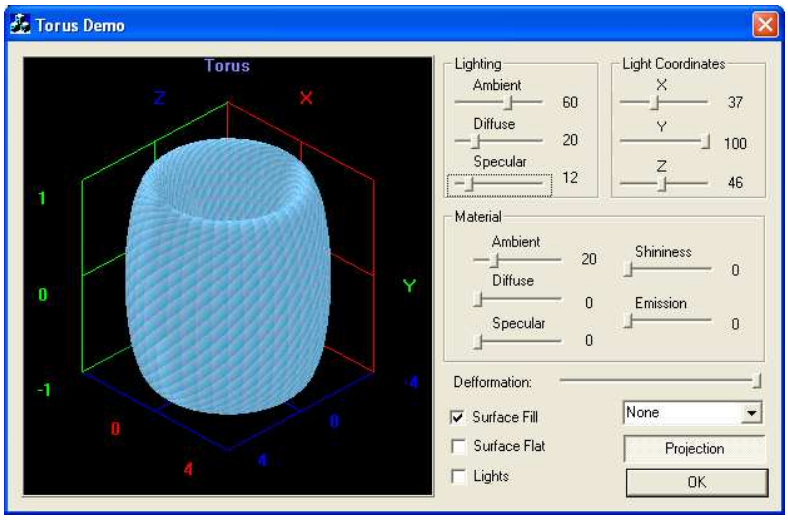

Figura 3.2: Programa de test de la librería inicial

Datos de Entrada

El módulo de representación funciona por coordenadas (x, y, z), donde x e y se corresponden con la posición de pantalla en la que dibujar y z el valor que se desea pintar, estas coordenadas de dibujado no tienen porque corresponderse con el plano que se ha seleccionado para pintar, sino que son sus correspondientes en pantalla. El módulo esta pensado para pintar funciones sencillas por lo que hay que modificar casi todos los procedimientos para conseguir la funcionalidad de dibujado de datos y/o de funciones.

El pintado del fichero con un formato correcto supone el desarrollo de un procedimiento que lea dicho formato y luego sea capaz de pasarle los datos a la función de dibujado. Aprovechando la creación de un procedimiento para la lectura del fichero de datos 3D se reutiliza este mismo procedimiento para leer el fichero y mostrar los datos en forma de texto, acción que tampoco estaba implementado.

Se crean tres nuevos procedimientos llamados, *Inicia* , *Escribe* y *Extrae_Loncha* [B.1.1] que se encargan de leer los datos por capas o de forma acumulativa.

Representación

La representación de datos proveída por el objeto original solo esta soportada para datos en formato lineal. Para poder adaptar la solución a datos en logarítmicos hay que adentrarse completamente dentro del código del módulo y realizar numerosos cambios.

El cambio más importante involucra a la función de representación de los datos, que debe saber cuando se requiere una presentación en formato lineal y cuando se requiere los datos en forma logarítmica. Hay que añadir nuevos procedimientos y variables públicas para poder indicarle al objeto desde el aplicativo cual es el modo de representación. El valor de los datos de cada punto se tiene que traducir a las coordenadas reales de la pantalla para poder dibujar en un lugar concreto. Esta transformación de puntos solo está contemplada para datos lineales. Con el uso de las nuevas variables introducidas para saber si se quieren los datos lineales o logarítmicos se hace la transformación del dato siguiendo una fórmula u otra [B.1.1]. La creación de estos procedimientos y variables es una tarea sencilla, el problema surge en la función de representación.

Además de cambiar la función de representación también es necesario modificar el dibujado de los ejes para que acepte los valores logarítmicos como rótulos de los ejes y las funciones de rango de dibujado y autorango. Para esto hay que rehacer todo el dibujado de los ejes, puesto que ya no eran lineales, y los valores de los ejes. No tiene sentido poner valores numéricos al azar en los ejes cuando se está dibujando la gráfica en logarítmico, por lo que se opta por marcar los ejes con potencias de 10, de forma que de una marca a otra se da un salto logarítmico de una década. Esto obliga a cambiar la forma de determinar el máximo y el mínimo de los valores a representar para que siempre fuesen una potencia de 10 y no los valores reales provenientes de los datos, para de esta manera determinar los valores de los rótulos de los ejes.

Aspectos visuales

El aspecto visual es el más importante del objeto. Debido que lo que se quiere es una muestra de datos a este apartado es sobre el que más se va a trabajar para que tenga la mayor cantidad de parámetros definibles por parte del usuario y así pueda definir las vistas a su gusto y para su comodidad.

La gráfica se pinta como una superficie sólida con líneas marcadas de color para que haga un efecto de tres dimensiones. Conseguir esta solución es fruto de muchas horas de prueba con objetos de OpenGL [18]. La superficie se pinta mediante polígonos de cuatro lados, también se realizó una prueba con un procedimiento que los dibujaba con triángulos, pero se desechó porque la visualización era muy similar, pero recargaba el dibujo con demasiadas líneas.

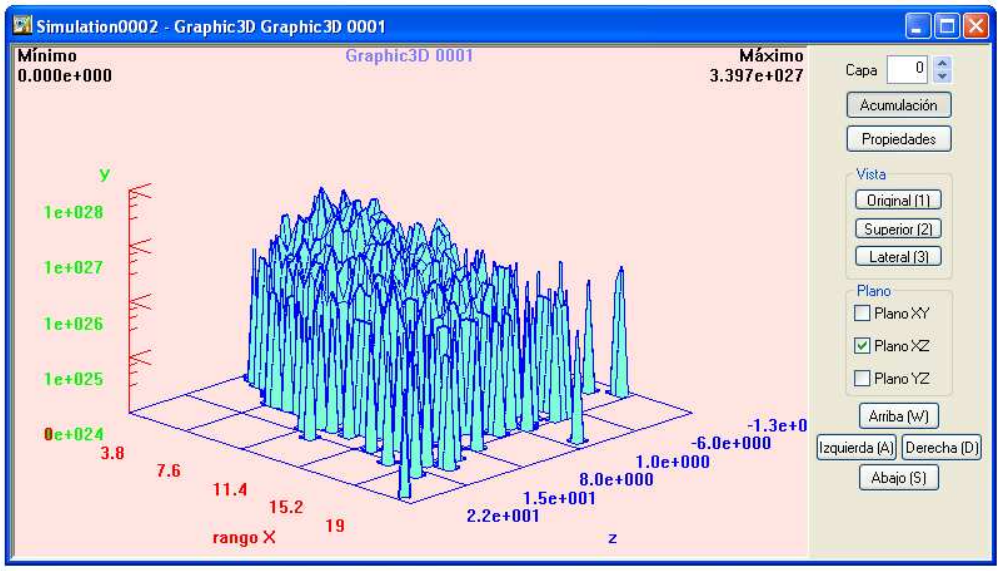

Figura 3.3: Objeto de visualización en 3D

Además de esta opción por defecto también se puede pintar la gráfica como solo líneas o solo los puntos donde están los valores, o una combinada de estas dos. Para la presentación de los ejes se utiliza el mínimo número de líneas posibles, reduciendo los ejes a la mínima expresión y componiéndolos únicamente por la base y un lateral sobre el que se rotulan las posiciones relativas de los datos dibujados. En la parte superior se muestra en la parte izquierda el valor mínimo de los datos y en la derecha el máximo, protegido entre estos valores se encuentra el título de la gráfica. Los valores máximo y

mínimo se dibujan con el conjunto de líneas y rótulos de los ejes y van cambiando según los datos mostrados para indicar siempre el máximo y el mínimo de los datos mostrados, no de todos los datos de la simulación.

Prácticamente todos los apartados de la gráfica se pueden configurar para mejorar la visión de los resultados. Todos los aspectos relacionados con los colores se pueden modificar poniéndolos de la manera que más le atraiga al usuario, así como el formato de representación de los ejes. El objeto realizado posee funciones publicadas para cambiar y modificar todos estos aspectos.

Propiedades

Para las opciones más importantes de la aplicación o van a ser de más uso, se opta por la posibilidad de situarlas en la propia ventana principal de visualización del objeto. El resto de opciones del módulo se incluyen en páginas de propiedades.

Figura 3.4: Página de propiedades *"Gráfica"*del componente de visualización en 3D

Las opciones que van a quedar en la ventana principal son las de cambio de capa y de plano de dibujado, así como una casilla que indica si se utiliza la acumulación (en este caso no hay capa efectiva) o por el contrario se dibujan los datos de una capa concreta, indicada por el valor del campo capa. Es esta página inicial también están unos botones con unas vistas predefinidas que permiten poner la imagen en cualquiera de ellas independientemente del ángulo de visualización en el que se esté mostrando la gráfica actual. También se incluyen unos botones de desplazamiento de la gráfica, que sirven para moverla de grado en grado en el sentido marcado por el botón. Estás opciones y

campos se definen directamente sobre la aplicación final, puesto que están situadas en la página principal del objeto.

Debido al impedimento de crear nuevos objetos del tipo ventana y hoja de propiedades en el objeto por problemas de incompatibilidades entre distintas clases se toma la decisión de integrarlas en la aplicación final.

El objeto tiene dos opciones de páginas de propiedades distintas una vez que se integra en la aplicación final. Tiene un apartado de páginas de propiedades propias, que son las que se generan de forma automática en todo componente ActiveX. En este apartado se pueden ver las fuentes y los colores de los ejes, que es lo que se puede modificar. Para acceder a estas propiedades hay que hacer doble click sobre el objeto.

Dentro de la propia aplicación final (IISGUI) se definen otra serie de páginas de propiedades que hacen llamadas a todas las funciones públicas de las que dispone el objeto con el fin de poder modificar el máximo número de apartados posibles. Existen tres pestañas diferentes con las opciones catalogadas según pertenezcan al entorno de la gráfica, el objeto dibujado o el formato en el que aparecen los ejes.

Figura 3.5: Página de propiedades *"Elemento"*del componente de visualización en 3D

Desde la primera de las páginas de propiedades denominada *Gráfica* se puede definir el título de la gráfica y los de los tres ejes de las gráficas, así como el modo de dibujado que se quiere para los datos de la gráfica y el color de fondo del objeto. También se puede indicar cuales serán el máximo y el mínimo de los datos a mostrar o seleccionar la opción de autorango que determinará por si sola los mejores rangos posibles para la visualización de los datos. La última posibilidad de elección es marcar si queremos que nos saque los

datos en lineal o logarítmico, esto también tiene repercusión sobre los rangos a visualizar. En el caso de que la visualización sea en modo logarítmico los rótulos de los ejes estarán en potencias de 10, lo que significa que si se indica un rango que no es potencia de 10, éste se modificará a la potencia más adecuada.

La segunda pestaña rotulada como *Elemento* permite elegir las opciones del elemento dibujado como son los colores, haciendo distinción entre punto, linea y superficie, y el tamaño con el que se dibujan las líneas y los puntos. Las opciones de punto y linea no se tendrán en cuenta si no se muestran dichas características, la gráfica por defecto se pinta como una superficie opaca con líneas uniendo los puntos representados.

La última pestaña llamada *Formato* permite elegir cual será el formato de los rótulos que marcan las divisiones en los ejes de datos. Se pueden elegir de forma individual para cada eje y tiene multiples opciones de visualización. Estos formatos son los mismos que los que se pueden ver en las gráficas en dos dimensiones.

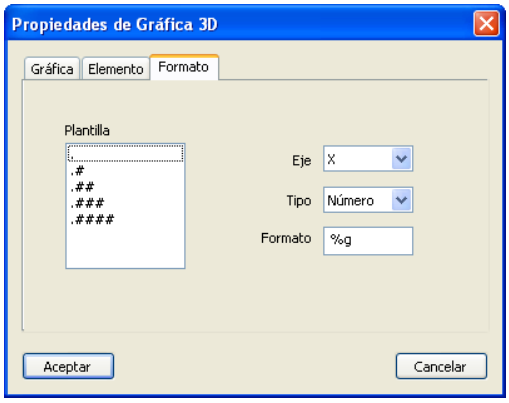

Figura 3.6: Página de propiedades *"Formato"*del componente de visualización en 3D

Todos los parámetros que se definen en estas pantallas son guardados en un fichero de configuración en modo texto. Este fichero puede editarse libremente, siempre que se tenga en cuenta que hay que respetar los valores que puede tener cada variable, el orden y la palabra que precede a cada valor. El archivo es independiente para cada gráfica y se identifica por llevar el mismo nombre que la gráfica a la que configura. Todos estos ficheros de configuración se encuentran en el directorio de la simulación propietaria de las gráficas.

Dentro de la propia aplicación se requiere definir que ficheros son los que van a generar las gráficas 3D y cual va a ser la clasificación del nuevo grupo de gráficas. Para

la clasificación se crea un nuevo item, una nueva rama, en el árbol de simulaciones [6], llamado **Gráficas3D** o **Graphic3D** dependiendo del idioma, que contendrá todas las gráficas 3D con nombre *Graphic3D* seguido del nombre propio de la gráfica, por defecto el nombre es Graphic3D y un número de cuatro dígitos. El nombre del fichero es lo que determina si un pertenece a las gráficas 3D y se puede dibujar una gráfica con ella o pertenece a otro tipo, a las gráficas en dos dimensiones o simplemente texto plano. Si el nombre del fichero termina en *histo3D.bin* entonces pertenece a las gráficas 3D.

Errores encontrados

Por el desarrollo del programa, y dado que se trata de una librería ya existente y no desarrollada por nosotros ni conocemos su idea de desarrollo, no se le pueden incluir la muestra de ventanas ni objetos similares. Al intentar añadirle estas ventanas para proveer al objeto de páginas de propiedades u otras funcionalidades presentaba conflictos entre algunas de las librerías necesarias para el funcionamiento del componente, por lo que se optó por hacer las páginas de propiedades en el programa principal. Como se trata de un objeto desarrollado con C++ bajo Visual Studio, eso se traduce en que el módulo no acepta objetos de las clases *CDialog*, *CPropertyPage* y similares.

El objeto para que responde a la presión de ciertas teclas debido el desarrollo ciertos eventos. Las teclas pulsadas pueden hacer rotar la gráfica en un sentido concreto dependiendo de la tecla pulsada o colocarla directamente en una posición predefinida. Conseguir que el objeto respondiese ante la pulsación de ciertas teclas de la manera correcta fue una difícil tarea que se resolvió de la mejor manera posible.

Como se ha indicado, el objeto dibuja las superficies con polígonos de cuatro lados. Se puede creer que esto hace perder calidad a la visualización pero no es así. También se ha hecho una prueba, a petición del tutor, dibujando la gráfica con triángulos; pero la sensación visual no cambiaba y si que se notaba una ralentización del objeto al tener una función de pintado más larga.

Se encontró un problema al visualizar los ficheros ya que aparecían triángulos de lado a lado de la zona de representación de la gráfica. Este problema se localizó en la función de dibujado ya que realizaba una unión del último punto de una serie con el primero de la siguiente. Este problema se solución localizando los finales de las series y uniéndolos consigo mismos de forma que el polígono de cuatro lados que se dibujaba quedase cerrado, aunque realmente solo se pintase una linea. Posteriormente surgió el mismo problema

pero en otra dirección. En esta ocasión el problema se debía a que se estaba accediendo a posiciones de array no existentes, nos estábamos saliendo de las dimensiones del array que contenía los datos que tenían que ser dibujados.

3.1.4. Pruebas

Con las pruebas realizadas al objeto y las funcionalidades realizadas en la aplicación principal se ha pretende probar al máximo el objeto y los accesos que se realizan al mismo desde la aplicación principal.

Como el objeto esta desarrollado sobre una aplicación de pruebas, desde ella se prueban todas las funciones de dibujado, y se comprueba como realizan su cometido y que no se producen errores, antes de iniciar su integración en el programa principal. La validación de la mayor parte de la aplicación se realiza sobre el programa de pruebas, pero aún así se requiere una aprobación final del objeto trabajando en la aplicación final.

Durante el desarrollo inicial y para que se pase la validación es necesario cambiar el modo de visualización de la gráfica durante la rotación. En el objeto original solo se mostraban los ejes y ahora la rotación se hace visualizando toda la gráfica, al igual que cuando no ser rota. Este cambió se origina durante una de las pruebas realizadas.

Una vez dentro de la aplicación principal se prueban que todos los cambios que se realizan desde las hojas de propiedades se trasladaban al objeto y viceversa. Todas las opciones son grabadas en un fichero para cada gráfica existente, este fichero puede editarse y sus características se tienen en cuenta cuando se visualiza dicha gráfica.

Se comprueba que un fichero correspondiente a datos de gráficas 3D genera este objeto y no el otro diseñado para las gráficas en 2D o cualquier otra vista. Como, aprovechando el objeto, también se pueden mostrar ficheros en modo texto, se comprueba que los fichero relacionados con estas gráficas también se visualizasen de forma correcta en modo texto, la distinción de este modo de muestreo del fichero de texto es la misma que para la utilización del objeto en modo gráfico.

Durante las pruebas realizadas sobre el árbol de simulaciones se detecta que no se sitúan las gráficas en su lugar correcto y que la numeración seguida no es la más adecuada. Con una corrección del refresco del árbol y añadiendo el nombre característico para las gráficas (Graficas3D o Graphic3D) se corrige el problema. Este fue el problema más grave

encontrado una vez añadido el objeto al programa principal, así como el más difícil de resolver.

3.2. Nuevos Parámetros

3.2.1. Introducción

Se requiere que la interfaz gráfica diseñada no bloquee el desarrollo del simulador que trabaja por debajo de ésta. Por eso se hace necesario la incorporación de este apartado dentro de la aplicación principal.

Este apartado fue uno de los últimos en iniciar el estudio y el desarrollo pues era uno de los más claros y más definidos, por lo que se preveía de mejor manera el tiempo que se iba a necesitar para la realización de esta parte.

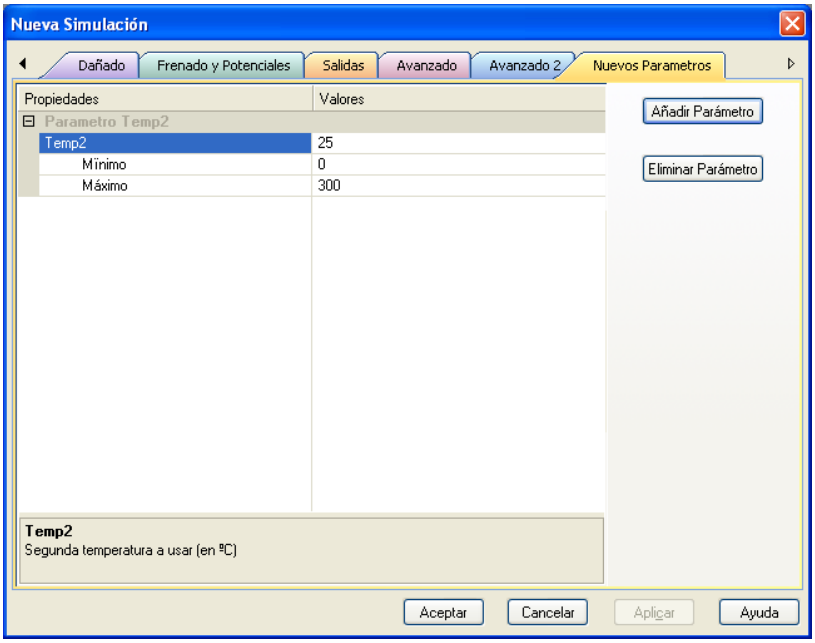

Figura 3.7: Página de propiedades de Nuevos Parámetros

Las especificaciones para este apartado son bastante sencillas, simplemente se requiere poder incorporar cuatro tipos de parámetros que permitan seguir con la evolución del simulador aparte de las posibles evoluciones que sufra esta interfaz gráfica. Estos parámetros se unirán al archivo de entrada del simulador y es éste el que tiene que ser capaz de

adaptarse a la nueva entrada. Es previsible que el simulador ya esté esperando este nuevo parámetro aportado.

El análisis de esta ampliación del programa incluye el estudio de la creación de los ficheros de entrada del simulador para poder incorporar estos nuevos parámetros y que tengan el mismo formato que los ya definidos, y el análisis del desarrollo de las nuevas ventanas necesarias y su colocación dentro de la aplicación.

Como el fichero de entrada al simulador se obtiene de las propiedades de la simulación y de todos los valores que están ahí definidos se acuerda que este apartado de definición de nuevos parámetros se encuentre localizado también como una hoja de propiedades de la aplicación.

Se decide realizar una pequeña pantalla para la poder añadir nuevos parámetros a la que se accede desde la ventana de previsualización de los parámetros ya definidos.

3.2.2. Implementación

Lo primero a realizar es añadir una página de propiedades nueva al programa que se dedique a leer los datos de los parámetros ya definidos desde un fichero externo como operación inicial al cargarse. Los datos son mostrados según el tipo al que pertenecen, mostrando más opciones en caso de parámetros de tipo numérico. Cada parámetro muestra una pequeña ayuda, si se ha definido, al ser seleccionado.

En la pantalla principal de la hoja de propiedades se sitúan dos botones, para acceder a la ventana de añadir nuevos parámetros o para eliminar el parámetro que se encuentre seleccionado.

La ventana de definición de nuevos parámetros presenta una interfaz sencilla, solo se permiten poner el nombre del parámetro a definir, una pequeña descripción del mismo a modo de futura ayuda y luego la definición del parámetro en sí, es decir, el tipo de parámetro y su correspondiente valor. El tipo del parámetro se selecciona de una lista desplegable que contiene los cuatro tipos de parámetros que se pueden definir:

- **BOOL:** Permite definir parámetros de tipo verdadero o falso.
- **INT:** Define un parámetro de tipo entero, hay que introducirle unos valores máximo y mínimo también en formato entero.

- **FLOAT:** Define lo mismo que el anterior solo que se permiten valores decimales.
- **STRING:** Con esta opción se permite definir una cadena de texto como parámetro.

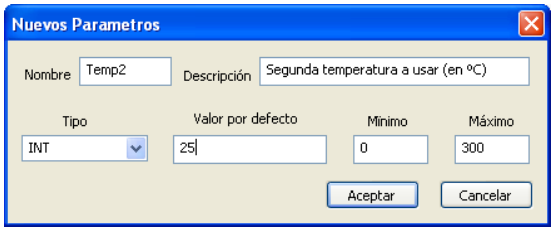

Figura 3.8: Cuadro de inserción de Nuevos Parámetros

Cuando se aplican los cambios de las hojas de propiedades o se aceptan los cambios realizados, todos estos valores de los parámetros definidos se guardan en un fichero de texto en el directorio de la simulación que se puede leer y modificar fuera del programa, eso sí,las modificaciones del fichero tiene que ser de acuerdo a los valores predefinidos. El fichero donde se guardan estos parámetros se localiza en el directorio de la simulación, por lo que ésta ha de ser guardada previamente para que dicho directorio exista, en caso de no ser así se indicará que se debe guardar la simulación previamente a la definición de nuevos parámetros [B.4.2].

3.2.3. Pruebas

Las pruebas del funcionamiento del apartado de *Nuevos Parámetros* consisten en la definición de nuevos elementos, al menos uno de cada tipo, y la prueba de todas las opciones para comprobar que los errores se presentan en caso de necesitarlos y que las opciones aceptadas se visualizan sin problemas.

Cada vez que se añaden nuevos parámetros se comprueba que en la pantalla se muestra correctamente, añadiéndolos a los ya existentes sin eliminar ninguno. También se prueba a realizar cambios directamente en el fichero para comprobar si se cargan los valores modificados al cargar de nuevo las propiedades de la simulación.

Por último se revisa el fichero generado para servir de entrada al simulador para comprobar que los datos se añaden en este fichero en el formato correcto y se realiza la traducción de ciertos valores a los valores oportunos del fichero, como pueden ser los booleanos que aparecen como verdadero/falso a los valores numéricos 0/1.

Con todas estas pruebas se acepta la solución propuesta para la incorporación de nuevos parámetros a la interfaz e independizar de esta forma el desarrollo del simulador del de esta interfaz.

Capítulo 4

Desarrollo del Proyecto: Mejoras

En este capítulo se describen todas las mejoras realizadas sobre los distintos componentes, ventanas, menus, etc., de la aplicación. Como se ha explicado en el capítulo 2 estas son las partes que necesitan ser modificadas según las premisas del proyecto.

4.1. Idiomas de la Aplicación

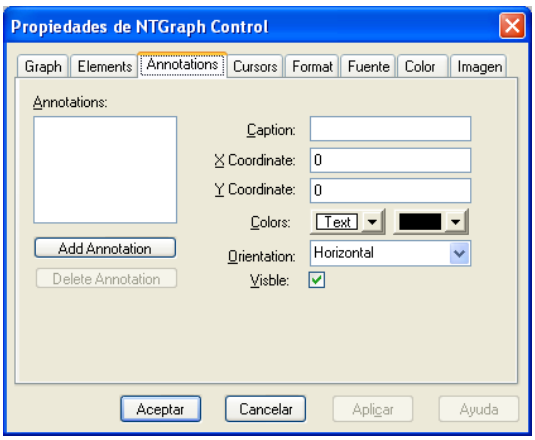

Figura 4.1: Página de propiedades de la visualización en dos dimensiones antes de la corrección del idioma

Se pretende que la aplicación pueda ser usada en castellano y en inglés, para ello es necesario poder tener la aplicación funcional en los dos idiomas. La aplicación ya permite la elección de un idioma u otro, pero presenta muchas deficiencias a la hora de utilizar la aplicación, presentando, en ciertas ocasiones, menús y ventanas de alertas en el idioma incorrecto.

La idea de los idiomas de la aplicación es que se pueda modificar el idioma en tiempo de ejecución del programa sin tener que reiniciar la aplicación como sucede hasta ahora, y presentar el objeto de gráficas en dos dimensiones en ambos idiomas, ya que hasta ahora solo se presenta en inglés. Obviamente, como el objeto para la visualización en tres dimensiones se he hecho nuevo ya cuenta con la posibilidad de mostrarse en los dos idiomas simplemente cambiando el valor de una variable.

4.1.1. Análisis y Resolución del Problema

La parte relativa al objeto de visualización de gráficas en dos dimensiones se tratará más adelante en el punto 4.2.

El paso de la aplicación a otro idioma se realiza mediante una dll que se encarga de hacer las traducción de todas las cadenas de texto del programa cuando el idioma seleccionado es Inglés. Conseguir la traducción a un tercer idioma consistiría con hacer otra dll del mismo formato y añadir las opciones oportunas para su carga, además de los valores a pasar a los distintos objetos para que tengan en cuenta el nuevo idioma.

El paso de un idioma a otro sin reiniciar la aplicación es difícil de lograr de lo esperado, llegando a la conclusión final de que debido a la arquitectura de la aplicación y la forma en que se definen los distintos componentes no se puede realizar. La opción de usar una dll para el cambio del idioma en lugar de un único archivo de cadenas tiene un problema: cuando se carga la dll del idioma ya no se puede regresar al idioma por defecto de la aplicación, el castellano.

Además, el uso de ciertos componentes visuales presentes en la aplicación hace que los menús no se carguen de forma dinámica sino que se hace al inicio de la aplicación, lo que también impide realizar el cambio del idioma incluso teniendo todas las cadenas en un único fichero.

Debido a los problemas expuestos la forma de atacar el problema tiene que tomar otro camino reutilizando la manera en que se realiza en la actualidad.

La mejora de este apartado se hace en todos los componentes que se puede. Se intenta no depender del los componentes existentes e intentar librar ciertas deficiencias existentes debidas a estos objetos. Se corrigen todas las cadenas de texto puestas entre el código, sustituyéndolas por variables dependientes del idioma [B.4.3, y añadiendo dichas cadenas

a la tabla de cadenas, de forma que se busquen en la dll del idioma o al programa principal si es el idioma por defecto.

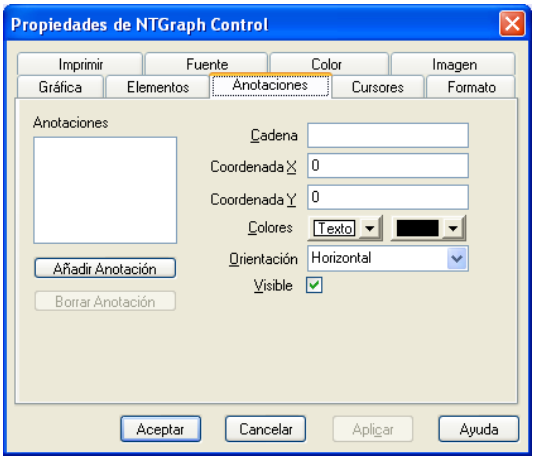

Figura 4.2: Página de propiedades de la visualización en dos dimensiones corregida

Debido a la envergadura de la aplicación y las múltiples funciones y apartados no se pueden abordar todos estos cambios de forma global. Los cambios de idioma se tienen que realizar de forma gradual por los nuevos componentes, verificando que los cambios aplicados no estropean otras partes de la aplicación.

La aplicación sigue necesitando reiniciarse antes de aplicar el cambio de idioma en alguno de los componentes por todos los problemas que se han expuesto con los menús. Toda la aplicación podría funcionar en ambos idiomas sin tener que reiniciar a excepción del menú principal, el cual se mantiene inamovible hasta que se reinicie la aplicación. Debido a esta limitación es preferible que la mayoría de componentes no cambien de idioma hasta que se haga efectivo el reinicio de la aplicación. Para así no confundir demasiado al usuario.

4.1.2. Pruebas

Las pruebas de esta parte no presentan mucha dificultad, solo hay que recorrer toda la aplicación, apartado por apartado, distinguiendo que cadenas que se muestran en cada unos de los idiomas en los que está disponible la aplicación y corroborar que las cadenas mostradas son las que se corresponden con el idioma elegido.

Como hemos indicado en el apartado anterior el desarrollo se hizo por partes, lo que facilita las pruebas ya que son pocas las cadenas que se tienen que verificar en cada revisión.

4.2. Visualización en Dos Dimensiones

La visualización de gráficas en dos dimensiones se realiza a través de un objeto ActiveX externo al programa principal [20]. Este componente provee al programa principal de grandes funcionalidades, pero tiene muchos defectos de visualización para poder realizar las tareas que requiere esta aplicación.

El componente ha sido modificado considerablemente desde el original para poder adaptarlo a las necesidades, pero aún así se ha descubierto con el uso que se necesitan nuevas funcionalidades para hacerle más completo. Prácticamente necesita retoques en todos los apartados, componiendo la tarea de mejoras más compleja de todas las acometidas en el programa.

4.2.1. Requerimientos

En esta sección del programa, como ya se ha indicado, existen muchas deficiencias. La mayoría de ellas tienen que ver con la muestra de datos y rótulos cuando los datos están en escala logarítmica. Se pide que se corrijan todos estos fallos.

El objeto permite la representación de más de una gráfica en cada momento, por eso es necesaria que se muestre una leyenda con todos los datos de cada gráfica dibujada. La leyenda tiene que poder resituarse y ciertos colores deben ser seleccionables, otros se deben corresponder con los de representación de la gráfica a la que corresponde la leyenda.

También se desea que se añada una nueva hoja de propiedades que permita elegir los tamaños para la impresión de las gráficas. y que desde aquí también se permita guardar la imagen en un archivo o copiarla al portapapeles.

El último requisito es que, al igual que la aplicación principal, soporte la presentación de mensajes y ventanas en varios idiomas [**??**], y que éste sea el mismo que el de la aplicación principal.

4.2.2. Fase de Análisis

Por suerte, el objeto no es demasiado grande y el estudio de funciones no es demasiado complicado. Además tiene mucho en común con el objeto de visualización de gráficas en 3D que se ha realizado para la ampliación.

Se comienza por los apartados en logarítmico, observando como funciona el objeto el este método de visualización y anotando todos los errores para que sean corregidos. Se encuentran algunos errores graves en la muestra de cursores y anotaciones, ya que las coordenadas mostradas en la gráfica no se corresponden con las reales de situación de los objetos. Se concentran muchos errores en la opción de **Snap on Element**, esta opción permite mostrar dos cursores (líneas de guía) por donde se mueve el ratón. Este apartado solo consiste en la modificación de métodos y rutinas ya existentes para corregirlos y adaptarlos.

La muestra de la leyenda era una funcionalidad nueva, al menos en lo referente a los colores y la colocación de la misma. Además se tienen que mostrar en una única leyenda los datos de todas las gráficas mostradas, anteriormente cada gráfica ponía su leyenda según sus propios datos. Para este apartado se tienen que crear nuevas variables y métodos de referencia que tendrán que ser accesibles desde la aplicación principal para poder realizar los cambios.

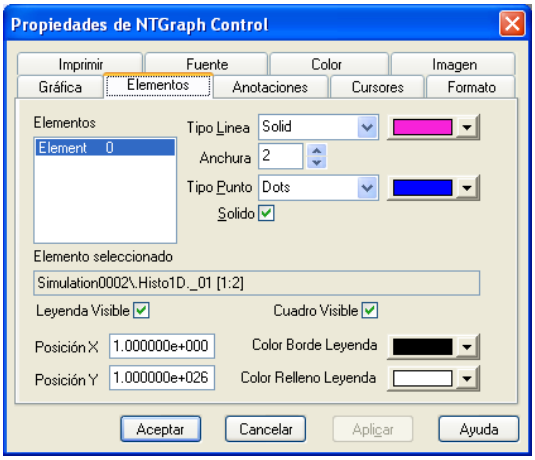

Figura 4.3: Página de propiedades *Elemento*

La opción de tener una nueva hoja de propiedades con las opciones de impresión, guardado y copia al portapapeles no parece ser demasiado complicada teniendo en cuenta que dichas opciones ya existen y solo hay que cambiar el lugar del que se llaman.

Además de estas correcciones grandes, también se detectaron otros errores o deficiencias menores, que por su problemática sencilla o tiempo de corrección prácticamente nulo no las vamos a nombrar. Simplemente decir que la mayoría de ellas se correspondían con desplazamientos o ubicaciones inadecuadas de objetos por no estar debidamente calculados, se olvidó de sumar los desplazamientos de los bordes o similares.

Lo primero de todo que ha realizar para mejorar el objeto es corregir los límites máximos y mínimos de los ejes en logarítmico. La hoja de propiedades *Gráfica* es la que permite definir los rangos de los ejes. En estos rangos se permite poner el máximo menor que el mínimo y el mínimo mayor que el máximo, se corrige este defecto añadiendo un control a los valores introducidos y mostrando mensajes de error en caso de que ocurra la casuística planteada. También se muestra un error al indicar que se quiere que los ejes estén en logaritmos en lugar de en lineal si los rangos contienen valores negativos; en este caso se pone el valor mínimo a 1 si el valor anterior era negativo, y el máximo siempre superior.

4.2.3. Solución

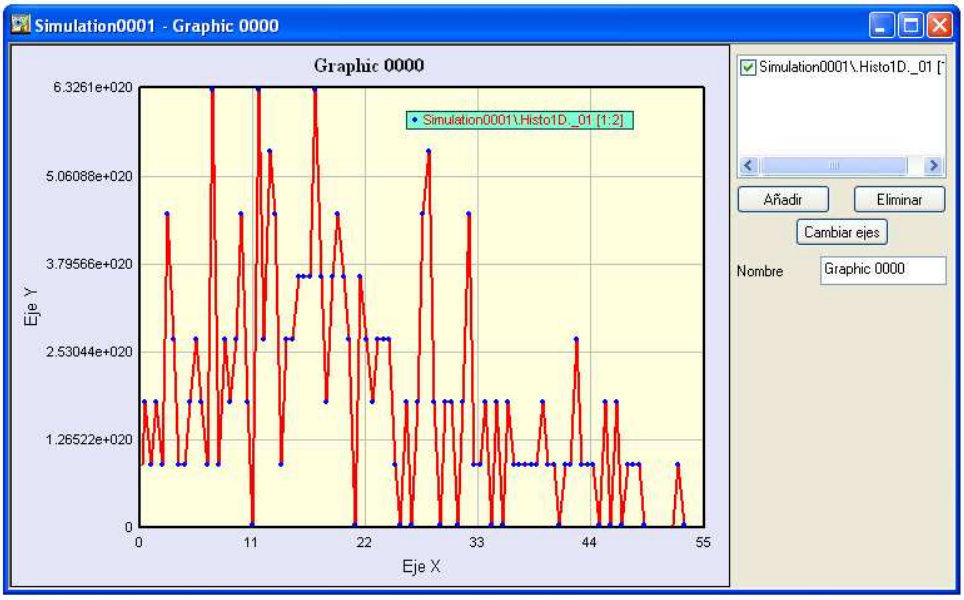

Figura 4.4: Gráfica 2D con visualización de leyendas

La siguiente mejora que es la corrección de los cursores [B.2.2]. Esta función se localiza la aplicación como *Snap on Element*. Los cursores funcionan casi correctamente en el

modo lineal, pero en cuanto se seleccionaba la opción de ejes logarítmicos (cualquiera de los dos ejes) esta opción ya no se correspondía con los puntos representados, las líneas que representaban el cursor no se colocaban sobre los puntos de la gráfica sino en cualquier otro sitio. En el modo lineal también existen errores ya que no se sitúa exactamente sobre el punto más cercano a la posición del ratón sino encima del primer punto por la izquierda que se encuentre a la altura del ratón. El fallo en logarítmico se debe a que el procedimiento que realiza los cálculos de posición esta pensado para trabajar en lineal y no contempla la opción de tener los ejes en escala logarítmica. Se corrige el procedimiento calculando los valores para lineal y para logarítmica de manera diferente dependiendo de si las variables asociadas a dicha representación están activas o no. El problema de la situación de cursor sobre el punto más cercano al ratón se debe a un error en la función matemática que se usa para detectar el punto más cercano. Se corrige dicha función con los cálculos más adecuados para la presentación correcta.

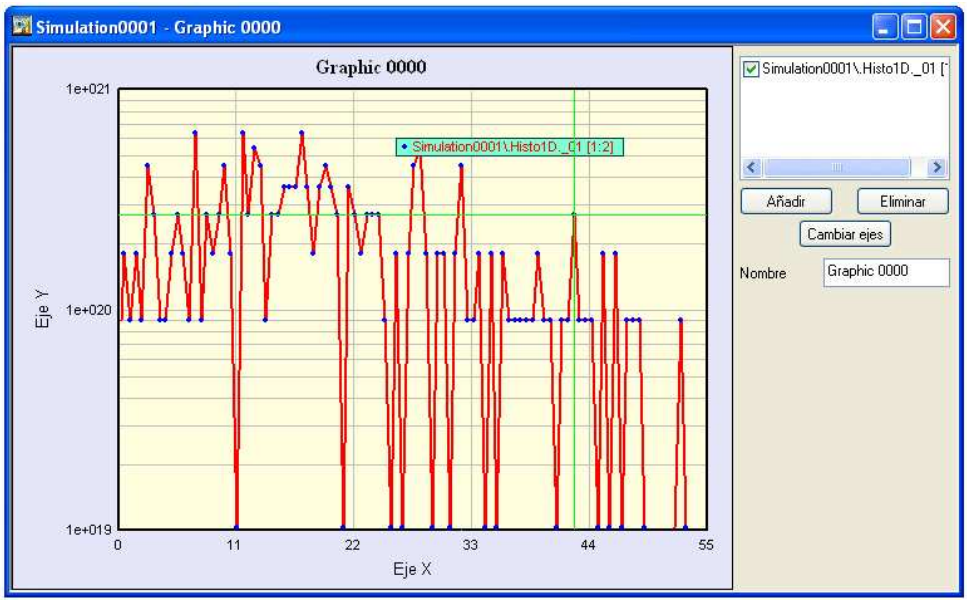

Figura 4.5: Muestra de punteros en escala logarítmica modificada

Realizando la mejora anterior se descubre que el método de desplazamiento (llamados *PanX*, *PanY* y *PanXY*) no funcionan en logarítmico por le mismo problema. El desplazamiento en lineal si se realiza correctamente porque es unitario, pero en logarítmica la gráfica se estropea. La realización del desplazamiento en logarítmico es más problemático porque es dependiente de los ejes, que son potencias de 10 (otra cosa no tiene sentido). Si el modo logarítmico esta activado en el modo de desplazamiento elegido, se toma co-

mo referencia el punto donde se ha apretado el ratón, y si se desplaza una década en la dirección logarítmica se producirá el desplazamiento. No se produce un desplazamiento continuo como cuando los datos son lineales sino por saltos de décadas.

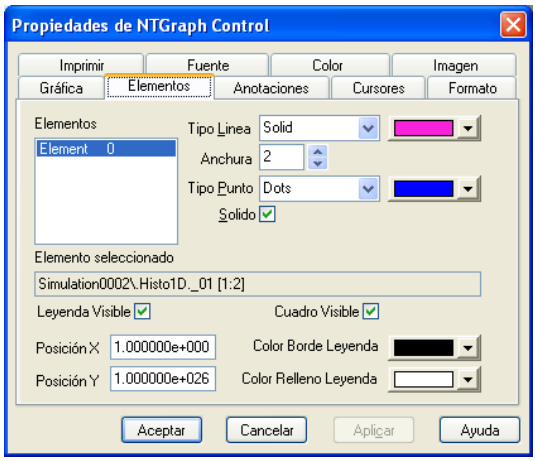

Figura 4.6: Página de propiedades con las opciones de la leyenda

Las opciones de la leyenda se sitúan en la pestaña de las propiedades llamada *Elemento*. Esta pestaña contiene las definiciones de colores y formas de representación de la gráfica, por lo que es el lugar más acertado para poner también los colores y la posición de la leyenda. La leyenda puede mostrarse o esconderse mediante una variable, así como el fondo de la misma. Los colores nos definen el borde de la leyenda y el relleno interior del mismo, el color de la letra mostrada se corresponde con el color de la linea de la gráfica en cuestión. Si se muestra más de una gráfica el color de borde y relleno, así como las coordenadas, serán comunes a todas las gráficas, la leyenda mostrará todos los nombres de las gráficas uno detrás de otro, precedidos por el topo con el que se dibujan los puntos. Los topos son del mismo tamaño del que se dibujan en la gráfica. Puede consultarse el código relativo a esta parte en B.2.1.

La última parte es la adición de una nueva hoja de propiedades [B.2.3] al objeto con una serie de botones para poder realizar las acciones de guardado de gráficas y copia al portapapeles y otro para poder realizar la impresión, pero que necesita que se le indiquen los valores de alto y ancho con los que queremos imprimir, para que la gráfica salga del tamaño deseado. Como se ha indicado en el punto anterior las tres opciones ya están disponibles por lo que el desarrollo sólo consiste en poner los controles y asignarles los eventos que llaman a las funciones ya definidas. El único desarrollo que se hay que realizar

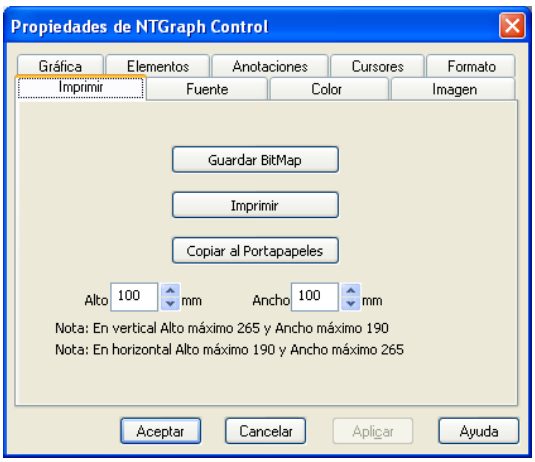

Figura 4.7: Página de propiedades *"Imprimir"*de las gráficas en 2D

es para poder introducir unas dimensiones para la impresión de la gráfica, antes siempre se imprimían al mismo tamaño.

4.2.4. Pruebas

Las pruebas de las modificaciones se van realizando conforme se acaba una mejora. Como las mejoras son independientes entre sí, se puede permitir realizar las pruebas de esta manera, consiguiendo así reducir el tiempo de pruebas al tener que probar partes pequeñas en cada fase.

La validación final de las opciones mejoradas en este objeto se realiza también por partes, aunque no por componentes como las pruebas.

Las pruebas consisten en la representación de varias gráficas con el objeto modificado y la observar los cambios producidos en la visualización de las mismas con los cambios introducidos. Normalmente las pruebas revelan nuevas deficiencias en la implementación realizada. Como ejemplo de esto digamos que la corrección de la muestra de cursores mostró una deficiencia en la muestra de las anotaciones.

4.3. Ejecución Batch

El programa ya es capaz de realizar las simulaciones deseadas en local o en remoto, facilitando los datos oportunos para cada una de las opciones. La ejecución en Batch hace

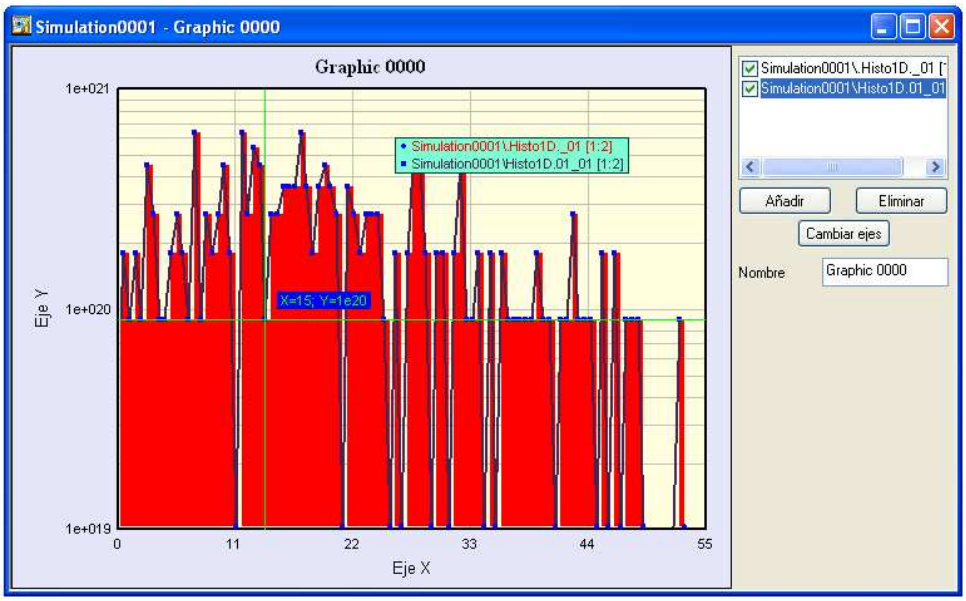

Figura 4.8: Objeto de visualización en dos dimensiones con todas las mejoras aplicadas

lo mismo que la ejecución en remoto solo que se puede romper la sesión mientras dura la ejecución de los comandos, permitiendo recoger la información en otro instante.

Esta necesidad se plantea porque una simulación puede tardar mucho tiempo en realizarse dependiendo de las características elegidas, podemos hablar incluso de horas, por lo que es necesario poder dejar la simulación trabajando sin tener que estar con la aplicación encendido.

Además de poder realizar las simulaciones en modo batch también se quiere que los datos puedan ser recogidos en cualquier momento, mientras dura la simulación o cuando ésta ya se encuentre finalizada.

Las especificaciones de este apartado no incluye mayores peticiones que esas. Pero bajo una petición tan pequeña no se encuentra un problema del mismo tamaño.

4.3.1. Estudio y Análisis del Problema

Para conseguir realizar la simulación en Batch, primero es necesario estudiar a fondo cual es el mecanismo de funcionamiento de los otros tipos de simulación, los únicos disponibles hasta ahora.

Las comunicaciones se realizan mediante hilos y utilizando una librería diseñada para implementar el protocolo SSH por lo que hay que revisar bastantes partes del programa principal y de la librería mencionada.

Una vez que sabemos como atacar el problema solo es necesario encontrar la manera de que el usuario defina que quiere realizar una simulación en modo batch y como se van a recoger los datos después.

4.3.2. Implementación

Lo primero es definir un nuevo parámetro dentro de la ventana de **Ejecución** con el nombre de **Batch** del tipo *checkbox* con el que se permita elegir este tipo de simulación.

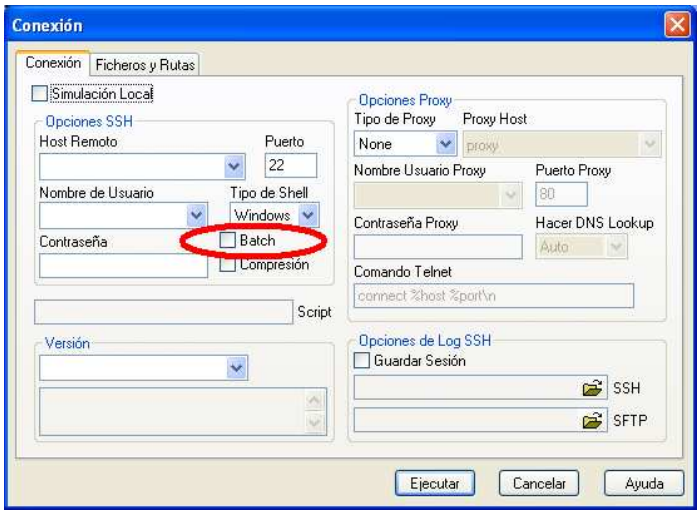

Figura 4.9: Detalle de la opción de ejecución en Batch

Luego se definirán dos nuevos hilos [B.4.5], uno para subir los datos y otro de comunicación, de manera similar a como están definidos para la ejecución remota. También se necesita la modificación de la instrucción a ejecutar para que se quede en background. A partir de aquí la simulación ya se está ejecutando en modo Batch, se ha establecido la comunicación, se está ejecutando la simulación, pero no estamos conectados al servidor. Una vez que la ejecución se ha iniciado con éxito la aplicación se desconecta del servidor, en este momento ya se puede cerrar la aplicación sin perjuicio de la simulación.

Aún queda la otra parte, la de recoger los datos que ha generado la simulación en Batch. Para recoger los datos hay que reconectarse de nuevo al servidor, por lo que se

provee de una nueva ventana a la aplicación para recoger los datos de: servidor, puerto, usuario y contraseña; para poder restablecer la comunicación. Esta opción está situada en el menú de *Propiedades* con el nombre *Actualizar*. Cuando se selecciona la opción de realizar la actualización se bajan a local todos los datos que se encuentren generados en la máquina indicada, de la misma forma que se hace en los otros tipos de ejecución periódicamente. Se pueden actualizar los datos todas las veces que se desee ya que no se borran de la máquina donde se ha realizado la simulación. Los datos se actualizan en el *Árbol de simulaciones* para poder trabajar con los ficheros que ha dado el simulador como salida.

Figura 4.10: Cuadro de petición de información para recogida de resultados

4.3.3. Pruebas

Para probar esta parte es necesario el permiso y un usuario para poder realizar pruebas en remoto contra servidores de la Universidad de Valladolid, concretamente en el servidor *trivial.ele.uva.es*.

Lo primero en probarse es el funcionamiento en remoto, y a continuación se comprueba el funcionamiento en modo batch. El resultado obtenido con el modo remoto y el modo batch, una vez actualizados los datos tiene que ser el mismo. Pero estos datos han sido obtenidos de la misma simulación, realizando los dos modos seguidos uno detrás del otro y en franjas de tiempo muy cortas, por lo que no es seguro que las pruebas sean correctas o solo falsos positivos.

Se vuelve probar el modo de ejecución batch con una nueva simulación, con datos totalmente distintos a los de la simulación anterior y no se descargan los datos, sino que se desconecta al ordenador de la red, cerrando la aplicación y apagando el ordenador

origen. Varios días más tarde se procede a actualizar los resultados de la simulación, pudiendo recuperar todos los archivos, con los que se puede trabajar de forma normal.

Con esta segunda prueba ya se puede concretar que la mejora introducida funciona correctamente y que permite realizar simulaciones y recoger los datos pudiendo pasar tiempo e incluso desconectar el ordenador de la red desde que se lanza la ejecución de la simulación hasta que se recogen los datos.

4.4. Control de Versiones

La ventana de control de versiones permite definir los ejecutables y los script con los que posteriormente se realizará la ejecución de la simulación. Dependiendo del sistema operativo tendremos que seleccionar una opción u otra. Se permiten definir tantas versiones de simulación como se quiera, permitiendo toda clase de combinaciones entre simulador y script de simulación, permitiendo incluso repetir todos los datos. Hemos de tener en cuenta que no todos los scripts puede hacer pareja con todos los modos de simulación, si escogemos mal la pareja el simulador no será capaz de ejecutar y dará error que será mostrado en las ventanas preparadas para tal efecto.

El *Control de Versiones* es accesible como una opción más dentro de las herramientas de la aplicación, pero se desea que a la ventana se pueda acceder directamente desde el menú y con un icono en la barra de tareas y no mediante el camino existente ahora. Además también se desea que el ejecutable y el script se seleccionen en la misma pantalla y queden configurados en una versión, y no en una pantalla aparte como sucedía antes.

4.4.1. Desarrollo

La página de *Control de Versiones* es accesible desde *Herramientas* ⇒ *Opciones*, y desde aquí podemos crear nuevos grupos de ejecutables. El script se indica en la ventana de *Ejecución*, donde también se escoge una de las versiones diseñadas con el *Control de Versiones*.

Se precisa de un estudio sobre las dos pantallas involucradas para poder realizar el cambio. Se han revisan las variables y controles necesarios a modificar, como se muestra en B.4.6. Al final se decida cambiar de idea y la ventana de *Control de Versiones*

Figura 4.11: Cuadro de Control de Versiones antiguo

seguirá apareciendo dentro del apartado de *Opciones*, pero también se le creará una entrada en el menú para que nos lleve directamente a esta parte y que, además, la muestre de forma única.

Lo primero a realizar de todas las tareas pendientes es incluir un nuevo apartado para poder seleccionar el script deseado dentro de la ventana, el tipo de control elegido es el mismo que se utiliza para elegir el ejecutable. No se elimina el apartado de selección de script que existe en la ventana de *Ejecución* sino que solo se pone en modo lectura, para poder presentar el script elegido en ese campo cuando se seleccione la versión de simulador a utilizar pero que no se pueda modificar.

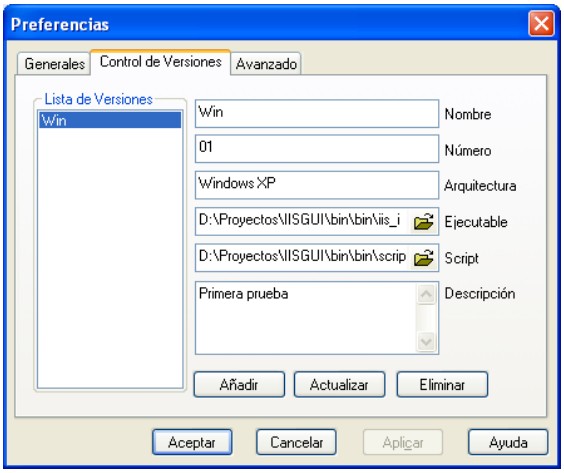

Figura 4.12: Cuadro de Control de Versiones Mejorado

El nuevo campo **script** en el *Control de Versiones* se guarda conjunto con al resto de parámetros que se pueden definir, y se carga en la simulación de la misma manera.

La ventana de *Control de Versiones* forma parte del conjunto *Opciones* como una pestaña. Lo que se a hace para poder mostrarla por separado es crear una nueva entrada en el menú *Herramientas* llamada *Control de Versiones* para acceder directamente a esta parte. Además se añade un nuevo icono en la barra de tareas que realiza las mismas funciones que el menú, es decir, lleva directamente a la ventana del *Control de Versiones*. El control sigue funcionando exactamente igual que antes solo que ahora tiene una opción más que elegir, la correspondiente al script, y la ventana de *Ejecución* tiene una opción de selección menos aunque no de visualización.

4.4.2. Verificaciones

Los cambios reales introducidos no son demasiados, por lo que las pruebas no requieren demasiada profundidad. El control ya funcionaba anteriormente de forma correcta, por lo que solo se ha comprobado que el nuevo campo es grabado correctamente y que se puede recuperar para ponerlo en la ventana de *Ejecución*, y que se carga también si deseamos modificar cualquiera de las versiones que hemos definido desde el propio control.

4.5. Lista de Simulaciones

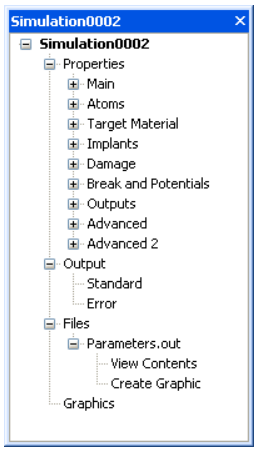

Figura 4.13: Árbol de simulaciones antiguo de la aplicación

La lista de simulaciones muestra ciertas características de las simulaciones. Estas características se pueden mostrar u ocultar según el gusto del usuario, pero no se pueden modificar. Esto significa que siempre se muestran las mismas características. El control de la lista de simulaciones proporciona muchas propiedades que hacen muy útil la visualización de datos, pero si no podemos cambiar las propiedades de la simulación que se muestran entonces todas las funcionalidades del control no nos sirven de gran ayuda.

Se plantea la necesidad de que ciertas características de las simulaciones aparezcan en este control a petición del usuario. Se han de identificar cuales serán las características que pueden aparecer y el número máximo de éstas que se visualizarán a la vez.

4.5.1. Soluciones Planteadas

En un primer acercamiento, se realiza una solución utilizando para la selección de las propiedades mostradas una hoja de propiedades de la simulación, poniendo a valor verdadero o falso la característica que se quiere mostrar. Esta primera solución funciona solo a medias a tenor de los resultados obtenidos en las pruebas realizadas. La solución planteada funciona para una simulación, pero falla sistemáticamente si se pretende que funcione con abierta más de una simulación. Los datos se guardan como propiedades de la simulación y chocan si dos simulaciones presentan distintas características seleccionadas. Además, solo se refresca la simulación activa, quedando los datos del resto de simulaciones borrados de la lista.

En el segundo intento se aprovecha al versatilidad que tiene el control del *Árbol de simulaciones*[B.4.7]. Este control recoge todas las características de las simulaciones. Por ello, se ponen iconos diferentes delante de los textos de las características, indicando cuales se pueden seleccionar, cuales no y cuales están ya seleccionados. Además, los elementos seleccionados se guardan como propiedades de la aplicación y no de la simulación, por lo que todas las simulaciones activas al mismo tiempo tienen seleccionadas las mismas características. Con estas dos modificaciones y un refresco de todas las simulaciones activas cada vez que se realiza una nueva selección o deselección se consigue solucionar el problema y los datos se muestran correctamente en el listado.

EL listado presenta seis campos fijos que están seleccionados siempre y no se pueden deseleccionar, aunque si ocultar en el listado, a excepción del nombre de la simulación. Además de estos campos fijos se pueden seleccionar hasta un máximo de nueve por parte del usuario para hacer un total de quince características mostradas en la lista de

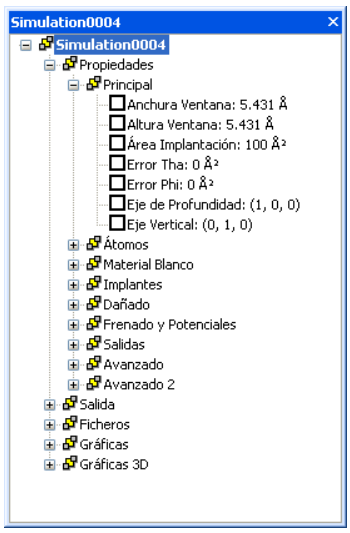

Figura 4.14: Árbol de simulaciones con opciones para mostrar en la Lista de simulaciones

simulaciones. Si una característica se encuentra repetida (una simulación puede tener muchos elementos de ese tipo de característica, *ejemplo: Proyectiles*) todas los elementos aparecerán en el mismo campo separados por coma uno detrás de otro. Se ha definido como nueve el máximo número de características seleccionables por el usuario, pero este valor puede cambiarse como se indica en el apartado 6.2.

4.6. Tabla de Elementos

La tabla de elementos es un recurso del programa que se usa para añadir nuevos átomos a la simulación. Estos átomos pueden servir luego para formar parte del material blanco o para servir de dopantes.

La ventana que se muestra en la aplicación está formada por dos imágenes, una de las imágenes presenta todos los elementos y la otra contiene los negativos, que se colocan sustituyendo al átomo sobre el que se sitúa el cursor. El modo de visualización no es malo, pero esta forma de proceder impide que se pueda poner en distintos idiomas.

Por las razones expuestas se decide sustituir la solución actual por un componente externo que se encargue de la muestra de una tabla periódica de elementos [B.3.1]. La tabla se compone de todos los elementos conocidos, mostrando su símbolo, el nombre del elemento en el idioma de la aplicación, y el número y el peso atómicos. Los nombre de los elementos se leen de un fichero de texto, existiendo uno para cada uno de los idiomas

permitidos. Los ficheros se pueden modificar, de forma que si uno de los elementos que actualmente no tiene nombre, en el futuro se le pone, se le puede cambiar.

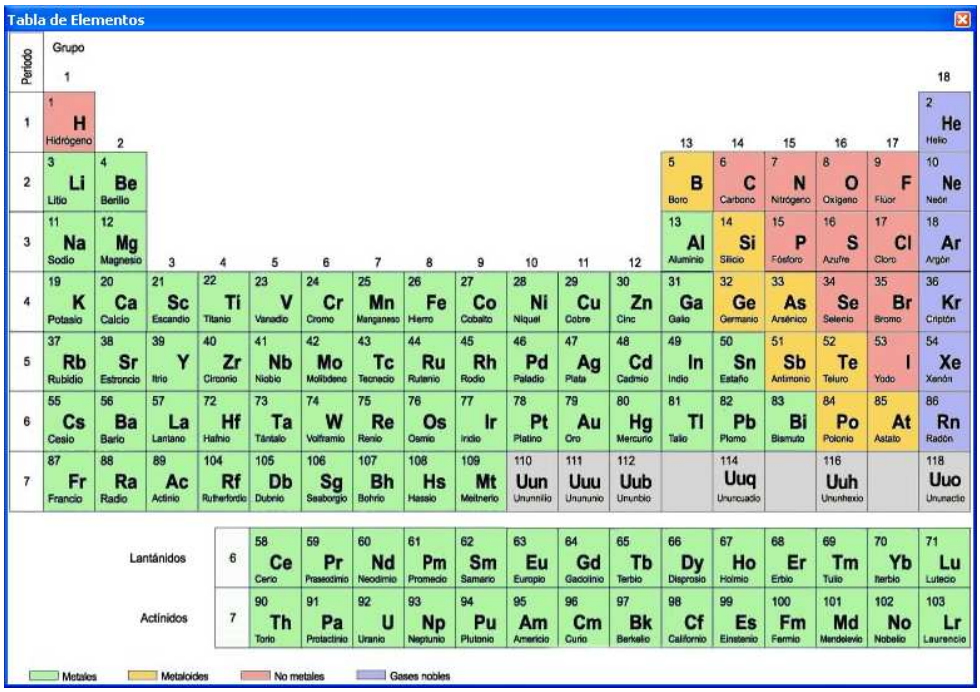

Figura 4.15: Anterior tabla de elementos

Para desarrollar este nuevo componente se utilizan procedimientos de OpenGL que facilitan el dibujado. Se realiza un dibujado por capas, pintando en primer lugar los colores de fondo correspondientes a cada zona de la tabla, luego se pintan los bordes de la tabla y la rejilla y por último los textos. Cuando el ratón se mueve por encima de la tabla, el elemento marcado aparecerá con un borde rojo para que se identifique cual es el elemento que se esta seleccionando. El elemento se elige haciendo click derecho sobre el elemento deseado, que aparecerá automáticamente en la lista de elementos definidos de la aplicación principal, permitiendo elegir aquí el isótopo si se ha cargado correctamente la tabla de isótopos.

El desarrollo esta mejora es relativamente sencillo después de todo el estudio realizado para el componente de visualización en tres dimensiones y las mejoras realizadas sobre el de dos dimensiones. Comparando este objeto con los otros dos podemos afirmar que es mucho más sencillo y pequeño, aunque igual de importante que el resto.

4.6.1. Características Importantes

Para que este elemento funcione correctamente es necesario que se le indique, mediante la llamada a un procedimiento público propio, donde se encuentra ubicado el directorio que contiene los fichero de los idiomas, en el caso de que en dicha ubicación no encuentre el fichero del idioma correspondiente lo creará, avisando al usuario de la ruta del fichero para su posterior modificación, en caso de que se crea conveniente. Si el programa siempre le ofrece al objeto la misma dirección, como está configurado en la aplicación **IISGUI**, el archivo solo se creará la primera vez y, a partir de ahí, usara el fichero generado. .

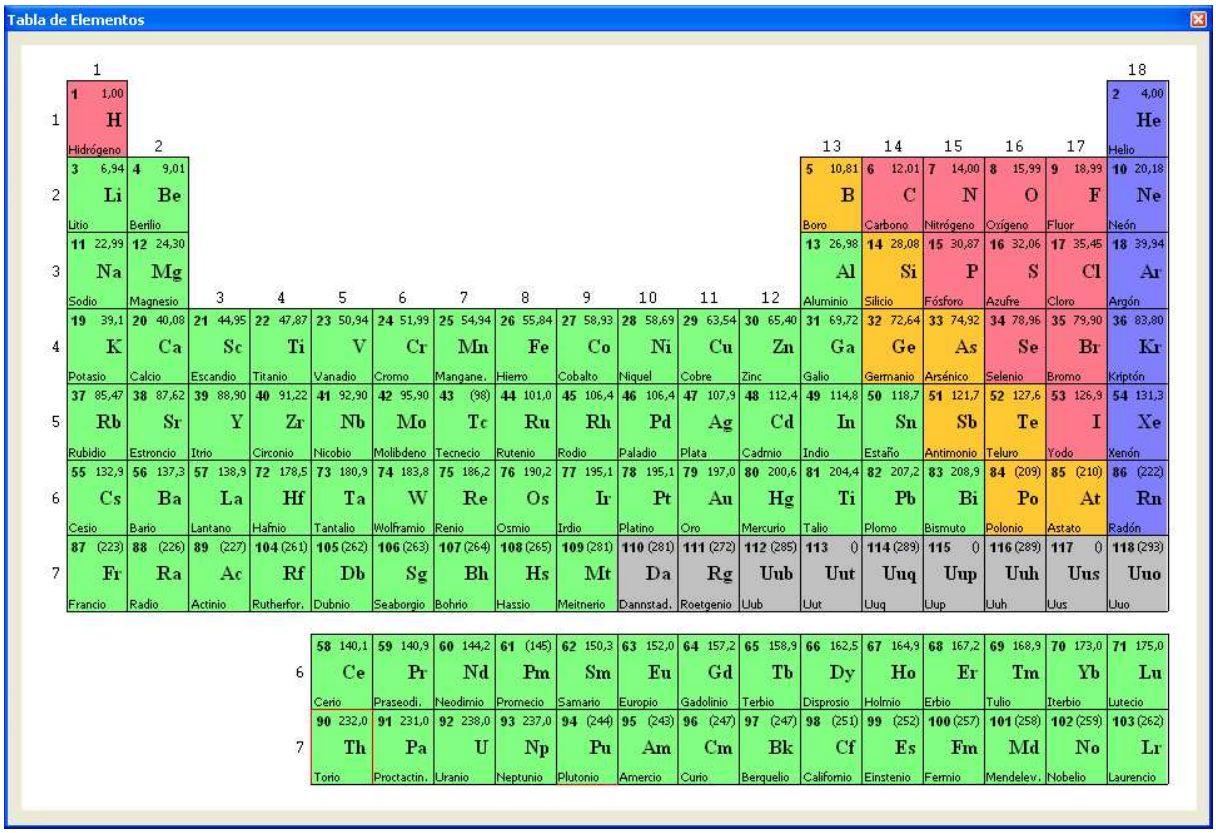

Figura 4.16: Nueva tabla periódica de la aplicación

El objeto no devuelve ningún parámetro cuando se cierra, sino que hay que realizar una llamada a otro procedimiento público preparado para devolver el valor del elemento seleccionado [B.4.4]. El valor devuelto se corresponde con el valor del número atómico del elemento seleccionado.

4.7. Atajos de Teclado

Es necesario que las opciones más importantes del programa puedan ser realizadas pulsando una combinación de teclas. Esto reduce el tiempo de acceso, ya que no hay que navegar por los menús para buscar al opción a seleccionar.

Además, se pretende que todos los atajos de teclado para las principales opciones de la aplicación se guarden en algún lugar, como puede ser un fichero en formato texto, de forma que se puedan modificar en cualquier momento y añadir más de manera sencilla.

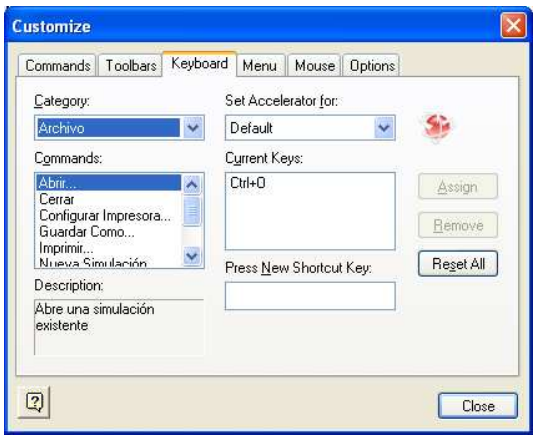

Figura 4.17: Opción que permite definir los atajos de teclado

4.7.1. Resolución

Para comprobar el funcionamiento de los atajos es preciso realizar un estudio sobre la aplicación viendo los atajos de teclado definidos y como se definen otros nuevos, para poder encontrar un patrón que permita guardar los atajos en un fichero conjuntamente con la acción a la que van asociados.

Como la aplicación está realizada con objetos que permiten la mejora de las interfaces gráficas, también se estudió el comportamiento de estos objetos, en este caso se usa el componente **BCGControlBar**.

Una de las opciones principales del control **BCGControlBar** permite la personalización de los menús y las barras de herramientas de la aplicación. Una de esas opciones incluye la posibilidad de definir los atajos de teclado. El componente permite realizar el

cambio de todos los atajos de teclado definidos en el menú desde la pestaña *Keyboard* dentro de las opciones del control.

Dado que se puede acceder a la pestaña anteriormente mencionada haciendo clic derecho sobre la aplicación y escogiendo la opción *Personalizar...* (o *Customize...* según el idioma de la aplicación) se usará esta posibilidad en lugar de realizar cualquier otro desarrollo, para así utilizar más características del objeto de mejora de interfaces.

En las mismas opciones de personalización existe otra opción que permite que todos los menús se desplieguen completamente aunque nunca hayan sido mostrados. Esta opción se encuentra en la última pestaña, *Options*, y está formada por las dos últimas líneas de la vista.

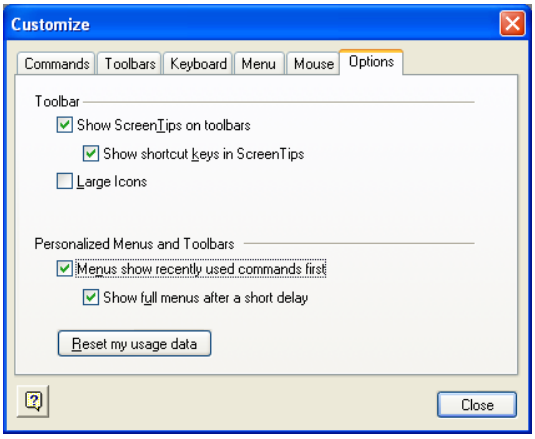

Figura 4.18: Opción que permite desplegar todos los menús sin haber sido usados

Estas propiedades son propias del objeto **BCGControlBar**, no se puede hacer nada para que aparezcan en un idioma distinto del que fueron programadas, por lo que aparecerán en inglés independiente del idioma en el que se ejecute la aplicación.

4.7.2. Comprobaciones

Para verificar que las opciones funcionan, a pesar de lo que digan los desarrollados que se encargan del objeto, se realizan pruebas de funcionamiento consistentes en la definición de nuevos atajos de teclado y el cambio de los ya existentes. No se realizan pruebas muy exhaustivas ya que el objeto se supone que ya se comercializa suficientemente probado.

4.8. Ficheros de Configuración en Modo Texto

Los fichero en formato texto facilitan el cambio de parámetros sin tener que utilizar la interfaz gráfica diseñada para modificar los archivos de entrada del simulador. Actualmente se tiende a que todo lo que se pueda se ponga en ficheros de configuración en modo texto, para que las aplicaciones sean parametrizables y no sean tan dependientes de las interfaces gráficas y de los desarrollos concretos, sino que puedan sufrir pequeñas modificaciones.

La aplicación **IISGUI** utiliza numerosos ficheros para guardar distintos parámetros necesarios durante su ejecución, además de muchas variables. Lo ideal es que todos estos valores se encuentren en ficheros fácilmente modificables.

4.8.1. Estudio y Análisis del Problema

Tener las variables guardadas en memoria es muy cómodo porque no es dependiente de las rutas de los ficheros, cualquier variable es accesible en cualquier momento desde cualquier parte si la buscamos de la forma correcta.

Todas las variables de datos que usa el programa se guardan en el registro, de forma que no se puede acceder a ellos ni cambiarlos si no es a través de la interfaz creada.

El programa contiene muchas variables y son guardadas desde lugares diferentes dependiendo de su uso, si éste es general se cargan al iniciar y se guardan el finalizar la aplicación, si son locales las acciones se realizan en ese ámbito.

Al final el proyecto cogió una envergadura suficiente y este apartado se queda fuera y sin realizar. Aún así cabe decir que el estudio si que se hizo y el cambio iba a requerir bastante trabajo para su consecución.

Debido a esta petición para el programa principal, todas las partes y objetos externos que se han desarrollado guardan sus datos en ficheros en modo texto, sin usar en ningún momento la memoria y pudiéndose modificar directamente desde un editor de texto cualquiera, ya que contienen entradas que definen que parámetro es cada uno dentro del fichero.

4.9. Otras Mejoras

Dentro de este apartado de **Otras Mejoras** recogeremos todos aquellos cambios de cierta envergadura o relevancia que se han realizado y que no se habían definido inicialmente en las especificaciones del proyecto.

Estas modificaciones han surgido de terceras personas que prueban el programa y se enteraron de la realización de esta ampliación y mejora, o simplemente son partes que se habían olvidado comentar en la solución inicial.

4.9.1. Autoguardado

El programa guarda todos las modificaciones realizadas a petición, mediante la opción de menú *Archivo* ⇒ *Guardar* o al salir de la aplicación, de forma automática, que se guardan todas las simulaciones abiertas. Pero si por cualquier extraña circunstancia el programa no acaba de forma correcta todas las modificaciones realizadas no se reflejarán la próxima vez que se abra el programa.

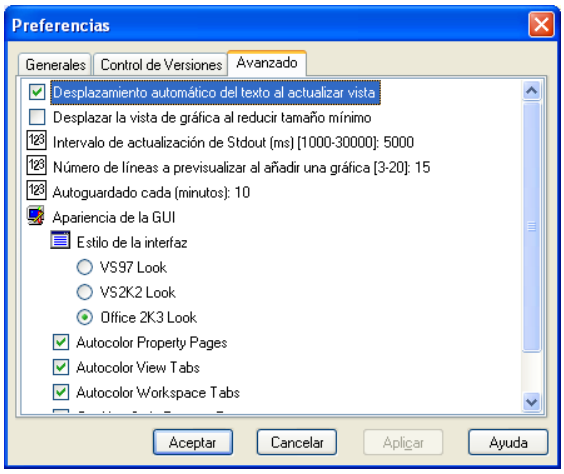

Figura 4.19: Vista de la opción de tiempo de autoguardado

Por lo comentado en el párrafo anterior se hace necesaria la inclusión de una opción de autoguardado o autosalvado de los datos que se ejecute periódicamente según un tiempo definido por el usuario. La opción de autoguardado tiene que trabajar igual que la función de guardado normal solo tiene que ser totalmente trasparente para el usuario.

El tiempo de autoguardado debe ser configurable desde las opciones principales del programa definidas en *Herramientas* ⇒ *Opciones*. El tiempo se define en minutos.

El funcionamiento del autoguardado consiste en un hilo que trabaja a la vez que la aplicación y que se "despierta" cada intervalo de tiempo definido en las opciones del programa para realizar el guardado de todos los valores de la aplicación. Se puede observar la definición del parámetro de las opciones y las definiciones de los hilos en el apéndice B.4.8

4.9.2. Menús

Algunos menús no se encuentran agrupados bajo los títulos correctos o adecuados y requieren una corrección. Para ello, es necesario dividir algún grupo de menús en dos o unir características similares bajo el mismo título principal. Estos arreglos son necesarios tanto en los menús principales como en los menús contextuales.

Destaca el cambio necesario en el menú principal. Se añade un grupo nuevo que recoge los eventos que se pueden realizar con las simulaciones, iniciarlas, pararlas, mostrar las hojas de propiedades o actualizarlas con los datos simulados en modo batch en algún servidor; se añaden apartados a otros menús y se renombrar grupos.

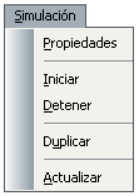

Figura 4.20: Menú principal de la aplicación modificado

Al primer grupo del menú principal se le cambia de nombre, siguiendo el estándar de todos los programas comerciales, dejando el título que presentaba para agrupar las opciones comentadas en el párrafo anterior. Al crear nuevas entradas de menú, también hay que crear nuevos controladores para esos eventos, ver [B.4.8].

4.9.3. Rutas de la aplicación

La aplicación utiliza numerosos ficheros externos de configuración o para contener distinta información, como pueden ser los parámetros por defecto de algunos apartados.

Algunos de estos ficheros también contienen las rutas de otros ficheros que se necesitan en ocasiones, como sucede en los ficheros del simulador y en el script que se van a usar para la simulación.

Todas las rutas de los ficheros, tanto las puestos directamente en la aplicación, como las contenidos dentro de otros ficheros, son direcciones absolutas. Se pretende que todos los ficheros hagan referencia a rutas partiendo de la posición del ejecutable, de forma que se eliminen todas las rutas absolutas de la aplicación.

Además de las rutas de los ficheros ya existentes en la aplicación, los nuevos ficheros de configuración o datos que se utilicen en los distintos objetos deben seguir también esta premisa.

Para conseguir que todas las rutas sean referentes a la ruta del ejecutable se necesita primero conocer esta ruta. La consecución de esta ruta no es tan sencilla como antiguamente, mediante el valor *arg[0]*, sino que hay que utilizar otros medios como se ve en B.4.8.

4.9.4. Memoria de documentos en uso

Para dar mayores facilidades a los usuarios que trabajan con el programa y que puedan cerrarlo y abrirlo y seguir trabajando en las mismas condiciones se ha habilitado una nueva característica que consiste en memorizar todos las ventanas con gráficas que se encuentran abiertas y visualizadas en pantalla, para que se vuelvan a abrir cuando lo hace la aplicación. De esta forma, cuando se vuelve a abrir la aplicación para trabajar la encontramos exactamente con la misma información en pantalla.

Para conseguir hacer funcionar la "memoria"se graban ciertos datos de todas las ventanas abiertas en un fichero dentro de la ruta de la aplicación. Cuando se vuelve a abrir el programa se carga de nuevo el fichero, leyéndose su contenido y realizando las llamadas oportunas para abrir de nuevo las ventanas.

De cada ventana se guarda en el fichero, es un fichero de texto plano que se puede leer con cualquier editor de textos, la ruta de la simulación con el nombre del fichero asociado que tiene el documento y el también se guarda el título que posee la ventana.

Esta parte ha requerido cambio de código en varias rutinas del programa, los más importantes cambios se muestran en el la siguiente referencia [B.4.10]. Ha sido necesario cambiar y crear rutinas en varias clases, pero no se muestra en este apartado todos los cambios pues son sencillos y no nos aportarían nada.

Capítulo 5

Desarrollo del Proyecto: Otras Funcionalidades

Las últimas partes que nos quedan por comentar son las de la creación de la ayuda y la de generar el distribuible de la aplicación. Para poder realizar estas partes es estrictamente necesario que todas las anteriores se encuentren finalizadas, ya que de no ser así estas partes necesitaran ser retocadas y corregidas antes de obtener la versión final.

5.1. Creación del Ayuda

El programa *Ion Implant Simulator GUI* presenta un archivo de ayuda que sirve como manual de usuario del programa, y ayuda a resolver los pequeños problemas que le pueden surgir al usuario durante el uso diario del programa.

La ayuda existente se encuentra en un estado muy precario, con unos contenidos muy pobres, muy pocas imágenes y con varios idiomas entremezclados, presenta partes en castellano y en inglés dentro de la misma ayuda, confundiendo al usuario. Además hay muchos apartados que no cuentan con su debida ayuda.

La intención es corregir todos estos defectos solucionando los problemas de idioma haciendo una ayuda en cada idioma para que en el programa se cargue la oportuna dependiendo del idioma de la aplicación. Además es necesario añadir los apartados relacionados con todas las mejoras realizadas en este proyecto, por otro lado, también es necesario corregir todos los apartados existentes.

Se da comienzo con por la ayuda en castellano, por ser lo más sencillo de realizar. Lo primero que hay que realizar es redactar la ayuda en este idioma, incluyendo todas las

mejoras que se han realizado durante el proyecto, por lo que no se puede dar por terminada la elaboración de la ayuda hasta que se corrijan todos los fallos y se realizan todas las ampliaciones requeridas y explicadas en los capítulos anteriores. Una vez revisada y verificada la ayuda en castellano se procederá a realizar una traducción de la misma al inglés para cuando la aplicación se use en este idioma.

Figura 5.1: Muestra de una página de ayuda de la aplicación

Es preciso revisar todos los apartados uno a uno, introduciendo nuevos apartados en caso de ser necesarios y corrigiendo y ampliando los ya existentes. Una serie de capítulos de la ayuda solo contienen información de ciertos parámetros del simulador (se corresponden con las propiedades de la simulación), estos capítulos están en inglés, por lo que se hace imprescindible traducirlos para la versión en castellano.

Una vez que esté la ayuda redactada se corrigen las imágenes de la aplicación necesarias y actualizándolas a la última versión para completar la ayuda y situar al usuario en el apartado correcto del programa. De esta forma se muestra más información al usuario sobre el tema tratado. Se van a cambiar las imágenes que existen en la ayuda aunque estén correctas para tenerlas todas con el mismo estilo de Windows. Las imágenes que presentaba la ayuda estaban obtenidas en un equipo que contenía un tema de Windows

distinto al original, lo que puede desconcentrar al usuario que use el Windows en el estilo clásico.

Tras la validación y aprobación de la ayuda en castellano se procede a realizar al versión en inglés. Se vuelven a recoger las imágenes del programa, pero esta vez utilizando al versión en inglés, para que la ayuda sea consistente y no presente textos en un idioma ye imágenes en otro distinto.

La aplicación ya reconoce el fichero de ayuda y la carga al iniciarse. Además también es capaz de distinguir el idioma en el que se ejecuta para utilizar la ayuda que corresponda. La carga de la ayuda en castellano es correcta, pero no así la carga en inglés, por lo que se hace necesario modificar alguna instrucción [B.4.9] para un correcto funcionamiento.

La realización de la ayuda no supone ningún problema ya que solo se trata de redactar texto en relación a la aplicación e insertar una serie de imágenes, pero si que lleva bastante tiempo. Aunque el programa de creación de la ayuda no se había usado nunca el aprendizaje fue rápido. La inserción de la ayuda en la aplicación tampoco es un proceso demasiado costoso ya que prácticamente la reconoce por sí solo, salvo las líneas que se han indicado que ha sido necesario modificar.

5.2. Software Distribuible

Una ver terminadas todas las mejoras y ampliaciones del programa, y concluida la revisión de la ayuda se realiza el distribuible del software. Toda aplicación que se precie tiene que tener una manera sencilla de instalarse en el cliente y de poder distribuirse para todos los usuarios.

Esta funcionalidad se realiza mediante el programa *InstallShield Express*. Este programa permite generar instalables para aplicaciones. Todo el programa y sus dependencias se aglutinan en un único fichero instalable, que se puede distribuir de forma sencilla.

En el caso que nos ocupa, el archivo distribuible no es demasiado grande, estamos hablando de que todo el conjunto de fichero necesarios para la ejecución de la aplicación ocupa poco más de quince megabytes de espacio en disco, aunque este tamaño puede crecer si se decide agregar alguna simulación de muestra en el software distribuido.

El proceso de creación del instalable es relativamente sencillo. Se toma como base el realizado en el anterior proyecto, pero, como casi siempre, es más sencillo y limpio

crear uno nuevo por completo. Debido a que no hay que realizar muchas acciones para la generación del instalable es más beneficioso partir de cero que utilizar algo hecho.

El programa presenta un tutorial con el que se rellenan casi todos los campos necesarios para el instalable, como son las rutas, los ficheros necesarios, etc. Pero hay que acceder a todos los apartados uno a uno para poder sacarle partido a la herramienta y configurar todas las opciones que nos son necesarias.

Capítulo 6

Notas de Interés

Este apartado incluye la localización de los parámetros que pueden ser modificables de forma sencilla para cambiar ciertas propiedades de la aplicación. Se incluye este apartado porque estos parámetros no son configurables desde la aplicación porque así se pidió, del mismo modo que se pidió que se indicase donde se encontraban localizados en el código.

6.1. Parámetros Generales

Aquí vamos a definir todos aquellos parámetros que es necesario tener configurados para poder compilar la aplicación y poder así continuar con el desarrollo.

- **Elementos a Instalar**

  Para poder compilar la aplicación necesitamos tener instalado el componente BCGControlBarPro en la versión 6.74, con todas las librerías no Unicode para *Microsoft Visual Studio .NET 2003* compiladas (esto es una opción durante la instalación).

  Además de este componente también necesitamos instalar el componente MDL-Chime, aunque en este caso solo es necesario para la ejecución y no para la compilación, y el programa funciona sin él aunque no muestra los elementos sobre los que se va a realizar la implantación.

- **Registro de Librerías**

  La aplicación utiliza varios objetos programados de forma externa a la aplicación y que funcionan de forma independiente. Por ello es necesario registrar mediante el comando **regsvr32** los componentes *NTGraph.ocx*, *NTGrpah3D.dll* y *Tabla Elementos.ocx*.

- **Librerías necesarias**

  La aplicación hace referencias a librerías propias de Windows. Durante las pruebas del programa en ordenadores normales se ha detectado que, al menos, se requiere tener las librerías *mfc71d.dll* y *msvcr71d.dll*.

- **Directorios de uso**

  Tenemos que configurar unos ciertos directorios en la herramienta *Microsoft Visual Studio .NET 2003* para poder hacer funcionar algunos controles y que no de errores de compilación.

  . Debemos acceder al menú *Herramientas* ⇒ *Opciones* y escoger la última opción del árbol, *Projects*. En el apartado *Mostrar directorios Para* escogemos la opción *archivos de inclusión*.

  Hemos de añadir la entrada donde se encuentra el control BCGControlBarPro, que puede coincidir con:
  Archivos de Programa\BCGSoft\BCGControlBarPro\BCGCNPro.

  También hemos de añadir en *Archivos de Biblioteca* la entrada:
  Archivos de Programa\BCGSoft\BCGControlBarPro\Release71.

6.2. Parámetros de la Aplicación

Este apartado recoge todas las peticiones planteadas por el tutor de parámetros que se puedan facilitar fácilmente pero desde el código, teniendo que volver a compilar la aplicación para que se hagan efectivos.

- **Lista de Simulaciones**

  En este apartado se ha decidido que el número máximo de columnas en la *Lista de Simulaciones* sea de quince, seis que son fijas y nueve para que las escoja el usuario a su gusto. Este parámetro puede cambiarse en el archivo **Preferences.h**, donde hay una entrada *#define MAX_MARCADOS 9* que indica que el número de columnas a marcar por el usuario es 9.

- **Nuevos Parametros**

  En el fichero **Simulation.h** se encuentra la entrada *#define MAX_PARAM 10*, que determina que el número máximo de nuevos parámetros que puede configurar el usuario es de 10.

- **Tabla de Elementos**

  En este componente tenemos fijados unos tipos de letra para las representaciones. Estos tipos se pueden modificar desde el fichero **Tabla ElementosCtrl.cpp**. En este fichero tenemos cuatro entradas similares, del tipo:
  static const FONTDESC _FontDescNombre = sizeof(FONTDESC), OLESTR ("Tahoma"), FONTSIZE(7), FW_NORMAL, ANSI_CHARSET, FALSE, FALSE, FALSE ;.
  Aquí puede cambiarse el tipo de letra y el tamaño de la misma, además, también puede modificarse el estilo que presenta dicho letra.

6.3. Otros Datos

Para el correcto funcionamiento de la aplicación se necesitan una serie de ficheros colocados en determinados directorios. Algunos de estos ficheros se pueden crear solos si no se encuentran en la ruta en la que deben presentarse, pero otros no, por lo que si no existen el programa presentará fallos o deficiencias.

Entre estos ficheros se encuentran todos los ficheros de configuración. Empezaremos nombrando los que son importantes puesto que no se crean automáticamente.

- **isotopes.dat:** Este fichero contiene los isótopos de los átomos. El programa puede funcionar sin él, pero no es capaz de crearle. Sin este fichero no se permite la selección de isótopos y muestra un error cada vez que se accede a las propiedades de la simulación. Este fichero debe encontrarse en el directorio **data**.

- **resdefdw.dat, tbldefdw.dat y tbldefup.dat:** Estos tres ficheros contienen los valores por defecto de los resultado a no bajar y las tablas a bajar y subir respectivamente. Sin estos archivos el programa puede funcionar pero hay que poner los valores a mano en el apartado correspondiente de las opciones de conexión. Deben encontrarse en el directorio **data**.

- **fichero.dat:** Este fichero es imprescindible para abrir una aplicación ya existente. Sin él no se pueden abrir las simulaciones puesto que contiene los valores de los campos que se muestran en la lista de simulaciones. Aunque no se puedan abrir simulaciones ya existentes si que se puede crear una nueva simulación, que creará este fichero si no existe, y luego ya se puede abrir cualquier otra simulación. Este fichero se encuentra en el mismo directorio que el ejecutable. En el directorio **iisgui**.

- **Tables:** Esto no es un fichero sino un directorio completo con los valores previamente calculados para la definición de ciertos átomos. Este directorio tiene que localizarse dentro del directorio **sims**.

Dentro de los fichero que no es necesario tener porque se generan automáticamente (todos estos ficheros se crean en el directorio **data** si no se encuentran) si no se encuentran tenemos :

- **Elementos_Espanol.txt y Elementos_Ingles.txt:** Estos dos ficheros contienen los nombres de los elementos de la tabla periódica en el idioma al que hacen referencia en el nombre.
- **versions.dat:** Este fichero contiene los valores de las de las versiones que hayamos definido desde el control de versiones para poder realizar las simulaciones. Este fichero se crea con la primera versión que se defina, si se borra el listado de versiones aparecerá vacío.
- **workspace.dat:** Aquí se encuentran guardados los valores de las simulaciones que se encontraban abiertas en el momento de cerrar la aplicación para volver a abrirlas al iniciarla de nuevo.
- **materials.dat:** Contiene el listado de materiales definidos por el usuario.

Capítulo 7

Conclusión

Comentaremos de forma breve las conclusiones sacadas de la realización de las mejoras aplicadas al programa y las líneas futuras que pueden seguirse para continuar con la evolución del mismo.

7.1. Conclusiones Generales

Tras la elaboración del contenido de este proyecto fin de carrera de Ingeniería Electrónica comprendo los graves problemas que supone el realizar modificaciones al software desarrollado por otras personas cuando las aplicaciones tienen un tamaño considerablemente grande y no están correctamente documentadas. Se pierde demasiado tiempo intentando comprender que es lo que realiza cada función y como ha querido dividir el desarrollador anterior los problemas que se le presentaron.

Tener que ampliar una aplicación de otra persona me ha hecho ver lo importantes que son los comentarios de código que expliquen cual es la utilidad de cada rutina y procedimiento, o que expliquen porque se realizan ciertas acciones que no parecen claras. Las partes que me ha tocado tocar y modificar han quedado detalladas de la mejor forma posible para que cualquier otra persona que tenga que realizar modificaciones en el futuro lo tenga algo más fácil.

Realizar este proyecto me ha servido para terminar de convencerme de que el mundo de la programación no es para mi. No me he sentido a gusto trabajando continuamente escribiendo código. Me he divertido mucho y he aprendido mucho más mientras realizaba el estudio y análisis de los problemas que suponía hacer cada mejora y realizar las planificaciones oportunas para cada una de ellas y poder ir superándolas.

7.2. Líneas Futuras de Desarrollo

Como ya se ha dicho en el apartado 4.8 esa mejora ha quedado sin realizar por falta de tiempo y estar cumplido de sobra el resto de objetivos. Este puede ser un buen comienzo para una posterior evolución del programa, si bien a nivel de uso no afectaría prácticamente.

Otra mejora interesante sería poder realizar el cambio de idioma del programa de forma que no sea dependiente de una dll externa. Como se ha planteado en el apartado 4.1 esto no es una tarea nada sencilla por la forma de programación que presenta el programa. Esto podría plantearse como un único proyecto. La envergadura y nivel de cambios a solucionar cubrirían correctamente los objetivos de un proyecto fin de carrera. Si se plantea un proyecto para realizar esta corrección, en mi opinión, tendría que ser para un alumno de Ingeniería Informática y no de Electrónica puesto que en este caso no se necesita comprender que es lo que realiza la aplicación sino el como esta programado.

De todas formas, todas las líneas futuras de desarrollo y mejora que pueda incluir yo como desarrollador y no usuario de la aplicación nunca serán muy objetivas. Las personas que tienen que definir el futuro de la aplicación, así como sus virtudes y carencias, tienen que ser los propios usuarios.

APÉNDICES

Apéndice A

Planificación

A.1. Diagrama de Gantt

A.2. Gantt de Seguimiento

A.3. Diagrama de Red

A.4. Calendario

Apéndice B

Código Fuente

B.1. NTGraph3D.dll

B.1.1. Visualización de Datos

```
////////////////////////////////////////////
// Clase que define las propiedades del objeto de representacion
// NTGraphCtl.h
////////////////////////////////////////////
class CElement
{
public:
  CElement()
  {
    min=max=CPoint3D(0,0,0);
    bIsPlotAvailable = FALSE ;
    m_bShow = TRUE;
    m_bFill = TRUE;
    m_bFlat = FALSE;
    m_bLights = FALSE;

    ////// Initial lighting params //////
    m_LightParam[0] = 40; // X position
    m_LightParam[1] = 100;  // Y position
    m_LightParam[2] = 80; // Z position
    m_LightParam[3] = 80; // Ambient light
    m_LightParam[4] = 50; // Diffuse light
    m_LightParam[5] = 50; // Specular light
    m_LightParam[6] = 50; // Ambient material
    m_LightParam[7] = 50; // Diffuse material
    m_LightParam[8] = 40; // Specular material
    m_LightParam[9] = 70; // Shininess material
    m_LightParam[10] = 50;  // Emission material

    m_LineColor = m_PointColor = RGB(100,100,100);

    m_SurfaceColor = RGB(255, 255, 255);

    m_nType  = LinePoint;
    m_LineWidth = 0.0f;
    m_PointSize = 0.0f;
  }

  BOOL bIsPlotAvailable;
  BOOL m_bShow;
  BOOL m_bFill;
```

```
    BOOL m_bFlat;
    BOOL m_bLights;
    int m_LightParam[11]; // Graphics dimension (along X-axis)
    COLORREF m_LineColor;
    COLORREF m_PointColor;
    LineType m_nType;
    GLfloat  m_LineWidth;
    GLfloat  m_PointSize;

    COLORREF m_SurfaceColor;

    CPoint3D min,max;

    POINTVECTOR m_PointList;
  };
```

```
  ////////////////////////////////////
  //Fichero NTGraphCtrl.cpp
  //Modificado a partir del original de Nikolai Teofilov
  ////////////////////////////////////
  //Definiciones generales

  #include "stdafx.h"
  #include "NTGraph3D.h"
  #include "GraphCtl.h"

  static const FONTDESC _fontdesc =
        {sizeof(FONTDESC), OLESTR("times␣new␣roman"), FONTSIZE( 14 ),
        FW_BOLD, ANSI_CHARSET, TRUE, FALSE, FALSE };

  ////////////////////////////////////////////////////////////////////////////
  // CGraphCtl
  unsigned char threeto8[8] =
  {
    0, 0111>>1, 0222>>1, 0333>>1, 0444>>1, 0555>>1, 0666>>1, 0377
  };

  unsigned char twoto8[4] =
  {
    0, 0x55, 0xaa, 0xff
  };

  unsigned char oneto8[2] =
  {
    0, 255
  };

  static int defaultOverride[13] =
  {
    0, 3, 24, 27, 64, 67, 88, 173, 181, 236, 247, 164, 91
  };

  static PALETTEENTRY defaultPalEntry[20] =
  {
    { 0,    0,    0,     0 },
    { 0x80,0,    0,     0 },
    { 0,    0x80,0,     0 },
    { 0x80,0x80,0,     0 },
    { 0,    0,    0x80, 0 },
    { 0x80,0,    0x80, 0 },
    { 0,    0x80,0x80, 0 },
    { 0xC0,0xC0,0xC0, 0 },

    { 192, 220, 192,  0 },
    { 166, 202, 240,  0 },
    { 255, 251, 240,  0 },
```

```
  { 160, 160, 164,  0 },

  { 0x80,0x80,0x80, 0 },
  { 0xFF,0,   0,    0 },
  { 0,   0xFF,0,    0 },
  { 0xFF,0xFF,0,    0 },
  { 0,   0,   0xFF, 0 },
  { 0xFF,0,   0xFF, 0 },
  { 0,   0xFF,0xFF, 0 },
  { 0xFF,0xFF,0xFF, 0 }
};


int logaritmica = 0;
double valMax = 0;
double valMin = 0;
```

```
void CGraphCtl::CreateContext(HDC hdc, RECT& rc)
{
  PIXELFORMATDESCRIPTOR pfd;
  GLfloat     fMaxObjSize, fAspect;
  GLfloat     fNearPlane, fFarPlane;

  if (!bSetupPixelFormat(hdc))
    return;

  CreateRGBPalette(hdc);

  ::SelectPalette(hdc, m_hPal, FALSE);
  ::RealizePalette(hdc);

  int n = ::GetPixelFormat(hdc);
  ::DescribePixelFormat(hdc, n, sizeof(pfd), &pfd);


  m_hrc = wglCreateContext(hdc);
  wglMakeCurrent(hdc, m_hrc);

  glClearDepth(10.0f);
  glEnable(GL_DEPTH_TEST);

  if (rc.bottom)
    fAspect = (GLfloat)rc.right/rc.bottom;
  else    // don't divide by zero, not that we should ever run into that...
    fAspect = 1.0f;

  fNearPlane = 3.0f;
  fFarPlane = 20.0f;
  fMaxObjSize = 3.0f;
  m_fRadius = fNearPlane + fMaxObjSize / 2.0f;

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();

  // update the camera
  if (m_nProjection == Orthographic)
    glOrtho(-1,1,-1,1,fNearPlane,fFarPlane);
  else
      gluPerspective(30.0f, fAspect, fNearPlane, fFarPlane);

  glMatrixMode(GL_MODELVIEW);
}
```

```
LRESULT CGraphCtl::OnCreate(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled)
{
  // Won't be here unless we just got activated and created a window
```

```
  HDC hdc = GetDC();
  RECT rc;
  GetClientRect(&rc);
  CreateContext(hdc, rc);
  return 0;
}
```

```
LRESULT CGraphCtl::OnDestroy(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled)
{

  ::wglMakeCurrent(NULL,  NULL);

  if (m_hrc)
  {
    ::wglDeleteContext(m_hrc);
    m_hrc = NULL;
  }
  return 0;
}
```

```
LRESULT CGraphCtl::OnLButtonDown(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled)
{

  SetCapture();
  m_bMouseCaptured = TRUE;
  m_xPos = (short)LOWORD(lParam);  // horizontal position of cursor
  m_yPos = (short)HIWORD(lParam);  // vertical position of cursor
  return 0;
}
```

```
LRESULT CGraphCtl::OnLButtonDblClk(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled)
{
  ShowPropertyPages();
  return 0;
}
```

```
LRESULT CGraphCtl::OnLButtonUP(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled)
{
  m_bMouseCaptured = FALSE;
  ReleaseCapture();
  FireViewChange();

  return 0;
}
```

```
LRESULT CGraphCtl::OnMouseMove(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled)
{
  if (m_bMouseCaptured)
  {
    int xPos = (short)LOWORD(lParam);  // horizontal position of cursor
    int yPos = (short)HIWORD(lParam);  // vertical position of cursor
    GLfloat fNearPlane = 3.0f;
    GLfloat fFarPlane = 7.0f;
    GLfloat fMaxObjSize = 3.0f;

    if (m_nTrackState == Zoom)
    {
      m_fRadius += (float)(m_xPos - xPos)/100.0f;
      FireViewChange();
    }
    else if (m_nTrackState == Rotate)
    {
      m_wAngleX -= (float)(m_yPos - yPos)/2.0f;
```

```
        m_wAngleY -= (float)(m_xPos - xPos)/2.0f;
        FireViewChange();
      }
      else if (m_nTrackState == Pan)
      {
        DoPan(xPos,yPos);
      }

      m_xPos = xPos;
      m_yPos = yPos;
    }
    return 0;
  }
```

```
  void CGraphCtl::DoPan(int x, int y)
  {
    double Y1 = y + dRangeY[MIN]/(dRangeY[MAX]-dRangeY[MIN]) + 0.5;
    double Y2 = m_yPos + dRangeY[MIN]/(dRangeY[MAX]-dRangeY[MIN]) + 0.5;
    double yOffset = (Y1 - Y2)/100.0;

    double X1 = x + dRangeX[MIN]/(dRangeX[MAX]-dRangeX[MIN]) + 0.5;
    double X2 = m_xPos + dRangeX[MIN]/(dRangeX[MAX]-dRangeX[MIN]) + 0.5;
    double xOffset = (X1 - X2)/100.0;


    SetRange(dRangeX[MIN] - xOffset, dRangeX[MAX] - xOffset,
         dRangeY[MIN] + yOffset, dRangeY[MAX] + yOffset,
         dRangeZ[MIN] + xOffset, dRangeZ[MAX] + xOffset, logaritmica);

  }
```

```
  HRESULT CGraphCtl::OnDraw(ATL_DRAWINFO& di)
  {
    HDC hdc = di.hdcDraw;
    RECT& rc = *(RECT*)di.prcBounds;
    DrawGraph(hdc, rc);

    return 1;
  }
```

```
  STDMETHODIMP CGraphCtl::CopyToClipboard()
  {

    HDC hdc = GetDC();

      // Get client geometry
    RECT rc;
    GetClientRect(&rc);
    LONG
       width = rc.right  - rc.left,
       height = rc.bottom - rc.top;

    int bitsPerPixel = ::GetDeviceCaps(hdc, BITSPIXEL);

    switch(bitsPerPixel)
    {
      case 8:
      case 24:
           width -= width % 4;
             break;
      case 16:
             width -= width % 2;
             break;
      case 32:
      default:
```

```
            break;
    }

    // Alloc pixel bytes
    int NbBytes = bitsPerPixel/8 * width * height;

    // Fill header
    BITMAPINFOHEADER header;
    header.biWidth = width;
    header.biHeight = height;
    header.biSizeImage = NbBytes;
    header.biSize = 40;
    header.biPlanes = 1;
    header.biBitCount =  3 * 8; // RGB
    header.biCompression = 0;
    header.biXPelsPerMeter = 0;
    header.biYPelsPerMeter = 0;
    header.biClrUsed = 0;
    header.biClrImportant = 0;

    // Generate handle
    HANDLE handle = (HANDLE)::GlobalAlloc (GHND,sizeof(BITMAPINFOHEADER) + NbBytes);
    if(handle != NULL)
    {
      // Lock handle
      char *pData = (char *) ::GlobalLock((HGLOBAL)handle);
      // Copy header and data
      memcpy(pData,&header,sizeof(BITMAPINFOHEADER));
      ::glReadPixels(0,0,width,height,GL_BGR_EXT,GL_UNSIGNED_BYTE,
              pData+sizeof(BITMAPINFOHEADER));
      // Unlock
      ::GlobalUnlock((HGLOBAL)handle);

      // Push DIB in clipboard
      OpenClipboard();
      EmptyClipboard();
      SetClipboardData(CF_DIB,handle);
      CloseClipboard();
    }

    ReleaseDC(hdc);
    return S_OK;

  }
```

```
STDMETHODIMP CGraphCtl::SaveBitMap(BSTR pszFile)
{
  HDC hdc = GetDC();

  HANDLE hf;
  DWORD dwTmp;

  // Get client geometry
  RECT rc;
  //de aqui sacamos el tama\~{n}o de la imagen
  GetClientRect(&rc);
  LONG
     width = rc.right  - rc.left,
     height = rc.bottom - rc.top;

  int bitsPerPixel = ::GetDeviceCaps(hdc, BITSPIXEL);

  //Obtenemos el numero de bits por pixel
  switch(bitsPerPixel)
  {
    case 8:
```

```
      case 24:
           width -= width % 4;
             break;
      case 16:
             width -= width % 2;
             break;
      case 32:
      default:
             break;
31  }
    // Alloc pixel bytes
    int NbBytes = bitsPerPixel/8 * width * height;

    // Fill header
    //BITMAPINFOHEADER informacion del la imagen
    BITMAPINFOHEADER header;
    header.biWidth = width;
    header.biHeight = height;
    header.biSizeImage = NbBytes;
41  header.biSize = 40;
    header.biPlanes = 1;
    header.biBitCount =  3 * 8; // RGB
    header.biCompression = 0;
    header.biXPelsPerMeter = 0;
    header.biYPelsPerMeter = 0;
    header.biClrUsed = 0;
    header.biClrImportant = 0;

    //BITMAPFILEHEADER Cabecera del archivo
51  BITMAPFILEHEADER hdr;
    hdr.bfType = 0x4d42;        // 0x42 = "B" 0x4d = "M"
    //Compute the size of the entire file.
    hdr.bfSize = (DWORD) (sizeof(BITMAPFILEHEADER) +
          header.biSize + header.biClrUsed * sizeof(RGBQUAD) +
          header.biSizeImage);

    hdr.bfReserved1 = 0;
    hdr.bfReserved2 = 0;

61  //Compute the offset to the array of color indices.
    hdr.bfOffBits = (DWORD) sizeof(BITMAPFILEHEADER) +
            header.biSize + header.biClrUsed * sizeof (RGBQUAD);


      // Generate handle
    HANDLE handle = (HANDLE)::GlobalAlloc (GHND,sizeof(BITMAPINFOHEADER) + NbBytes);
    if(handle != NULL)
    {
      // Lock handle
71    char *pData = (char *) ::GlobalLock((HGLOBAL)handle);
      // Copy header and data
      memcpy(pData,&header,sizeof(BITMAPINFOHEADER));
      ::glReadPixels(0,0,width,height,GL_BGR_EXT,GL_UNSIGNED_BYTE,
              pData+sizeof(BITMAPINFOHEADER));
      // Unlock
      ::GlobalUnlock((HGLOBAL)handle);


      hf = CreateFile((CString)pszFile,
81          GENERIC_READ | GENERIC_WRITE,
            (DWORD) 0,
            NULL,
            CREATE_ALWAYS,
            FILE_ATTRIBUTE_NORMAL,
            (HANDLE) NULL);
      //Copy the BITMAPFILEHEADER into the .BMP file.
```

```
      if(!WriteFile(hf, (LPVOID) &hdr, sizeof(BITMAPFILEHEADER), (LPDWORD) &dwTmp,  NULL))
      {
        CString resString;
        resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_FILE_HEADER + m_Idioma, NULL);
        MessageBox(resString, "", MB_OK|MB_ICONERROR);
        return S_OK;
      }

      if(!WriteFile(hf, (LPSTR) pData, (int) header.biSizeImage, (LPDWORD) &dwTmp,NULL))
      {
        CString resString;
        resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_WRITE_COLOR + m_Idioma, NULL);
        MessageBox(resString, "", MB_OK|MB_ICONERROR);

        return S_OK;
      }

      // Close the .BMP file.
      if (!CloseHandle(hf))
      {
        CString resString;
        resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_CLOSE_BITMAP + m_Idioma, NULL);
        MessageBox(resString, "", MB_OK|MB_ICONERROR);
        return S_OK;
      }
    }

    ReleaseDC(hdc);
    return S_OK;
  }
```

```
  void CGraphCtl::DrawBorder(HDC hdc, RECT rc)
  {

   if (m_bBorderVisible)
       switch(m_nBorderStyle)
       {
         case bump:
         if(m_nAppearance)
           ::DrawEdge(hdc, &rc, EDGE_BUMP|EDGE_SUNKEN, BF_RECT);
         else
           ::DrawEdge(hdc, &rc, EDGE_BUMP, BF_RECT);

         break;
         case etched:
         if(m_nAppearance)
           ::DrawEdge(hdc, &rc, EDGE_ETCHED|EDGE_SUNKEN, BF_RECT);
         else
           ::DrawEdge(hdc, &rc, EDGE_ETCHED, BF_RECT);
           break;
         case raised:
        if(m_nAppearance)
          ::DrawEdge(hdc, &rc, EDGE_RAISED|EDGE_SUNKEN, BF_RECT);
        else
          ::DrawEdge(hdc, &rc, EDGE_RAISED, BF_RECT);
          break;
         default:
             ::DrawEdge(hdc, &rc, EDGE_SUNKEN, BF_RECT);
        break;

       }
    else if(m_nAppearance && !m_bBorderVisible)
       ::DrawEdge(hdc, &rc, EDGE_SUNKEN, BF_RECT);

  }
```

```
void CGraphCtl::DrawGraph(HDC hdc, RECT rc)
{
  ::SelectPalette(hdc, m_hPal, FALSE);
  ::RealizePalette(hdc);

  wglMakeCurrent(hdc, m_hrc);

  COLORREF colBkgn;
  OleTranslateColor(m_clrBackColor, m_hPal, &colBkgn);
  SetTextColor(hdc, colBkgn);

    SetBkColor(colBkgn);

  glPushMatrix();

  glTranslatef(0.0f, 0.0f, -m_fRadius);
  glRotatef(m_wAngleX, 1.0f, 0.0f, 0.0f);
  glRotatef(m_wAngleY, 0.0f, 1.0f, 0.0f);
  glRotatef(m_wAngleZ, 0.0f, 0.0f, 1.0f);

  DrawAxisBox();
  DrawAxisGrid();
  PlotElement();
  DrawAxisLabels(hdc);

  glPopMatrix();

  glFinish();
  SwapBuffers(wglGetCurrentDC());

    DrawGraphTitle(hdc,rc);
  DrawBorder(hdc,rc);
}
```

```
void CGraphCtl::DrawGraphTitle(HDC hdc, RECT rc)
{
  COLORREF colT;
  OleTranslateColor(m_clrCaptionColor, m_hPal, &colT);
  SetTextColor(hdc, colT);
  SetBkMode(hdc, TRANSPARENT);
  USES_CONVERSION;
  LPCTSTR lpsz = OLE2T(m_bstrCaption);
  DrawText(hdc, lpsz, -1, &rc, DT_CENTER | DT_TOP | DT_SINGLELINE);
  SetTextColor(hdc,0);

  //Maximo y minimo
  char str[50];
  if (logaritmica)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MAXIMO + m_Idioma, NULL);
    sprintf(str,resString,pow(10.0,valMax));
  }
  else
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MAXIMO + m_Idioma, NULL);
    sprintf(str,resString,valMax);

  }
  DrawText(hdc, str, -1, &rc, DT_RIGHT | DT_TOP);
  if (logaritmica)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MINIMO + m_Idioma, NULL);
```

```
      sprintf(str,resString,pow(10.0,valMin));
    }
    else
    {
      CString resString;
      resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MINIMO + m_Idioma, NULL);
      sprintf(str,resString,valMin);

    }
    DrawText(hdc, str, -1, &rc, DT_LEFT | DT_TOP);
  }
```

```
  void CGraphCtl::DrawAxisBox()
  {
    COLORREF colX,colY,colZ;
    OleTranslateColor(m_clrXGridColor, m_hPal, &colX);
    OleTranslateColor(m_clrYGridColor, m_hPal, &colY);
    OleTranslateColor(m_clrZGridColor, m_hPal, &colZ);

    glLineWidth(1.0f);

    glBegin(GL_LINES );
      SetColor(colZ);
      glVertex3f(-0.5f, -0.5f, -0.5f);
      glVertex3f(-0.5f, -0.5f, 0.5f);
      glVertex3f(-0.5f, -0.5f, 0.5f);
      glVertex3f(0.5f, -0.5f, 0.5f);
      glVertex3f(0.5f, -0.5f, 0.5f);
      glVertex3f(0.5f, -0.5f, 0.5f);
      glVertex3f(0.5f, -0.5f, -0.5f);
      SetColor(colX);
    glEnd();
    glBegin(GL_LINES);
      glVertex3f(-0.5f, 0.5f, 0.5f);
      glVertex3f(-0.5f, -0.5f, 0.5f);

    glEnd();

  }
```

```
  void CGraphCtl::DrawAxisGrid()
  {

    glBegin(GL_LINES);

    float X, Y, Z;

    int i;

    COLORREF colX, colY, colZ;
    OleTranslateColor(m_clrXGridColor, m_hPal, &colX);
    OleTranslateColor(m_clrYGridColor, m_hPal, &colY);
    OleTranslateColor(m_clrZGridColor, m_hPal, &colZ);

  // Draw X Grid

    for ( i = 0 ; i <= m_nGridX ; i++)
    {
      X = ((float)i)/m_nGridX;
      //  No queremos rejilla en altura
      // x axis
      SetColor(colZ);

      glVertex3f(X-0.5f, -0.5f, -0.5f);
      glVertex3f(X-0.5f, -0.5f,  0.5f);
```

```
    }

  // Draw Y Grid
    float log;
    log = (float)1.0/m_nGridY;
    //No queremos rejilla en eje Y solo peque\~{n}as marcas
    for ( i = 0 ; i <= m_nGridY ; i++)
    {

        Y = ((float)i)/m_nGridY;

      // y axis
      SetColor(colX);
      //plano Z
      glVertex3f(-0.5f, Y-0.5f,  0.42f);
      glVertex3f(-0.5f, Y-0.5f,  0.5f);
      //plano X
      glVertex3f(-0.42f,  Y-0.5f,  0.5f);
      glVertex3f(-0.5f, Y-0.5f,  0.5f);
      if (logaritmica)
      {
        if ((Y!=1) && (dRangeY[MAX]-dRangeY[MIN]==m_nGridY))
        {
          SetColor(colX);
          //plano Z
          glVertex3f((GLfloat)-0.5f,  (GLfloat)(Y+(log*log10(3.0))-0.5f),
                      (GLfloat)0.47f);
          glVertex3f((GLfloat)-0.5f,  (GLfloat)(Y+(log*log10(3.0))-0.5f),
                      (GLfloat)0.5f);
          glVertex3f((GLfloat)-0.5f,  (GLfloat)(Y+(log*log10(5.0))-0.5f),
                      (GLfloat)0.47f);
          glVertex3f((GLfloat)-0.5f,  (GLfloat)(Y+(log*log10(5.0))-0.5f),
                      (GLfloat)0.5f);
          glVertex3f((GLfloat)-0.5f,  (GLfloat)(Y+(log*log10(7.0))-0.5f),
                      (GLfloat)0.47f);
          glVertex3f((GLfloat)-0.5f,  (GLfloat)(Y+(log*log10(7.0))-0.5f),
                      (GLfloat)0.5f);
          //plano X suprimido, no se aprecia nada
        }
      }
    }

  // Draw Z Grid
    for ( i = 0 ; i <= m_nGridZ ; i++)
    {
      Z = ((float)i)/m_nGridZ;

      // z axis
      SetColor(colZ);

      glVertex3f(-0.5f  ,-0.5f  ,Z-0.5f);
      glVertex3f( 0.5f  ,-0.5f  ,Z-0.5f);

      //No queremos rejilla en altura
    }
    glEnd();

  }
```

```
  void CGraphCtl::DrawAxisLabels(HDC hdc)

  {
    double X,Y,Z,res;
    int i;
    CPoint3D pt;
```

```
    double ExpMin;
    double parcial;
    int exponente;
    CString str;

    COLORREF colX, colY, colZ;
    OleTranslateColor(m_clrXGridColor, m_hPal, &colX);
    OleTranslateColor(m_clrYGridColor, m_hPal, &colY);
16  OleTranslateColor(m_clrZGridColor, m_hPal, &colZ);

    // Draw X Grid Label
    res = (dRangeX[MAX] - dRangeX[MIN]) / m_nGridX ;

    for ( i = 0 ; i <= m_nGridX ; i++)
    {
      X = dRangeX[MIN] + (res * (float)i);

      str=FormatAxisLabel(X,m_TiempoX, (CString)m_bstrFormatoX);
26
      X = ((float)i)/m_nGridX;

      SetColor(colX);

      pt=CPoint3D(-0.5,-0.5,0.5);
      pt.Translate(X,0.0,0.3);

      PrintText(hdc, pt, str);
    }
36  if (logaritmica == 0)
    {
    //dibujado lineal
      // Draw Y Grid Label
      res = (dRangeY[MAX] - dRangeY[MIN]) / m_nGridY ;

      for ( i = 0 ; i <= m_nGridY ; i++)
      {
        Y = dRangeY[MIN] + (res * (float)i) ;
        str=FormatAxisLabel(Y,m_TiempoY, (CString)m_bstrFormatoY);
46
        Y = ((float)i)/m_nGridY;

        SetColor(colY);

        pt = CPoint3D(-0.5,-0.5,0.5);
        pt.Translate(0.0,Y,0.3);
        PrintText(hdc, pt, str);
      }
    }
56  else
    {
      ExpMin = pow(10.0, floor(dRangeY[MIN]));
      res = (dRangeY[MAX] - dRangeY[MIN]) / m_nGridY ;

      for ( i = 0 ; i <= m_nGridY ; i++)
      {
        exponente =(int)( res * (float)i);
        parcial = pow(10.0, exponente);

66      Y = ExpMin * parcial;

        str=FormatAxisLabel(Y,m_TiempoY, (CString)m_bstrFormatoY);

        Y = ((float)i)/m_nGridY;

        // green y axis
        //glColor3f(0.f,1.f,0.f);
```

```
        SetColor(colY);

        pt = CPoint3D(-0.5,-0.5,0.5);
        pt.Translate(0.0,Y,0.3);
        PrintText(hdc, pt, str);
      }

    }

    // Draw Z Grid Label
    res = (dRangeZ[MAX] - dRangeZ[MIN]) / m_nGridZ ;

    for ( i = 0 ; i <= m_nGridZ ; i++)
    {
      Z = dRangeZ[MIN] + (res * (float)i) ;

      str=FormatAxisLabel(Z,m_TiempoZ, (CString)m_bstrFormatoZ);
      Z = ((float)i)/m_nGridZ;

      SetColor(colZ);

      pt = CPoint3D(0.5,-0.5,-0.5);
      pt.Translate (0.2,0.0,Z);
      PrintText(hdc, pt, str);

    }

    // Draw Axis Titles
      SetColor(RGB(0,0,0));

    // Native C++ compiler COM support
    // BSTR, VARIANT wrappers header
    using namespace _com_util;

    SetColor(colX);
    PrintText(hdc,CPoint3D(0.0f,-0.8f,0.7f),
      ConvertBSTRToString(m_bstrXLabel));
    SetColor(colY);
    PrintText(hdc,CPoint3D(-0.5f,0.6f,0.6f),
      ConvertBSTRToString(m_bstrYLabel));
    SetColor(colZ);
    PrintText(hdc,CPoint3D(0.7f,-0.8f,-0.0f),
      ConvertBSTRToString(m_bstrZLabel));
    glEnd();
  }
```

```
  BOOL CGraphCtl::bSetupPixelFormat(HDC hdc)
  {
    static PIXELFORMATDESCRIPTOR pfd =
    {
      sizeof(PIXELFORMATDESCRIPTOR),  // size of this pfd
      1,                              // version number
      PFD_DRAW_TO_WINDOW |            // support window
        PFD_SUPPORT_OPENGL |          // support OpenGL
        PFD_DOUBLEBUFFER,             // double buffered
      PFD_TYPE_RGBA,                  // RGBA type
      24,                             // 24-bit color depth
      0, 0, 0, 0, 0, 0,               // color bits ignored
      0,                              // no alpha buffer
      0,                              // shift bit ignored
      0,                              // no accumulation buffer
      0, 0, 0, 0,                     // accum bits ignored
      32,                             // 32-bit z-buffer
      0,                              // no stencil buffer
      0,                              // no auxiliary buffer
      PFD_MAIN_PLANE,                 // main layer
```

```
    0,                                  // reserved
    0, 0, 0                             // layer masks ignored
  };
  int pixelformat;

  if ( (pixelformat = ChoosePixelFormat(hdc, &pfd)) == 0 )
  {
    ATLASSERT(FALSE);
    return FALSE;
  }

  if (SetPixelFormat(hdc, pixelformat, &pfd) == FALSE)
  {
    ATLASSERT(FALSE);
    return FALSE;
  }

  return TRUE;
}
```

```
void CGraphCtl::PrintText(HDC hdc, CPoint3D pt, const char *str)
{
  HFONT hOldFont = NULL;
  if (m_pFont)
  {
    CComPtr<IFont> pFont;
    m_pFont->QueryInterface(IID_IFont, (void**)&pFont);
    HFONT hfont;
    pFont->get_hFont(&hfont);
    hOldFont = (HFONT) SelectObject(hdc, hfont);
  }
  else
    SelectObject (hdc, GetStockObject (SYSTEM_FONT));

  // create bitmaps for the device context font's first 256 glyphs
  wglUseFontBitmaps(hdc, 0, 256, 1000);

  // move to position
  glRasterPos3f(pt.x,pt.y,pt.z);

  // set up for a string-drawing display list call
  glListBase(1000);

  // draw a string using font display lists
  glCallLists(strlen(str), GL_UNSIGNED_BYTE, (GLubyte*)str);

  // get all those commands to execute
  glFlush();

  // delete our 256 glyph display lists
  glDeleteLists(1000, 256) ;

  if (hOldFont)
    SelectObject(hdc, hOldFont);
}
```

```
unsigned char CGraphCtl::ComponentFromIndex(int i, UINT nbits, UINT shift)
{
  unsigned char val;

  val = (unsigned char) (i >> shift);
  switch (nbits)
  {

  case 1:
    val &= 0x1;
```

```
      return oneto8[val];
    case 2:
      val &= 0x3;
      return twoto8[val];
    case 3:
      val &= 0x7;
      return threeto8[val];

    default:
      return 0;
    }
  }
```

```
  void CGraphCtl::CreateRGBPalette(HDC hdc)
  {
    PIXELFORMATDESCRIPTOR pfd;
    int n, i;

    if (m_pPal)
      return;

    n = ::GetPixelFormat(hdc);
    ::DescribePixelFormat(hdc, n, sizeof(pfd), &pfd);

    if (pfd.dwFlags & PFD_NEED_PALETTE)
    {
      n = 1 << pfd.cColorBits;
      m_pPal = (PLOGPALETTE) new char[sizeof(LOGPALETTE) + n * sizeof(PALETTEENTRY)];

      ATLASSERT(m_pPal != NULL);

      m_pPal->palVersion = 0x300;
      m_pPal->palNumEntries = n;
      for (i=0; i<n; i++)
      {
        m_pPal->palPalEntry[i].peRed =
            ComponentFromIndex(i, pfd.cRedBits, pfd.cRedShift);
        m_pPal->palPalEntry[i].peGreen =
            ComponentFromIndex(i, pfd.cGreenBits, pfd.cGreenShift);
        m_pPal->palPalEntry[i].peBlue =
            ComponentFromIndex(i, pfd.cBlueBits, pfd.cBlueShift);
        m_pPal->palPalEntry[i].peFlags = 0;
      }

      /* fix up the palette to include the default GDI palette */
      if ((pfd.cColorBits == 8)                           &&
        (pfd.cRedBits   == 3) && (pfd.cRedShift   == 0) &&
        (pfd.cGreenBits == 3) && (pfd.cGreenShift == 3) &&
        (pfd.cBlueBits  == 2) && (pfd.cBlueShift  == 6)
         )
      {
        for (i = 1 ; i <= 12 ; i++)
          m_pPal->palPalEntry[defaultOverride[i]] = defaultPalEntry[i];
      }

      m_hPal = ::CreatePalette((LPLOGPALETTE)m_pPal);
  //        delete pPal;

      ::SelectPalette(hdc, m_hPal, FALSE);
      ::RealizePalette(hdc);
    }
  }
```

```
STDMETHODIMP CGraphCtl::get_CaptionColor(OLE_COLOR *pVal)
{
  *pVal = m_clrCaptionColor;
```

```
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_CaptionColor(OLE_COLOR newVal)
{
  m_clrCaptionColor = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::SetRange(double xmin, double xmax, double ymin, double ymax,
    double zmin, double zmax, int esclog)
{
  logaritmica=esclog;
  if (xmin==xmax || ymin==ymax || zmin==zmax)
    return S_OK;
  else if (xmin>xmax || ymin>ymax || zmin>zmax)
    return S_OK;
  else
  {
    dRangeX[MIN]=xmin;
    dRangeX[MAX]=xmax;
    dRangeZ[MIN]=zmin;
    dRangeZ[MAX]=zmax;
    dRangeY[MIN]=ymin;
    dRangeY[MAX]=ymax;
    }

  FireViewChange(); // call the InvalidateRect API directly!
  SetDirty(TRUE);
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::AutoRange()
{
  if (!bIsPlotAvailable) {

  SetRange(0,1,0,1,0,1, logaritmica);
  return S_OK;

  }
  SetRange(dAutoRangeX[MIN],dAutoRangeX[MAX],
         dAutoRangeY[MIN],dAutoRangeY[MAX],
       dAutoRangeZ[MIN],dAutoRangeZ[MAX], logaritmica);

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::ShowPropertyPages()
{
  DoVerbProperties(NULL,NULL);

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_TrackMode(short *pVal)
{
  *pVal = m_nTrackState;
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_TrackMode(short newVal)
{
   if (newVal >= 0 && newVal < 4)
   {
      m_nTrackState = newVal;
      SetDirty(TRUE);

      return S_OK;
   }
   else
   {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_TRACK_STATE + m_Idioma, NULL);
    return Error(_T(resString));
   }


  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_Projection(short newVal)
{
  newVal = newVal < 0 ? 0: newVal;
  newVal = newVal > 1 ? 1: newVal;

  m_nProjection = newVal;
  SetDirty(TRUE);

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();

  // update the camera
  if (m_nProjection == Orthographic)
    glOrtho(-1,1,-1,1,3.0f,20.0f);
  else
      gluPerspective(30.0f, 1.0f, 3.0f, 20.0f);

  glMatrixMode(GL_MODELVIEW);

  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_Projection(short *pVal)
{
  *pVal =(short) m_nProjection;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_XLabel(BSTR *pVal)
{
  ATLTRACE(_T("IGraph3D::get_XLabel\n"));
  *pVal = m_bstrXLabel.Copy();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_XLabel(BSTR newVal)
{
  USES_CONVERSION;
  ATLTRACE(_T("IGraph3D::put_XLabel\n"));
```

```
  m_bstrXLabel = newVal;
  SetDirty(TRUE);
  FireViewChange();


  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_YLabel(BSTR *pVal)
{
  ATLTRACE(_T("IGraph3D::get_YLabel\n"));
  *pVal = m_bstrYLabel.Copy();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_YLabel(BSTR newVal)
{
  USES_CONVERSION;
  ATLTRACE(_T("IGraph3D::put_YLabel\n"));
  m_bstrYLabel = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ZLabel(BSTR *pVal)
{
  ATLTRACE(_T("IGraph3D::get_ZLabel\n"));
  *pVal = m_bstrZLabel.Copy();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ZLabel(BSTR newVal)
{
  USES_CONVERSION;
  ATLTRACE(_T("IGraph3D::put_ZLabel\n"));
  m_bstrZLabel = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_XGridNumber(short *pVal)
{
    *pVal = m_nGridX;
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_XGridNumber(short newVal)
{
  newVal = newVal < 0 ? 1: newVal;
  newVal = newVal > 10 ? 10: newVal;

  m_nGridX = newVal;
  SetDirty(TRUE);

  FireViewChange(); // call the InvalidateRect API directly!

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_YGridNumber(short *pVal)
{
    *pVal = m_nGridY;
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_YGridNumber(short newVal)
{
  newVal = newVal < 0 ? 1: newVal;
  newVal = newVal > 10 ? 10: newVal;

  m_nGridY = newVal;
  SetDirty(TRUE);

  FireViewChange(); // call the InvalidateRect API directly!

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ZGridNumber(short *pVal)
{
    *pVal = m_nGridZ;
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ZGridNumber(short newVal)
{
  newVal = newVal < 0 ? 1: newVal;
  newVal = newVal > 10 ? 10: newVal;

  m_nGridZ = newVal;
  SetDirty(TRUE);

  FireViewChange(); // call the InvalidateRect API directly!

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_XGridColor(OLE_COLOR *pVal)
{
  *pVal = m_clrXGridColor;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_XGridColor(OLE_COLOR newVal)
{
  m_clrXGridColor = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_YGridColor(OLE_COLOR *pVal)
{
  *pVal = m_clrYGridColor;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_YGridColor(OLE_COLOR newVal)
{
  m_clrYGridColor = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ZGridColor(OLE_COLOR *pVal)
{
  *pVal = m_clrZGridColor;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ZGridColor(OLE_COLOR newVal)
{
  m_clrZGridColor = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
void CGraphCtl::PlotElement()
{

  ELEMENTVECTOR::iterator theElement;

  for (theElement = m_ElementList.begin();
    theElement != m_ElementList.end(); theElement++)
  {

    // Prevent plotting of non-existing data
    if (! theElement->bIsPlotAvailable )
       continue;

    ////////////////////////////////
    // Check show state of the element
    if (! theElement->m_bShow )
      continue;

    if (theElement->m_bLights)
    {
          SetLight2(theElement);
      glEnable(GL_LIGHTING);  // Lighting will be used
      glEnable(GL_LIGHT0);    // Only one (first) source of light
      glEnable(GL_DEPTH_TEST);// The depth of the Z-buffer will be taken into account
      glEnable(GL_COLOR_MATERIAL);// Material colors will be taken into account

    }

    POINTVECTOR::iterator aPoint;

        // Draw Lines
    if (theElement->m_nType == Lines || theElement->m_nType == LinePoint)
    {
      glLineWidth(theElement->m_LineWidth);
      glBegin(GL_LINE_STRIP);

      SetColor(theElement->m_LineColor);

      aPoint = theElement->m_PointList.begin();
```

```
      int i,npt = (int)sqrt(float(theElement->m_PointList.size()));
      for (aPoint, i=0; aPoint != theElement->m_PointList.end(); aPoint++, i++)
42    {
        if (i%npt==0)
        {
          glEnd();
          glBegin(GL_LINE_STRIP);
        }
        if (aPoint->x>=dRangeX[MIN] && aPoint->x<=dRangeX[MAX] &&
          aPoint->y>=dRangeY[MIN] && aPoint->y<=dRangeY[MAX] &&
          aPoint->z>=dRangeZ[MIN] && aPoint->z<=dRangeZ[MAX])
        {
52        float xF=(float)(dRangeX[MAX]-dRangeX[MIN]);
          float yF=(float)(dRangeY[MAX]-dRangeY[MIN]);
          float zF=(float)(dRangeZ[MAX]-dRangeZ[MIN]);

          CPoint3D pt;
          pt.x = (float)((aPoint->x-dRangeX[MIN])/xF - 0.5f);
          pt.y = (float)((aPoint->y-dRangeY[MIN])/yF - 0.5f);
          pt.z = (float)((aPoint->z-dRangeZ[MIN])/zF - 0.5f);

          glVertex3f(pt.x,pt.y,pt.z);
62      }
      }
      glEnd();
    }

    // Now draw points.
    if ( theElement->m_nType == Points || theElement->m_nType == LinePoint )
       {
      glPointSize(theElement->m_PointSize);
      glBegin(GL_POINTS);
72    SetColor(theElement->m_PointColor);

      aPoint = theElement->m_PointList.begin();
      for (aPoint; aPoint != theElement->m_PointList.end(); aPoint++)
      {
        if (aPoint->x>=dRangeX[MIN] && aPoint->x<=dRangeX[MAX] &&
          aPoint->y>=dRangeY[MIN] && aPoint->y<=dRangeY[MAX] &&
          aPoint->z>=dRangeZ[MIN] && aPoint->z<=dRangeZ[MAX])
        {
          float xF=(float)(dRangeX[MAX]-dRangeX[MIN]);
82        float yF=(float)(dRangeY[MAX]-dRangeY[MIN]);
          float zF=(float)(dRangeZ[MAX]-dRangeZ[MIN]);

          CPoint3D pt;
          pt.x = (float)((aPoint->x-dRangeX[MIN])/xF - 0.5f);
          pt.y = (float)((aPoint->y-dRangeY[MIN])/yF - 0.5f);
          pt.z = (float)((aPoint->z-dRangeZ[MIN])/zF - 0.5f);

          glVertex3f(pt.x,pt.y,pt.z);
        }
92    }
      glEnd();
    }
    //Draw surfaces and lines
    if ( theElement->m_nType == Surface )
    {
      glPolygonMode(GL_FRONT_AND_BACK,GL_FILL);

      if (theElement->m_bFlat)
        glShadeModel(GL_FLAT);
102   else
        glShadeModel(GL_SMOOTH);
      // Turn on the primitive connection mode (not connected)
```

```
glBegin (GL_QUADS);
  float r=float(GetRValue(theElement->m_SurfaceColor))/255;
  float g=float(GetGValue(theElement->m_SurfaceColor))/255;
  float b=float(GetBValue(theElement->m_SurfaceColor))/255;
  float rl=float(GetRValue(theElement->m_LineColor))/255;
  float gl=float(GetGValue(theElement->m_LineColor))/255;
  float bl=float(GetBValue(theElement->m_LineColor))/255;

int i, npt = (int)sqrt(float(theElement->m_PointList.size()));
int j,k,n;
int noFin = 1;
aPoint = theElement->m_PointList.begin();
for (aPoint, i=0; aPoint != theElement->m_PointList.end(); aPoint++,i++)
{
  noFin=1;
  if (i%npt==49)
  {
                // Si son los dos ultimos puntos
    //nos los saltamos para  que no los una con
    //los dos primeros de la siguiente linea
    noFin=0;
  }

  if (noFin)
  {
    // i, j, k, n   4 indices of a quad
    // Counter Clockwise direction

      j = i + npt; // Other vertices indices
      k = j+1;   //npt=50;
      n = i+1;

      //Para que no se nos salga del rango de los valores
      //j va 50 por delante de i
      if (k>=float(theElement->m_PointList.size()))
        break;
    // Get coordinates of 4 vertices
    CPoint3D
      iPt = Corrdinate(&(theElement->m_PointList[i])),
      jPt = Corrdinate(&(theElement->m_PointList[j])),
      kPt = Corrdinate(&(theElement->m_PointList[k])),
      nPt = Corrdinate(&(theElement->m_PointList[n]));

    float
      az = iPt.z-nPt.z,
      ay = iPt.y-nPt.y,

      by = jPt.y-iPt.y,
      bx = jPt.x-iPt.x,

      // Normal vector coordinates
      vx = ay*az,
      vy = -az*bx,
      vz = bx*by,

      // Normal vector length
      v  = float(sqrt(vx*vx + vy*vy + vz*vz));

    // Scale to unity
    vx /= v;
    vy /= v;
    vz /= v;

    // Set the normal vector
    glNormal3f (vx,vy,vz);
```

```
172         bool
              iEnRango=PtInAxisBox(&iPt),
              jEnRango=PtInAxisBox(&jPt),
              kEnRango=PtInAxisBox(&kPt),
              nEnRango=PtInAxisBox(&nPt);


            glColor3f (r,g,b);            // color puro

            if (iEnRango || jEnRango ||
182           kEnRango || nEnRango)
            {
              if (!iEnRango)
                iPt.y=-0.5;
              if (abs(iPt.x)<=0.5 && abs(iPt.z)<=0.5)
                glVertex3f (iPt.x, iPt.y, iPt.z);

              if (!jEnRango)
                jPt.y=-0.5;
              if(abs(jPt.x)<=0.5 && abs(jPt.z)<=0.5)
192             glVertex3f (jPt.x, jPt.y, jPt.z);

              if (!kEnRango)
                kPt.y=-0.5;
              if(abs(kPt.x)<=0.5 && abs(kPt.z)<=0.5)
                glVertex3f (kPt.x, kPt.y, kPt.z);

              if (!nEnRango)
                nPt.y=-0.5;
              if(abs(nPt.x)<=0.5 && abs(nPt.z)<=0.5)
202             glVertex3f (nPt.x, nPt.y, nPt.z);
            }

            glEnd();

            glLineWidth(theElement->m_LineWidth);
            glBegin(GL_LINE_LOOP);

              glColor3f (rl, gl, bl);
              if (iEnRango || jEnRango ||
212             kEnRango || nEnRango)
              {
                if (!iEnRango)
                  iPt.y=-0.5;
                if (abs(iPt.x)<=0.5 && abs(iPt.z)<=0.5)
                  glVertex3f (iPt.x, iPt.y, iPt.z);

                if (!jEnRango)
                  jPt.y=-0.5;
                if(abs(jPt.x)<=0.5 && abs(jPt.z)<=0.5)
222               glVertex3f (jPt.x, jPt.y, jPt.z);

                if (!kEnRango)
                  kPt.y=-0.5;
                if(abs(kPt.x)<=0.5 && abs(kPt.z)<=0.5)
                  glVertex3f (kPt.x, kPt.y, kPt.z);

                if (!nEnRango)
                  nPt.y=-0.5;
                if(abs(nPt.x)<=0.5 && abs(nPt.z)<=0.5)
232               glVertex3f (nPt.x, nPt.y, nPt.z);
              }

            glEnd();
            glBegin (GL_QUADS);
          }
```

```
        }
        glEnd();
      }
      if(glIsEnabled(GL_LIGHTING))
            glDisable(GL_LIGHTING);

  }
}
```

```
BOOL CGraphCtl::PtInRange(CPoint3D *pt)
{
  if (pt->x>=dRangeX[MIN] && pt->x<=dRangeX[MAX] &&
    pt->y>=dRangeY[MIN] && pt->y<=dRangeY[MAX] &&
    pt->z>=dRangeZ[MIN] && pt->z<=dRangeZ[MAX])
    return TRUE;
  return FALSE;
}
```

```
BOOL CGraphCtl::PtInAxisBox(CPoint3D* pt)
{
  float xF=(float)(dRangeX[MAX]-dRangeX[MIN]);
  float yF=(float)(dRangeY[MAX]-dRangeY[MIN]);
  float zF=(float)(dRangeZ[MAX]-dRangeZ[MIN]);

  CPoint3D point;

  point.x = (float)((pt->x + 0.5f) * xF + dRangeX[MIN]) ;
  point.y = (float)((pt->y + 0.5f) * yF + dRangeY[MIN]) ;
  point.z = (float)((pt->z + 0.5f) * zF + dRangeZ[MIN]) ;


  return (PtInRange(&point));
}
```

```
void CGraphCtl::SetLight(CElement *pElement)
{
  //====== Both surface sides are considered when calculating
  //====== each pixel color with the lighting formula

  glLightModeli(GL_LIGHT_MODEL_TWO_SIDE,1);

  //====== Light source position depends on the object sizes scaled to (0,100)

  float fPos[] =
  {
    ((pElement->m_LightParam[0])-50)*m_fRadius/100,
    ((pElement->m_LightParam[1])-50)*m_fRadius/100,
    ((pElement->m_LightParam[2])-50)*m_fRadius/100,
    1.f
  };
  glLightfv(GL_LIGHT0, GL_POSITION, fPos);

  //====== Ambient light intensity
  float f = (pElement->m_LightParam[3])/100.f;
  float fAmbient[4] = { f, f, f, 0.f };
  glLightfv(GL_LIGHT0, GL_AMBIENT, fAmbient);

  //====== Diffuse light intensity
  f = (pElement->m_LightParam[4])/100.f;
  float fDiffuse[4] = { f, f, f, 0.f };
  glLightfv(GL_LIGHT0, GL_DIFFUSE, fDiffuse);

  //====== Specular light intensity
  f = (pElement->m_LightParam[5])/100.f;
  float fSpecular[4] = { f, f, f, 0.f };
```

```
  glLightfv(GL_LIGHT0, GL_SPECULAR, fSpecular);

  //====== Surface material reflection properties for each light component
  f = (pElement->m_LightParam[6])/100.f;
  float fAmbMat[4] = { f, f, f, 0.f };
  glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, fAmbMat);

  f = (pElement->m_LightParam[7])/100.f;
  float fDifMat[4] = { f, f, f, 1.f };
  glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, fDifMat);

  f = (pElement->m_LightParam[8])/100.f;
  float fSpecMat[4] = { f, f, f, 0.f };
  glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, fSpecMat);

  //====== Material shininess
  float fShine = 128 * (pElement->m_LightParam[9])/100.f;
  glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, fShine);

  //====== Material light emission property
  f = (pElement->m_LightParam[10])/100.f;
  float fEmission[4] = { f, f, f, 0.f };
  glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, fEmission);
}
```

```
void CGraphCtl::SetLight2(ELEMENTVECTOR::iterator pElement)
{
  //====== Both surface sides are considered when calculating
  //====== each pixel color with the lighting formula

  glLightModeli(GL_LIGHT_MODEL_TWO_SIDE,1);

  //====== Light source position depends on the object sizes scaled to (0,100)

  float fPos[] =
  {
    ((pElement->m_LightParam[0])-50)*m_fRadius/100,
    ((pElement->m_LightParam[1])-50)*m_fRadius/100,
    ((pElement->m_LightParam[2])-50)*m_fRadius/100,
    1.f
  };
  glLightfv(GL_LIGHT0, GL_POSITION, fPos);

  //====== Ambient light intensity
  float f = (pElement->m_LightParam[3])/100.f;
  float fAmbient[4] = { f, f, f, 0.f };
  glLightfv(GL_LIGHT0, GL_AMBIENT, fAmbient);

  //====== Diffuse light intensity
  f = (pElement->m_LightParam[4])/100.f;
  float fDiffuse[4] = { f, f, f, 0.f };
  glLightfv(GL_LIGHT0, GL_DIFFUSE, fDiffuse);

  //====== Specular light intensity
  f = (pElement->m_LightParam[5])/100.f;
  float fSpecular[4] = { f, f, f, 0.f };
  glLightfv(GL_LIGHT0, GL_SPECULAR, fSpecular);

  //====== Surface material reflection properties for each light component
  f = (pElement->m_LightParam[6])/100.f;
  float fAmbMat[4] = { f, f, f, 0.f };
  glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, fAmbMat);

  f = (pElement->m_LightParam[7])/100.f;
  float fDifMat[4] = { f, f, f, 1.f };
  glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, fDifMat);
```

```
  f = (pElement->m_LightParam[8])/100.f;
  float fSpecMat[4] = { f, f, f, 0.f };
  glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, fSpecMat);

  //====== Material shininess
  float fShine = 128 * (pElement->m_LightParam[9])/100.f;
  glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, fShine);

  //====== Material light emission property
  f = (pElement->m_LightParam[10])/100.f;
  float fEmission[4] = { f, f, f, 0.f };
  glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, fEmission);
}
```

```
CPoint3D CGraphCtl::Corrdinate(CPoint3D *point)
{
  float xF=(float)(dRangeX[MAX]-dRangeX[MIN]);
  float yF=(float)(dRangeY[MAX]-dRangeY[MIN]);
  float zF=(float)(dRangeZ[MAX]-dRangeZ[MIN]);

  CPoint3D result;

  result.x = (float)((point->x-dRangeX[MIN])/xF - 0.5f);
  result.y = (float)((point->y-dRangeY[MIN])/yF - 0.5f);
  result.z = (float)((point->z-dRangeZ[MIN])/zF - 0.5f);

  return result;
}
```

```
STDMETHODIMP CGraphCtl::AddElement()
{
  valMax = -1e50;
  valMin =  1e50;
  CElement Element ;
  m_ElementList.push_back(Element);

  SetDirty(TRUE);
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::DeleteElement(short ElementID)
{
  if (m_ElementList.empty() || m_ElementList.size() < ElementID - 1)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),
            IDS_DELETE_ELEMENT + m_Idioma, NULL);
    return Error (_T(resString));
  }
    else
    m_ElementList.erase (m_ElementList.begin() + ElementID - 1);

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::ClearGraph()
{
  if (m_ElementList.empty())
    return S_OK;
    else
    m_ElementList.clear ();

  bIsPlotAvailable = FALSE;
```

```
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::PlotXYZ(double x, double y, double z, short ElementID)
{
  if (m_ElementList.empty())
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_PLOTXYZ + m_Idioma, NULL);
    return Error (_T(resString));
    //return Error(_T("You didn't say please first."));
  }

  if(bIsPlotAvailable)
  {
    if(x<dAutoRangeX[MIN])  dAutoRangeX[MIN]=floor(x);
    if(y<dAutoRangeY[MIN])  dAutoRangeY[MIN]=floor(y);
    if(z<dAutoRangeZ[MIN])  dAutoRangeZ[MIN]=floor(z);

    if(x>dAutoRangeX[MAX])  dAutoRangeX[MAX]=ceil(x);
    if(y>dAutoRangeY[MAX])  dAutoRangeY[MAX]=ceil(y);
    if(z>dAutoRangeZ[MAX])  dAutoRangeZ[MAX]=ceil(z);

  }
  else
  {

        dAutoRangeX[MIN] = floor(x);
        dAutoRangeY[MIN] = floor(y);
        dAutoRangeZ[MIN] = floor(z);

    dAutoRangeX[MAX] = ceil(x);
    dAutoRangeY[MAX] = ceil(y);
        dAutoRangeZ[MAX] = ceil(z);

    bIsPlotAvailable= TRUE ;
    }

  if (y>valMax)
    valMax = y;
  if (y<valMin)
    valMin = y;
  CPoint3D point((float)x,(float)y,(float)z);

  // Gets the position of the element by index.

  m_ElementList[ElementID].m_PointList.push_back(point);

  if(m_ElementList[ElementID].min.x > point.x)
       m_ElementList[ElementID].min.x=point.x ;

  if(m_ElementList[ElementID].min.y > point.y)
       m_ElementList[ElementID].min.y=point.y ;

  if(m_ElementList[ElementID].min.z > point.z)
       m_ElementList[ElementID].min.z=point.z ;

    if(m_ElementList[ElementID].max.x < point.x)
       m_ElementList[ElementID].max.x=point.x ;

  if(m_ElementList[ElementID].max.y < point.y)
       m_ElementList[ElementID].max.y=point.y ;

  if(m_ElementList[ElementID].max.z < point.z)
       m_ElementList[ElementID].max.z=point.z ;
```

```
  m_ElementList[ElementID].bIsPlotAvailable = TRUE ;

  FireViewChange(); // call the InvalidateRect API directly!

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementLineColor(short ElementID, OLE_COLOR *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
    //return Error(_T("Yo ho ho and a bottle of rum for that!"));
  }

  *pVal = m_ElementList[ElementID].m_LineColor;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementLineColor(short ElementID, OLE_COLOR newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LineColor = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_AnguloX(short ElementID, float *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }

  *pVal = (float) m_wAngleX;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_AnguloX(short ElementID, float newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_wAngleX = newVal;
  SetDirty(TRUE);
  FireViewChange();
```

```
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_AnguloY(short ElementID, float *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }

  *pVal = (float) m_wAngleY;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_AnguloY(short ElementID, float newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_wAngleY = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::ElementSurfaceColor(short ElementID, OLE_COLOR newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_SurfaceColor = newVal;

  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementPointColor(short ElementID, OLE_COLOR *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_PointColor;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementPointColor(short ElementID, OLE_COLOR newVal)
{
```

```
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_PointColor = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementLineWidth(short ElementID, float *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_LineWidth;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementLineWidth(short ElementID, float newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LineWidth = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementPointSize(short ElementID, float *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_PointSize;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementPointSize(short ElementID, float newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_PointSize = newVal;
  SetDirty(TRUE);
```

```
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementType(short ElementID, short *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_nType;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementType(short ElementID, short newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_nType = (LineType)newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementShow(short ElementID, BOOL *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_bShow;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementShow(short ElementID, BOOL newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_bShow = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementSurfaceFill(short ElementID, BOOL *pVal)
{
```

```
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_bFill;

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementSurfaceFill(short ElementID, BOOL newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_bFill = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementSurfaceFlat(short ElementID, BOOL *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_bFlat;

  return S_OK;
}

STDMETHODIMP CGraphCtl::put_ElementSurfaceFlat(short ElementID, BOOL newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_bFlat = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementLight(short ElementID, BOOL *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_bLights;

  return S_OK;
```

```
}
```

```
STDMETHODIMP CGraphCtl::put_ElementLight(short ElementID, BOOL newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_bLights = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementLightingAmbient(short ElementID, short *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_LightParam[3];

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementLightingAmbient(short ElementID, short newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LightParam[3] = newVal;
  SetDirty(TRUE);
  FireViewChange();


  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementLightingDiffuse(short ElementID, short *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_LightParam[4];

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementLightingDiffuse(short ElementID, short newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
```

```
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LightParam[4] = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementLightingSpecular(short ElementID, short *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_LightParam[5];

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementLightingSpecular(short ElementID, short newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LightParam[5] = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementMaterialAmbient(short ElementID, short *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_LightParam[6];

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementMaterialAmbient(short ElementID, short newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LightParam[7] = newVal;
  SetDirty(TRUE);
  FireViewChange();
```

```
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementMaterialDiffuse(short ElementID, short *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_LightParam[7];

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementMaterialDiffuse(short ElementID, short newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LightParam[7] = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementMaterialSpecular(short ElementID, short *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_LightParam[8];

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementMaterialSpecular(short ElementID, short newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LightParam[8] = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementMaterialShinines(short ElementID, short *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
```

```
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_LightParam[9];

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementMaterialShinines(short ElementID, short newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LightParam[9] = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_ElementMaterialEmission(short ElementID, short *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_LightParam[10];

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_ElementMaterialEmission(short ElementID, short newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LightParam[10] = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::SetLightCoordinates(short ElementID, float x, float y, float z)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }
  m_ElementList[ElementID].m_LightParam[0] = (int)x;
  m_ElementList[ElementID].m_LightParam[1] = (int)y;
  m_ElementList[ElementID].m_LightParam[2] = (int)z;
```

```
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::get_Lighting(short ElementID, BOOL *pVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_BOTTLE + m_Idioma, NULL);
    return Error (_T(resString));
  }
  *pVal = m_ElementList[ElementID].m_bLights;
  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::put_Lighting(short ElementID, BOOL newVal)
{
  if (m_ElementList.empty() || m_ElementList.size() - 1 < ElementID)
  {
    CString resString;
    resString.LoadString(::GetModuleHandle("NTGraph3D"),IDS_MOTHER + m_Idioma, NULL);
    return Error (_T(resString));
  }

  m_ElementList[ElementID].m_bLights = newVal;
  SetDirty(TRUE);
  FireViewChange();

  return S_OK;
}
```

```
STDMETHODIMP CGraphCtl::TipoEjes(BSTR TipoX, BSTR TipoY, BSTR TipoZ, BOOL TiempoX,
    BOOL TiempoY, BOOL TiempoZ)
{
  m_bstrFormatoX =TipoX;
  m_bstrFormatoY=TipoY;
  m_bstrFormatoZ=TipoZ;
  m_TiempoX=TiempoX;
  m_TiempoY=TiempoY;
  m_TiempoZ=TiempoZ;
  return S_OK;
}
```

```
CString CGraphCtl::FormatAxisLabel(double data, BOOL m_Time, CString Tipo)
{
  CString result;
  char format[200];
  char buffer[200];


  if (m_Time)
  {
      COleDateTime t((DATE)data);
    result = t.Format(Tipo);
  }
  else
  {
    sprintf(format,"%s",(LPCTSTR)Tipo);
    sprintf(buffer,format,data);
    result = buffer;
  }
```

```
  return result;
}
```

```
LRESULT CGraphCtl::OnKeyDown(UINT uMsg, UINT nChar, UINT nRepCnt, UINT nFlags)
{
  //La cosa marcha bien, no se porque hace el ruido
  switch (nChar)
  {
    case 87: //letra W
      m_wAngleX = m_wAngleX + 1;
      break;
        case 83: //letra S
      m_wAngleX = m_wAngleX - 1;
      break;
    case 65: //letra A
      m_wAngleY = m_wAngleY + 1;
      break;
    case 68: //letra D
      m_wAngleY = m_wAngleY - 1;
      break;
    case '1': //Original
      m_wAngleX = 30;
      m_wAngleY = -45;
      break;
    case '2': //Superior
      m_wAngleX = 90;
      m_wAngleY = 0;
      break;
    case '3': //Lateral
      m_wAngleX = 0;
      m_wAngleY = 0;
      break;
  }
  SetDirty(TRUE);
  FireViewChange();

  return WM_KEYDOWN;
}
```

```
STDMETHODIMP CGraphCtl::Idioma(long numIdioma)
{
  switch (numIdioma)
  {
    case 1: //Idioma espa\~{n}ol
            m_Idioma = 0;
      break;
    case 0: //Idioma Ingles
      m_Idioma = 100;
      break;
    default: //Si es otro ponemos espa\~{n}ol
      m_Idioma = 0;
      break;
  }

  return  S_OK;
}
```

B.2. NTGraph.ocx

B.2.1. Leyenda

```
////////////////////////////////////
//En fichero NTGraphCtrl.cpp
////////////////////////////////////
// Draw element's annotation
void CNTGraphCtrl::DrawElementLabel(CDC *pDC, CGraphElement *pElement, int index)
{
  if(m_elementShow)
  {
    CFont* pOldFont;
    TEXTMETRIC tm;
    CString strCaption;
    pOldFont = SelectFontObject (pDC,m_fontIdent);

    pDC->GetTextMetrics(&tm);
    pDC->SetTextAlign(TA_TOP | TA_LEFT);
    pDC->SetBkMode(TRANSPARENT);

    CRect rect(m_axisRect);
    //los valores de rect.left y rect.top ser\'{a}n dados en hoja propiedades
    CPoint pnt;
    if (m_xLog || m_yLog)
      pnt=CorrdinateLog(m_elementPosX,m_elementPosY);
    else
      pnt=Corrdinate(m_elementPosX,m_elementPosY);
    rect.left=pnt.x;
    rect.top=pnt.y;
    //Pintar el cuadro
    short m_nElementID = 0;
    POSITION pos = m_ElementList.FindIndex(m_nElementID);
    int largo=0;
    while(pos!=NULL)
    {
      strCaption = m_ElementList.GetAt(pos)->m_strName;
      if (largo<strCaption.GetLength())
        largo = strCaption.GetLength();
      m_nElementID++;
      pos = m_ElementList.FindIndex(m_nElementID);
    }
    int ancho = m_nElementID;
    if(m_elementCuadro)
    {
      CPoint pts[4];
      pts[0].x = rect.left;
      pts[0].y = rect.top + tm.tmHeight*ancho;

      pts[1].x = rect.left;
      pts[1].y = rect.top;

      pts[2].x = rect.left + tm.tmAveCharWidth*largo + 16;
      pts[2].y = rect.top;

      pts[3].x = rect.left + tm.tmAveCharWidth*largo + 16;
      pts[3].y = rect.top + tm.tmHeight*ancho;

      //Utilizar colores propios para borde y relleno

      m_pointPen.CreatePen (0, 0,m_elementColorBordeLeyenda);
      CPen* pPenSave = pDC->SelectObject(&m_pointPen);
      m_pointBrush.CreateSolidBrush (m_elementColorRellenoLeyenda);
      CBrush* pBrushSave = pDC->SelectObject(&m_pointBrush);
```

```
      pDC->Polygon(pts, 4);
63
      if (! IsOptimizedDraw())
      {
        pDC->SelectObject(pPenSave);
        pDC->SelectObject(pBrushSave);
      }
      m_pointPen.DeleteObject();
      m_pointBrush.DeleteObject();
    }
    //Fin Pintar el cuadro
73  m_nElementID = 0;
    pos = m_ElementList.FindIndex(m_nElementID);

    while(pos!=NULL)
    {
      pDC->SetTextColor(m_ElementList.GetAt(pos)->m_LineColor);
      strCaption = m_ElementList.GetAt(pos)->m_strName;

      if (m_ElementList.GetAt(pos)->m_nSymbol == Dots)    // Draw dots.
      {
83      if (m_pointPen.m_hObject == NULL)
          m_pointPen.CreatePen (0, 0, m_ElementList.GetAt(pos)->m_PointColor);

        CPen* pPenSave = pDC->SelectObject(&m_pointPen);

        if (m_pointBrush.m_hObject == NULL)
          m_pointBrush.CreateSolidBrush (m_ElementList.GetAt(pos)->m_PointColor);

        CBrush* pBrushSave = pDC->SelectObject(&m_pointBrush);

93      //Pintado del relleno o no
        if (m_ElementList.GetAt(pos)->m_bSolid==FALSE)
          pDC->SelectObject(pBrushSave);

        int symsz = m_ElementList.GetAt(pos)->m_nWidth ;
        if (symsz<0) symsz = 0;
        pDC->Ellipse(rect.left+7-symsz, rect.top+(tm.tmHeight/2)+ tm.tmHeight *
                m_nElementID-symsz,rect.left+7+symsz, rect.top+(tm.tmHeight/2) +
                tm.tmHeight*m_nElementID+symsz);

103     if (! IsOptimizedDraw())
        {
          pDC->SelectObject(pPenSave);
          pDC->SelectObject(pBrushSave);
        }
        m_pointPen.DeleteObject();
        m_pointBrush.DeleteObject();
      }
      if (m_ElementList.GetAt(pos)->m_nSymbol == Rectangles)    // Draw rectangles.
      {
113     if (m_pointPen.m_hObject == NULL)
          m_pointPen.CreatePen (0, 0, m_ElementList.GetAt(pos)->m_PointColor);

        CPen* pPenSave = pDC->SelectObject(&m_pointPen);

        if (m_pointBrush.m_hObject == NULL)
          m_pointBrush.CreateSolidBrush (m_ElementList.GetAt(pos)->m_PointColor);

        CBrush* pBrushSave = pDC->SelectObject(&m_pointBrush);

123     //Pintado del relleno o no
        if (m_ElementList.GetAt(pos)->m_bSolid==FALSE)
          pDC->SelectObject(pBrushSave);

        int symsz = m_ElementList.GetAt(pos)->m_nWidth ;
```

```
        if (symsz<0) symsz = 0;
        pDC->Rectangle(rect.left+7-symsz,rect.top+(tm.tmHeight/2) + tm.tmHeight *
                m_nElementID-symsz, rect.left+7+symsz, rect.top+(tm.tmHeight/2) +
                tm.tmHeight*m_nElementID+symsz);
        if (! IsOptimizedDraw())
        {
          pDC->SelectObject(pPenSave);
          pDC->SelectObject(pBrushSave);
        }
        m_pointPen.DeleteObject();
        m_pointBrush.DeleteObject();
      }

      if (m_ElementList.GetAt(pos)->m_nSymbol == Diamonds)    // Draw Diamonds
      {
        if (m_pointPen.m_hObject == NULL)
          m_pointPen.CreatePen (0, 0, m_ElementList.GetAt(pos)->m_PointColor);

        CPen* pPenSave = pDC->SelectObject(&m_pointPen);

        if (m_pointBrush.m_hObject == NULL)
          m_pointBrush.CreateSolidBrush (m_ElementList.GetAt(pos)->m_PointColor);

        CBrush* pBrushSave = pDC->SelectObject(&m_pointBrush);

        //Pintado del relleno o no
        if (m_ElementList.GetAt(pos)->m_bSolid==FALSE)
          pDC->SelectObject(pBrushSave);

        CPoint pts;
        pts.x=rect.left+7;
        pts.y=rect.top+(tm.tmHeight/2)+ tm.tmHeight*m_nElementID;


        int symsz = m_ElementList.GetAt(pos)->m_nWidth ;
        if (symsz<0) symsz = 0;

        DrawDiamond(pDC, pts, symsz);

        if (! IsOptimizedDraw())
        {
          pDC->SelectObject(pPenSave);
          pDC->SelectObject(pBrushSave);
        }
        m_pointPen.DeleteObject();
        m_pointBrush.DeleteObject();
      }
      if (m_ElementList.GetAt(pos)->m_nSymbol == DownTriangles) // Draw Down Triangles
      {
        if (m_pointPen.m_hObject == NULL)
          m_pointPen.CreatePen (0, 0, m_ElementList.GetAt(pos)->m_PointColor);

        CPen* pPenSave = pDC->SelectObject(&m_pointPen);

        if (m_pointBrush.m_hObject == NULL)
          m_pointBrush.CreateSolidBrush (m_ElementList.GetAt(pos)->m_PointColor);

        CBrush* pBrushSave = pDC->SelectObject(&m_pointBrush);

        //Pintado del relleno o no
        if (m_ElementList.GetAt(pos)->m_bSolid==FALSE)
          pDC->SelectObject(pBrushSave);

        CPoint pts;
        pts.x=rect.left+7;
        pts.y=rect.top+(tm.tmHeight/2)+ tm.tmHeight*m_nElementID;
```

```
        int symsz = m_ElementList.GetAt(pos)->m_nWidth ;
        if (symsz<0) symsz = 0;

        DrawDownTriangle(pDC, pts, symsz);

        if (! IsOptimizedDraw())
        {
          pDC->SelectObject(pPenSave);
203       pDC->SelectObject(pBrushSave);
        }
        m_pointPen.DeleteObject();
        m_pointBrush.DeleteObject();
      }
      if (m_ElementList.GetAt(pos)->m_nSymbol == RightTriangles)  // Draw Right Triangles
      {
        if (m_pointPen.m_hObject == NULL)
          m_pointPen.CreatePen (0, 0, m_ElementList.GetAt(pos)->m_PointColor);

213     CPen* pPenSave = pDC->SelectObject(&m_pointPen);

        if (m_pointBrush.m_hObject == NULL)
          m_pointBrush.CreateSolidBrush (m_ElementList.GetAt(pos)->m_PointColor);

        CBrush* pBrushSave = pDC->SelectObject(&m_pointBrush);

        //Pintado del relleno o no
        if (m_ElementList.GetAt(pos)->m_bSolid==FALSE)
          pDC->SelectObject(pBrushSave);
223
        CPoint pts;
        pts.x=rect.left+7;
        pts.y=rect.top+(tm.tmHeight/2)+ tm.tmHeight*m_nElementID;

        int symsz = m_ElementList.GetAt(pos)->m_nWidth ;
        if (symsz<0) symsz = 0;

        DrawRightTriangle(pDC, pts, symsz);

233     if (! IsOptimizedDraw())
        {
          pDC->SelectObject(pPenSave);
          pDC->SelectObject(pBrushSave);
        }
        m_pointPen.DeleteObject();
        m_pointBrush.DeleteObject();
      }
      if (m_ElementList.GetAt(pos)->m_nSymbol == UpTriangles)   // Draw Up Triangles
      {
243     if (m_pointPen.m_hObject == NULL)
          m_pointPen.CreatePen (0, 0, m_ElementList.GetAt(pos)->m_PointColor);

        CPen* pPenSave = pDC->SelectObject(&m_pointPen);

        if (m_pointBrush.m_hObject == NULL)
          m_pointBrush.CreateSolidBrush (m_ElementList.GetAt(pos)->m_PointColor);

        CBrush* pBrushSave = pDC->SelectObject(&m_pointBrush);

253     //Pintado del relleno o no
        if (m_ElementList.GetAt(pos)->m_bSolid==FALSE)
          pDC->SelectObject(pBrushSave);

        CPoint pts;
        pts.x=rect.left+7;
        pts.y=rect.top+(tm.tmHeight/2)+ tm.tmHeight*m_nElementID;
```

```
        int symsz = m_ElementList.GetAt(pos)->m_nWidth ;
        if (symsz<0) symsz = 0;
263
        DrawUpTriangle(pDC, pts, symsz);

        if (! IsOptimizedDraw())
        {
          pDC->SelectObject(pPenSave);
          pDC->SelectObject(pBrushSave);
        }

        m_pointPen.DeleteObject();
273     m_pointBrush.DeleteObject();
      }

      if (m_ElementList.GetAt(pos)->m_nSymbol == LeftTriangles) // Draw Left Triangles
      {
        if (m_pointPen.m_hObject == NULL)
          m_pointPen.CreatePen (0, 0, m_ElementList.GetAt(pos)->m_PointColor);

        CPen* pPenSave = pDC->SelectObject(&m_pointPen);

283     if (m_pointBrush.m_hObject == NULL)
          m_pointBrush.CreateSolidBrush (m_ElementList.GetAt(pos)->m_PointColor);

        CBrush* pBrushSave = pDC->SelectObject(&m_pointBrush);

        //Pintado del relleno o no
        if (m_ElementList.GetAt(pos)->m_bSolid==FALSE)
          pDC->SelectObject(pBrushSave);

        CPoint pts;
293     pts.x=rect.left+7;
        pts.y=rect.top+(tm.tmHeight/2)+ tm.tmHeight*m_nElementID;

        int symsz = m_ElementList.GetAt(pos)->m_nWidth ;
        if (symsz<0) symsz = 0;

        DrawLeftTriangle(pDC, pts, symsz);

        if (! IsOptimizedDraw())
        {
303       pDC->SelectObject(pPenSave);
          pDC->SelectObject(pBrushSave);
        }

        m_pointPen.DeleteObject();
        m_pointBrush.DeleteObject();
      }

      if (m_ElementList.GetAt(pos)->m_nSymbol == Asterisk)    // Draw Asterisks
      {
313     if (m_pointPen.m_hObject == NULL)
          m_pointPen.CreatePen (0, 0, m_ElementList.GetAt(pos)->m_PointColor);

        CPen* pPenSave = pDC->SelectObject(&m_pointPen);

        if (m_pointBrush.m_hObject == NULL)
          m_pointBrush.CreateSolidBrush (m_ElementList.GetAt(pos)->m_PointColor);

        CBrush* pBrushSave = pDC->SelectObject(&m_pointBrush);

323     //Pintado del relleno o no
        if (m_ElementList.GetAt(pos)->m_bSolid==FALSE)
          pDC->SelectObject(pBrushSave);
```

```
        CPoint pts;
        pts.x=rect.left+7;
        pts.y=rect.top+(tm.tmHeight/2)+ tm.tmHeight*m_nElementID;

        int symsz = m_ElementList.GetAt(pos)->m_nWidth ;
        if (symsz<0) symsz = 0;

        DrawAsterisk(pDC, pts, symsz);

        if (! IsOptimizedDraw())
        {
          pDC->SelectObject(pPenSave);
          pDC->SelectObject(pBrushSave);
        }

        m_pointPen.DeleteObject();
        m_pointBrush.DeleteObject();
      }

      pDC->TextOut(rect.left + 14, rect.top + tm.tmHeight *
            m_nElementID , strCaption, strCaption.GetLength());

      m_nElementID++;
      pos = m_ElementList.FindIndex(m_nElementID);
    }
    pDC->SelectObject(pOldFont);

    //Para pintar simepre la misma leyenda
    m_nElementID = 0;
    pos = m_ElementList.FindIndex(m_nElementID);

    while(pos!=NULL)
    {
      POSITION posAux = m_ElementList.FindIndex(m_nElementID+1);
      if (posAux!=NULL)
      {
        m_ElementList.GetAt(posAux)->m_bShow = m_ElementList.GetAt(pos)->m_bShow;
        m_ElementList.GetAt(posAux)->m_bCuadro = m_ElementList.GetAt(pos)->m_bCuadro;
        m_ElementList.GetAt(posAux)->m_nPosX = m_ElementList.GetAt(pos)->m_nPosX;
        m_ElementList.GetAt(posAux)->m_nPosY = m_ElementList.GetAt(pos)->m_nPosY;
        m_ElementList.GetAt(posAux)->m_ColorBordeLeyenda =
                m_ElementList.GetAt(pos)->m_ColorBordeLeyenda;
        m_ElementList.GetAt(posAux)->m_ColorRellenoLeyenda =
                m_ElementList.GetAt(pos)->m_ColorRellenoLeyenda;
      }
      m_nElementID++;
      pos = m_ElementList.FindIndex(m_nElementID);
    }
  }
}
```

```
//ElementPpg.cpp
void CElementPropPage::OnBnClickedCleyendaVisible()
{
  int visible = m_ButtonLeyenda.GetCheck();
  SetPropText("ElementShow", visible );
}
```

```
void CElementPropPage::OnBnClickedCsinCuadro()
{
  int visible = m_ButtonCuadro.GetCheck();
  SetPropText("ElementCuadro", visible );
}
```

```
void CElementPropPage::OnEnChangeEposicionX()
{
  CString str;
  GetDlgItemText(IDC_EPOSICION_X,str);
  double posX = atof(str);
  SetPropText("ElementPosX", posX);
}
```

```
void CElementPropPage::OnEnChangeEposicionY()
{
  CString str;
  GetDlgItemText(IDC_EPOSICION_Y,str);
  double posY = atof(str);
  SetPropText("ElementPosY", posY);
}
```

```
//ElementPpg.h
public:
  CColourPicker m_btnColorBordeLeyenda;
  CColourPicker m_btnColorRellenoLeyenda;
  afx_msg void OnBnClickedCleyendaVisible();
  CButton m_ButtonLeyenda;
  CButton m_ButtonCuadro;
  afx_msg void OnBnClickedCsinCuadro();
  afx_msg void OnEnChangeEposicionX();
```

B.2.2. Cursores

```
// En fichero NTGraphCtrl.cpp

//Encuentra las posiciones en logaritmico
CPoint CNTGraphCtrl::CorrdinateLog(double x, double y)
{
  double rx , ry ;
  int xPixel , yPixel ;
  CPoint retPt ;
  double TempResX, TempResY, dpixelx, dpixely;

  dpixelx = (double)m_axisRect.Width() ;
  dpixely = (double)m_axisRect.Height() ;
  TempResX = (log10(dRangeX[MAX]) - log10(dRangeX[MIN])) / dpixelx ;
  TempResY = (log10(dRangeY[MAX]) - log10(dRangeY[MIN])) / dpixely ;

  if (m_xLog)
    rx = log10(x) - log10(dRangeX[MIN]);
  else
    rx = x - dRangeX[MIN] ; // Calculate horizontal offset from origin
  if (m_yLog)
    ry = log10(y) - log10(dRangeY[MIN]);
  else
    ry = y - dRangeY[MIN]; // Calculate vertical offset from origin .

  // Convert offset to be number of pixel on screen
  if (m_xLog)
    xPixel = (int)(rx / TempResX) ;
  else
    xPixel = (int)(rx / dResX) ;
  if (m_yLog)
    yPixel = (int)(ry / TempResY) ;
  else
```

```
    yPixel = (int)(ry / dResY) ;

  //Calulate point to be drawn .
  retPt.x= xPixel + m_axisRect.left ;
  retPt.y= m_axisRect.bottom - yPixel;
  return retPt ;
}
```

```
////////////////////////////////////////////////////
// Draw Cursors
void CNTGraphCtrl::DrawCursor(CDC *pDC)
{

  POSITION pos = m_CursorList.GetHeadPosition ();

  CGraphCursor cursor = m_CursorList.GetHead ();
    int index=0;

  //Start drawing all available labels.
  while(pos)
  {
    cursor = m_CursorList.GetNext(pos);
    if(!cursor.m_bVisible)
      continue;

    if (m_cursorPen.m_hObject == NULL)
      m_cursorPen.CreatePen(PS_SOLID, 0, cursor.m_Color);

    CPen* pPenSave = pDC->SelectObject(&m_cursorPen);

    //Pinta lineas horizontales
    if(cursor.m_nStyle == XY || cursor.m_nStyle == X)
    {
            //Antiguas
      //pDC->MoveTo(Corrdinate(dRangeX[MIN],cursor.position.y));
      //pDC->LineTo(Corrdinate(dRangeX[MAX],cursor.position.y));
      pDC->MoveTo(CorrdinateLog(dRangeX[MIN],cursor.position.y));
      pDC->LineTo(CorrdinateLog(dRangeX[MAX],cursor.position.y));
    }

    //Pinta lineas verticales
    if(cursor.m_nStyle == XY || cursor.m_nStyle == Y)
    {
            //Antiguas
            //pDC->MoveTo(Corrdinate(cursor.position.x,dRangeY[MIN]));
      //pDC->LineTo(Corrdinate(cursor.position.x,dRangeY[MAX]));
      pDC->MoveTo(CorrdinateLog(cursor.position.x,dRangeY[MIN]));
      pDC->LineTo(CorrdinateLog(cursor.position.x,dRangeY[MAX]));

    }

    if (! IsOptimizedDraw())
      pDC->SelectObject(pPenSave);

    m_cursorPen.DeleteObject();
    index++;
  }
}
```

```
void CNTGraphCtrl::CursorPosition(CPoint point)
{
     double rx,ry;

   rx = PT2DBLX(point.x);
   ry = PT2DBLY(point.y);
```

```
    if(m_yLog)
    {
10     m_cursorY = dRangeY[MIN]*
          pow(10.0, (ry-dRangeY[MIN]) *
           (log10(dRangeY[MAX])-log10(dRangeY[MIN]))
          / (dRangeY[MAX]-dRangeY[MIN]) );

       ry = m_cursorY;
    }
    else m_cursorY = ry;

    if(m_xLog)
20  {
       m_cursorX = dRangeX[MIN]*
          pow(10.0, (rx-dRangeX[MIN]) *
           (log10(dRangeX[MAX])-log10(dRangeX[MIN]))
          / (dRangeX[MAX]-dRangeX[MIN]) );

       rx = m_cursorX;
    }
    else m_cursorX = rx;

30  CElementPoint pt(rx,ry);
    POSITION pos = m_CursorList.FindIndex(m_nCursorID);

    if (m_axisRect.PtInRect(point) && pos)
    {
      if(m_CursorList.GetAt(pos).m_nMode > 0)
      {
       if (m_CursorList.GetAt(pos).m_nMode == Snap && m_elementCount > 0)
       {
         pt = FindPoint(rx,ry);
40       rx = pt.x;
         ry = pt.y;
       }

       m_CursorList.GetAt(pos).position.x = rx;
       m_CursorList.GetAt(pos).position.y = ry;
       InvalidateControl(m_axisRect);

      }
    }
50
  FireCursorPosition(rx,ry);
}
```

```
CElementPoint CNTGraphCtrl::FindPoint(double cursor_x, double cursor_y)
{
  CGraphElement* pElement = m_ElementList.GetAt(m_Position);
  int i=0, index = 0;
  POSITION pos = pElement->m_PointList.GetHeadPosition();
    CElementPoint point = pElement->m_PointList.GetHead();

 8 //corregido este metodo de busqueda
  double m, n, tempCursorY, tempPointY;

  if (m_xLog && m_yLog)
  {
    m = (log10(dRangeX[MAX]) - log10(dRangeX[MIN]))/
        (log10(dRangeX[MAX])-log10(dRangeY[MIN]));
    n =  log10(dRangeX[MIN]) - (log10(dRangeY[MIN]) * m);
    tempCursorY = m*log10(cursor_y) + n;
    tempPointY  = m*log10(point.y) + n;
18  }
  else if (m_xLog)
    {
```

```
      m = (log10(dRangeX[MAX]) - log10(dRangeX[MIN]))/
          (dRangeY[MAX]-dRangeY[MIN]);
      n =  log10(dRangeX[MIN]) - (dRangeY[MIN] * m);
      tempCursorY = m*cursor_y + n;
      tempPointY  = m*point.y + n;
    }
    else if (m_yLog)
      {
        m = (dRangeX[MAX] - dRangeX[MIN])/
          (log10(dRangeY[MAX])-log10(dRangeY[MIN]));
        n = dRangeX[MIN] - (log10(dRangeY[MIN]) * m);
        tempCursorY = m*log10(cursor_y) + n;
        tempPointY  = m*log10(point.y) + n;
      }
      else
      {
        m = (dRangeX[MAX] - dRangeX[MIN])/(dRangeY[MAX]-dRangeY[MIN]);
        n =  dRangeX[MIN] - (dRangeY[MIN] * m);
        tempCursorY = m*cursor_y + n;
        tempPointY  = m*point.y + n;
      }
  double dx;
  if (m_xLog && m_yLog)
    dx = fabs(log10(cursor_x) - log10(point.x));
  else if (m_xLog)
      dx = fabs(log10(cursor_x) - log10(point.x));
    else if (m_yLog)
        dx = fabs(cursor_x - point.x);
      else
        dx = fabs(cursor_x - point.x);

  double dy = fabs(tempCursorY - tempPointY);

  double dr = sqrt(dx*dx + dy*dy);

  while(pos)
  {
    point = pElement->m_PointList.GetNext(pos);
    if (m_xLog && m_yLog)
    {
      tempCursorY = m*log10(cursor_y) + n;
      tempPointY  = m*log10(point.y) + n;
    }
    else if (m_xLog)
      {
        tempCursorY = m*cursor_y + n;
        tempPointY  = m*point.y + n;
      }
      else if (m_yLog)
        {
          tempCursorY = m*log10(cursor_y) + n;
          tempPointY  = m*log10(point.y) + n;
        }
        else
        {
          tempCursorY = m*cursor_y + n;
          tempPointY  = m*point.y + n;
        }
    if (m_xLog && m_yLog)
      dx = fabs(log10(cursor_x) - log10(point.x));
    else if (m_xLog)
        dx = fabs(log10(cursor_x) - log10(point.x));
      else if (m_yLog)
          dx = fabs(cursor_x - point.x);
      else
        dx = fabs(cursor_x - point.x);
```

```
88      dy = fabs(tempCursorY - tempPointY);


        if (sqrt(dx*dx+dy*dy) < dr )
        {
          dr = sqrt(dx*dx+dy*dy);
                index = i;
        }
        i++;
      }
98
      pos = pElement->m_PointList.FindIndex(index);

      return pElement->m_PointList.GetAt(pos);
    }
```

B.2.3. Nueva Página de Propiedades

```
    // SubMenuPpg.h: archivo de definiciones
    #pragma once
    #include "NTGraphCtl.h"

    // CSubMenuPpg: cuadro de di\'{a}logo de la p\'{a}gina de propiedades
    class CSubMenuPpg : public COlePropertyPage
    {
 8    DECLARE_DYNCREATE(CSubMenuPpg)
      DECLARE_OLECREATE_EX(CSubMenuPpg)

    // Constructores
    public:
      CSubMenuPpg();

      BOOL OnHelp(LPCTSTR helpdir);
        // Datos del cuadro de di\'{a}logo
      enum { IDD = IDD_PROPPAGE_SUBMENU };
18    short desp_Idioma;
        // Implementaci\'{o}n
    protected:
      virtual void DoDataExchange(CDataExchange* pDX);        // Compatibilidad con DDX o DDV
    private:
      bool m_bCanSaveChgs; //XAT

     // Mapas de mensajes
    protected:
      DECLARE_MESSAGE_MAP()
28    afx_msg BOOL OnHelpInfo(HELPINFO* pHelpInfo);

    public:
      afx_msg void OnBnClickedBguardarBitmap();
      afx_msg void OnBnClickedBimprimir();
      afx_msg void OnBnClickedBcopiarPortapapeles();
      BOOL OnInitDialog();
      afx_msg void OnDeltaposSpalto(NMHDR *pNMHDR, LRESULT *pResult);
      afx_msg void OnDeltaposSpancho(NMHDR *pNMHDR, LRESULT *pResult);
      afx_msg void OnEnChangeEalto();
38    afx_msg void OnEnChangeEancho();
    };
```

```
 1  // SubMenuPpg.cpp: archivo de implementaci\'{o}n
    #include "stdafx.h"
    #include "NTGraph.h"
```

```
#include "SubMenuPpg.h"
#include ".\submenuppg.h"

// Cuadro de di\'{a}logo de CSubMenuPpg
IMPLEMENT_DYNCREATE(CSubMenuPpg, COlePropertyPage)

// Mapa de mensajes
BEGIN_MESSAGE_MAP(CSubMenuPpg, COlePropertyPage)
  ON_BN_CLICKED(IDC_BGUARDAR_BITMAP, OnBnClickedBguardarBitmap)
  ON_BN_CLICKED(IDC_BIMPRIMIR, OnBnClickedBimprimir)
  ON_BN_CLICKED(IDC_BCOPIAR_PORTAPAPELES, OnBnClickedBcopiarPortapapeles)
  ON_NOTIFY(UDN_DELTAPOS, IDC_SPALTO, OnDeltaposSpalto)
  ON_NOTIFY(UDN_DELTAPOS, IDC_SPANCHO, OnDeltaposSpancho)
  ON_EN_CHANGE(IDC_EALTO, OnEnChangeEalto)
  ON_EN_CHANGE(IDC_EANCHO, OnEnChangeEancho)
END_MESSAGE_MAP()
```

```
// Controladores de mensajes de CSubMenuPpg

void CSubMenuPpg::OnBnClickedBguardarBitmap()
{
  BYTE guardar = TRUE;
  SetPropText("GuardarBitmap",guardar);
}
```

```
void CSubMenuPpg::OnBnClickedBimprimir()
{
  BYTE imp = TRUE;
  SetPropText("Imprimir",imp);
}
```

```
void CSubMenuPpg::OnBnClickedBcopiarPortapapeles()
{
  BYTE copiar = TRUE;
  SetPropText("CopiarPortapapeles",copiar);
}
```

```
BOOL CSubMenuPpg::OnInitDialog()
{
  SetDlgItemInt(IDC_EALTO,100);
  SetDlgItemInt(IDC_EANCHO,100);

  CSpinButtonCtrl* pSpinAlto;
  pSpinAlto = (CSpinButtonCtrl*)GetDlgItem(IDC_SPALTO);

  // Set the buddy control
  pSpinAlto->SetBuddy (GetDlgItem(IDC_EALTO));

  // Set Spin Control Range
  pSpinAlto->SetRange(0,260);

  CSpinButtonCtrl* pSpinAncho;
  pSpinAncho = (CSpinButtonCtrl*)GetDlgItem(IDC_SPANCHO);

  // Set the buddy control
  pSpinAncho->SetBuddy (GetDlgItem(IDC_EANCHO));

  // Set Spin Control Range
  pSpinAncho->SetRange(0,180);

  //Cambio al lenguaje oportuno
  //Con esto ponemos todas las ventanas en el idioma correcto
  GetPropText("Idioma1", &desp_Idioma);
  CString resString;
```

```
  resString.LoadString(::GetModuleHandle("NTGraph.ocx"),
    IDS_GUARDAR_BITMAP + desp_Idioma, NULL);
  SetDlgItemText(IDC_BGUARDAR_BITMAP,resString);
  resString.LoadString(::GetModuleHandle("NTGraph.ocx"),
    IDS_IMPRIMIR + desp_Idioma, NULL);
  SetDlgItemText(IDC_BIMPRIMIR,resString);
  resString.LoadString(::GetModuleHandle("NTGraph.ocx"),
    IDS_COPIAR_PORTAPAPELES + desp_Idioma, NULL);
  SetDlgItemText(IDC_BCOPIAR_PORTAPAPELES,resString);
  resString.LoadString(::GetModuleHandle("NTGraph.ocx"),IDS_ALTO + desp_Idioma, NULL);
  SetDlgItemText(IDC_SALTO,resString);
  resString.LoadString(::GetModuleHandle("NTGraph.ocx"),IDS_ANCHO + desp_Idioma, NULL);
  SetDlgItemText(IDC_SANCHO,resString);
  resString.LoadString(::GetModuleHandle("NTGraph.ocx"),IDS_NOTA1 + desp_Idioma, NULL);
  SetDlgItemText(IDC_SNOTA,resString);
  resString.LoadString(::GetModuleHandle("NTGraph.ocx"),IDS_NOTA2 + desp_Idioma, NULL);
  SetDlgItemText(IDC_SNOTA2,resString);

  return 0;
}
```

```
void CSubMenuPpg::OnDeltaposSpalto(NMHDR *pNMHDR, LRESULT *pResult)
{
  NM_UPDOWN* pNMUpDown = (NM_UPDOWN*)pNMHDR;
  int alto=GetDlgItemInt(IDC_EALTO);
  SetPropText( "porcAlto", alto);
  *pResult = 0;
}
```

```
void CSubMenuPpg::OnDeltaposSpancho(NMHDR *pNMHDR, LRESULT *pResult)
{
  NM_UPDOWN* pNMUpDown = (NM_UPDOWN*)pNMHDR;

  int ancho=GetDlgItemInt(IDC_EANCHO);
  SetPropText( "porcAncho", ancho);

  *pResult = 0;
}
```

```
void CSubMenuPpg::OnEnChangeEalto()
{
  int alto=GetDlgItemInt(IDC_EALTO);
  SetPropText( "porcAlto", alto);
}
```

```
void CSubMenuPpg::OnEnChangeEancho()
{
  int ancho=GetDlgItemInt(IDC_EANCHO);
  SetPropText( "porcAncho", ancho);
}
```

B.2.4. Idioma del Objeto

```
\label{Idioma_2D}
// NTGraphCtl.cpp
void CNTGraphCtrl::SetIdioma(short nNewValue)
{
  switch (nNewValue)
  {
    case 1: //Idioma espa\~{n}ol
```

```
            m_Idioma = 0;
        break;
      case 0: //Idioma Ingles
        m_Idioma = 300;
        break;
      default: //Si es otro ponemos espa\~{n}ol
        m_Idioma = 0;
        break;
    }

    SetModifiedFlag();
  }
```

```
void CNTGraphCtrl::OnIdiomaChanged()
{
  SetModifiedFlag();
}
```

```
////////////////////////////////
//varias cadenas con los siguientes formatos repartidas por el fichero
    CString resString;
  resString.LoadString(::GetModuleHandle("NTGraph.ocx"),IDS_ELEMENT_FOUND + m_Idioma, NULL);
  AfxMessageBox(resString);
////////////////////////////////
  resString.LoadString(::GetModuleHandle("NTGraph.ocx"),IDS_NOTA2 + desp_Idioma, NULL);
  SetDlgItemText(IDC_SNOTA2,resString);
////////////////////////////////
```

B.3. Tabla Elementos.ocx

B.3.1. Tabla de Elementos

```
////////////////////////////////
//TablaElementosCtrl.h
////////////////////////////////
class CTablaElementosCtrl : public COleControl
{
  DECLARE_DYNCREATE(CTablaElementosCtrl)

// Constructor
public:
  CTablaElementosCtrl();

// Reemplazos
public:
  virtual void OnDraw(CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid);
  virtual void DoPropExchange(CPropExchange* pPX);
  virtual void OnResetState();

// Implementaci\'{o}n
protected:
  ~CTablaElementosCtrl();

  DECLARE_OLECREATE_EX(CTablaElementosCtrl)    // Generador y guid de clases
  DECLARE_OLETYPELIB(CTablaElementosCtrl)      // Obtener la informaci\'{o}n de tipos
  // Identificadores de la p\'{a}gina de propiedades
    DECLARE_PROPPAGEIDS(CTablaElementosCtrl)
  DECLARE_OLECTLTYPE(CTablaElementosCtrl)   // Escribir el nombre y los diversos estados
```

```
// Mapas de mensajes
  DECLARE_MESSAGE_MAP()

// Mapas de env\'{\i}o
  DECLARE_DISPATCH_MAP()
  afx_msg void AboutBox();

// Mapas de eventos
  DECLARE_EVENT_MAP()

// Identificadores de env\'{\i}o y de eventos
public:
  enum {
    dispidRuta = 7,
    dispidElemento = 6,
    dispidIdioma = 5,
    dispidFuenteNombre = 4,
    dispidFuenteNumero = 3,
    dispidFuenteNotas = 2,
    dispidFuenteElemento = 1
  };
private:
  CRect m_ctlRect, m_ejesRect;
  double dResY, dResX ;
  bool Inicializado, Movimiento;
  double rx,ry;
  int Desp_Idioma;
protected:
  CFont m_FuenteY;
  void OnFontChanged();
  LPFONTDISP ObtenerFuenteElemento(void);
  void SetFuenteElemento(LPFONTDISP pVal);
  CFontHolder m_FuenteElemento;
  LPFONTDISP ObtenerFuenteNotas(void);
  void SetFuenteNotas(LPFONTDISP pVal);
  CFontHolder m_FuenteNotas;
  LPFONTDISP ObtenerFuenteNumero(void);
  void SetFuenteNumero(LPFONTDISP pVal);
  CFontHolder m_FuenteNumero;
  LPFONTDISP ObtenerFuenteNombre(void);
  void SetFuenteNombre(LPFONTDISP pVal);
  CFontHolder m_FuenteNombre;
  SHORT ObtenerIdioma(void);
  void SetIdioma(SHORT nuevoValor);
  SHORT m_Idioma;
  void OnLButtonDblClk(UINT nFlags, CPoint point);
  void OnMouseMove(UINT nFlags, CPoint point);
  SHORT ObtenerElemento(void);
  void SetElemento(SHORT nuevoValor);
  SHORT m_Elemento;
  BSTR ObtenerRuta(void);
  void SetRuta(LPCTSTR newVal);
  LPCTSTR m_Ruta;
public:
  void DibujarTabla(CDC* pDC);
  void PreparandoArea(CDC *pDC, CRect rect);
  // Crear fuentes inderectamente
  void CrearFuente(CDC*  pDC);
  // Calculo del Rectangulo de dibujo
  void CalcRect(CDC* pDC);
  CString FormatoEtiquetas(double data);
  // Calculo de la resoluci\'{o}n por punto
  void CalcResolucion(void);
  // Calculo de coordenada por punto de pantalla
  CPoint Coordenada(double x, double y);
  // Relleno de la tabla con los elementos correspondientes
```

```
    void DibujarElementos(CDC* pCD);
    // Numero de grupos y periodos
    void DibujarRotulos(CDC* pDC);
    // Define los colores de las zonos diferentes de la tabla
    void DibujarZonas(CDC* pDC);
    // Pintado de los n\'{u}meros at\'{o}micos de los elementos
    void DibujarNumeros(CDC* pDC);
    // Escribe el nombre de los elementos
    void DibujarNombre(CDC* pDC);
    // Nos muestra un marco de color en el elemento sobre el que se situa el rat\'{o}n
    void DibujarMovRaton(CDC* pDC);
    // Escribe el valor de la masa at\'{o}mica promedio de cada elemento
    void DibujarMasa(CDC* pDC);
  private:
    // Crea el fichero de nombres de elementos si no existe
    int CrearFichero(LPCSTR Ruta);
  }
```

```
  ////////////////////////////////
  //TablaElementosCtrl.cpp
  ////////////////////////////////
  //definimos esquinas izquierdas de grupos
  #define g0    0
  #define g1   10
  #define g2   20
  #define g3   30
  #define g4   40
  #define g5   50
  #define g6   60
  #define g7   70
  #define g8   80
  #define g9   90
  #define g10 100
  #define g11 110
  #define g12 120
  #define g13 130
  #define g14 140
  #define g15 150
  #define g16 160
  #define g17 170
  #define g18 180

  //definimos esquinas inferiores de periodos
  #define p0  93
  #define p1  83
  #define p2  73
  #define p3  63
  #define p4  53
  #define p5  43
  #define p6  33
  #define p7  23
  #define sep 20
  #define p6a 10
  #define p7a  0

  //funciones de traducci\'{o}n de puntos
  #define PT2DBLX(x) (double)((x - m_ejesRect.left)*dResX)
  #define PT2DBLY(y) (double)((m_ejesRect.bottom - y)*dResY)

  //Tipos de letra de los componentes
  static const FONTDESC _FontDescElemento =
    { sizeof(FONTDESC), OLESTR("Times␣New␣Roman"), FONTSIZE( 12 ), FW_BOLD,
       ANSI_CHARSET, FALSE, FALSE, FALSE };

  static const FONTDESC _FontDescNotas =
    { sizeof(FONTDESC), OLESTR("Courier"), FONTSIZE( 10 ), FW_NORMAL,
```

```
      ANSI_CHARSET, FALSE, FALSE, FALSE };

static const FONTDESC _FontDescNumero =
  { sizeof(FONTDESC), OLESTR("Arial"), FONTSIZE( 8 ), FW_BOLD,
      ANSI_CHARSET, FALSE, FALSE, FALSE };

static const FONTDESC _FontDescNombre =
  { sizeof(FONTDESC), OLESTR("Tahoma"), FONTSIZE( 7 ), FW_NORMAL,
      ANSI_CHARSET, FALSE, FALSE, FALSE };
```

```
// CTablaElementosCtrl::OnDraw: funci\'{o}n para dibujar
void CTablaElementosCtrl::OnDraw(CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid)
{
  if (!pdc)
    return;
  if (!Movimiento)
  {
    pdc->FillRect(rcBounds, CBrush::FromHandle((HBRUSH )GetStockObject(WHITE_BRUSH)));
    PreparandoArea(pdc, rcBounds);
    //Inicializaci\'{o}n de las fuentes
    if (!Inicializado)
    {
      Inicializado = TRUE;
      m_FuenteElemento.InitializeFont(&_FontDescElemento);
      m_FuenteNombre.InitializeFont(&_FontDescNombre);
      m_FuenteNotas.InitializeFont(&_FontDescNotas);
      m_FuenteNumero.InitializeFont(&_FontDescNumero);
    }

    DibujarZonas(pdc);
    DibujarTabla(pdc);
    DibujarNumeros(pdc);
    DibujarMasa(pdc);
    DibujarNombre(pdc);
    DibujarElementos(pdc);
    DibujarRotulos(pdc);
    DibujarMovRaton(pdc);
  }
  else
  {
    Movimiento = FALSE;
    DibujarTabla(pdc);
    DibujarMovRaton(pdc);
  }

}
\end{codigo
// Dibujado de todo el marco de la tabla
void CTablaElementosCtrl::DibujarTabla(CDC* pDC)
{
  CPen pen(PS_SOLID, 1, RGB(0, 0, 0));
  CPen* pOldPen = pDC->SelectObject(&pen);
  //Divisi\'{o}n en grupos
  pDC->MoveTo(Coordenada( g0,p0));
  pDC->LineTo(Coordenada( g0,p7));
  pDC->MoveTo(Coordenada( g1,p0));
  pDC->LineTo(Coordenada( g1,p7));
  pDC->MoveTo(Coordenada( g2,p1));
  pDC->LineTo(Coordenada( g2,p7));
  pDC->MoveTo(Coordenada( g3,p3));
  pDC->LineTo(Coordenada( g3,p7));
  pDC->MoveTo(Coordenada( g4,p3));
  pDC->LineTo(Coordenada( g4,p7));
  pDC->MoveTo(Coordenada( g5,p3));
  pDC->LineTo(Coordenada( g5,p7));
  pDC->MoveTo(Coordenada( g6,p3));
```

```
pDC->LineTo(Coordenada( g6,p7));
pDC->MoveTo(Coordenada( g7,p3));
pDC->LineTo(Coordenada( g7,p7));
pDC->MoveTo(Coordenada( g8,p3));
pDC->LineTo(Coordenada( g8,p7));
pDC->MoveTo(Coordenada( g9,p3));
pDC->LineTo(Coordenada( g9,p7));
pDC->MoveTo(Coordenada(g10,p3));
pDC->LineTo(Coordenada(g10,p7));
pDC->MoveTo(Coordenada(g11,p3));
pDC->LineTo(Coordenada(g11,p7));
pDC->MoveTo(Coordenada(g12,p1));
pDC->LineTo(Coordenada(g12,p7));
pDC->MoveTo(Coordenada(g13,p1));
pDC->LineTo(Coordenada(g13,p7));
pDC->MoveTo(Coordenada(g14,p1));
pDC->LineTo(Coordenada(g14,p7));
pDC->MoveTo(Coordenada(g15,p1));
pDC->LineTo(Coordenada(g15,p7));
pDC->MoveTo(Coordenada(g16,p1));
pDC->LineTo(Coordenada(g16,p7));
pDC->MoveTo(Coordenada(g17,p0));
pDC->LineTo(Coordenada(g17,p7));
pDC->MoveTo(Coordenada(g18,p0));
pDC->LineTo(Coordenada(g18,p7));
//Divisi\'{o}n en peri\'{o}dos
pDC->MoveTo(Coordenada( g0,p0));
pDC->LineTo(Coordenada( g1,p0));
pDC->MoveTo(Coordenada(g17,p0));
pDC->LineTo(Coordenada(g18,p0));
pDC->MoveTo(Coordenada( g0,p1));
pDC->LineTo(Coordenada( g2,p1));
pDC->MoveTo(Coordenada(g12,p1));
pDC->LineTo(Coordenada(g18,p1));
pDC->MoveTo(Coordenada( g0,p2));
pDC->LineTo(Coordenada( 20,p2));
pDC->MoveTo(Coordenada(g12,p2));
pDC->LineTo(Coordenada(g18,p2));
pDC->MoveTo(Coordenada( g0,p3));
pDC->LineTo(Coordenada(g18,p3));
pDC->MoveTo(Coordenada( g0,p4));
pDC->LineTo(Coordenada(g18,p4));
pDC->MoveTo(Coordenada( g0,p5));
pDC->LineTo(Coordenada(g18,p5));
pDC->MoveTo(Coordenada( g0,p6));
pDC->LineTo(Coordenada(g18,p6));
pDC->MoveTo(Coordenada( g0,p7));
pDC->LineTo(Coordenada(g18,p7));

//Lantanidos y Actinidos
pDC->MoveTo(Coordenada( g4,sep));
pDC->LineTo(Coordenada(g18,sep));
pDC->MoveTo(Coordenada( g4,p6a));
pDC->LineTo(Coordenada(g18,p6a));
pDC->MoveTo(Coordenada( g4,p7a));
pDC->LineTo(Coordenada(g18,p7a));
pDC->MoveTo(Coordenada( g4,sep));
pDC->LineTo(Coordenada( g4,p7a));
pDC->MoveTo(Coordenada( g5,sep));
pDC->LineTo(Coordenada( g5,p7a));
pDC->MoveTo(Coordenada( g6,sep));
pDC->LineTo(Coordenada( g6,p7a));
pDC->MoveTo(Coordenada( g7,sep));
pDC->LineTo(Coordenada( g7,p7a));
pDC->MoveTo(Coordenada( g8,sep));
pDC->LineTo(Coordenada( g8,p7a));
```

```
  pDC->MoveTo(Coordenada( g9,sep));
  pDC->LineTo(Coordenada( g9,p7a));
  pDC->MoveTo(Coordenada(g10,sep));
  pDC->LineTo(Coordenada(g10,p7a));
  pDC->MoveTo(Coordenada(g11,sep));
  pDC->LineTo(Coordenada(g11,p7a));
  pDC->MoveTo(Coordenada(g12,sep));
  pDC->LineTo(Coordenada(g12,p7a));
  pDC->MoveTo(Coordenada(g13,sep));
  pDC->LineTo(Coordenada(g13,p7a));
  pDC->MoveTo(Coordenada(g14,sep));
  pDC->LineTo(Coordenada(g14,p7a));
  pDC->MoveTo(Coordenada(g15,sep));
  pDC->LineTo(Coordenada(g15,p7a));
  pDC->MoveTo(Coordenada(g16,sep));
  pDC->LineTo(Coordenada(g16,p7a));
  pDC->MoveTo(Coordenada(g17,sep));
  pDC->LineTo(Coordenada(g17,p7a));
  pDC->MoveTo(Coordenada(g18,sep));
  pDC->LineTo(Coordenada(g18,p7a));

  pDC->SelectObject(&pOldPen);
}
```

```
// Preparamos el \'{a}rea de dibujo, para de jar margenes para textos
void CTablaElementosCtrl::PreparandoArea(CDC *pDC, CRect rect)
{
  pDC->SetMapMode(MM_TEXT) ;
  pDC->SetWindowOrg(0,0) ;
  pDC->SetWindowExt(rect.right , rect.bottom) ;
  pDC->SetViewportOrg(0,0) ;
  pDC->SetViewportExt (rect.right , rect.bottom );

  m_ctlRect = rect ;
  pDC->DPtoLP(&m_ctlRect);
  m_FuenteY.DeleteObject();
  CrearFuente(pDC); //Crear la fuente del eje Y
  CalcRect(pDC) ;   // Rectangulo de dibujo
  CalcResolucion(); // Resoluci\'{o}n por punto de pantalla
}
```

```
// Calculo del Rectangulo de dibujo
void CTablaElementosCtrl::CalcRect(CDC* pDC)
{
  int offset = 15;

  CSize txtXTamano,txtYTamano(0,0),txtTituloTamano ;
  CString str ;
  CFont *pOldFuente, *pFuenteGuardada ;

  pOldFuente = SelectFontObject(pDC, m_FuenteNotas);

  // Calculando tama\~{n}o de etiquetas en Y.
  str = FormatoEtiquetas(7); //M\'{a}ximo periodo
    txtYTamano = pDC->GetTextExtent(str);

  //Calculando tama\~{n}o de etiquetas en X.
  str = FormatoEtiquetas(18); //m\'{a}ximo grupo
  txtXTamano = pDC->GetTextExtent(str);

  pDC->SelectObject(pOldFuente);
  pFuenteGuardada = SelectFontObject(pDC,m_FuenteElemento);

  const CString& strCaption = InternalGetText();
  txtTituloTamano = pDC->GetTextExtent(strCaption);
```

```
  pDC->SelectObject(pFuenteGuardada);
  //Definici\'{o}n del area de dibujo
  m_ejesRect.left = m_ctlRect.left + txtYTamano.cy + txtYTamano.cx + offset;
  m_ejesRect.right = m_ctlRect.right - (txtXTamano.cx/2) - offset;
  m_ejesRect.top = m_ctlRect.top + txtTituloTamano.cy + txtYTamano.cy + offset;
  m_ejesRect.bottom= m_ctlRect.bottom - (txtXTamano.cy/2) - offset;
}
```

```
// Calculo de la resoluci\'{o}n por punto
void CTablaElementosCtrl::CalcResolucion(void)
{
  double dpixelx, dpixely ;

  dpixelx = (double)m_ejesRect.Width() ;
  dpixely = (double)m_ejesRect.Height() ;
  dResY = AltoTabla / dpixely ;
  dResX = AnchoTabla / dpixelx ;
}
```

```
// Calculo de coordenada por punto de pantalla
CPoint CTablaElementosCtrl::Coordenada(double x, double y)
{
  int xPixel , yPixel ;
  CPoint retPt ;

    // Conmvertimos el valor a un valor de pixel de pantalla
  xPixel = (int)(x / dResX) ;
  yPixel = (int)(y / dResY) ;

  // Calculamos el punto que se debe pintar
  retPt.x= xPixel + m_ejesRect.left ;
  retPt.y= m_ejesRect.bottom - yPixel;
  return retPt ;
}
```

```
// Define los colores de las zonos diferentes de la tabla
void CTablaElementosCtrl::DibujarZonas(CDC* pDC)
{
  //Definimos sin linea
  CPen pen(PS_NULL, 0, RGB(0, 0, 0));
  CPen* pOldPen = pDC->SelectObject(&pen);

  //Creamos las brochas de pintura
  CBrush brushVerde(RGB(128, 255, 128));
  CBrush brushGris(RGB(192, 192, 192));
  CBrush brushRosa(RGB(255, 121, 141));
  CBrush brushAmarillo(RGB(255, 200, 50));
  CBrush brushAzul(RGB(128, 128, 255));
  CBrush* pOldBrush;

  //Definimos los vertices y pintamos

  //Elementos Nuevos
  pOldBrush = pDC->SelectObject(&brushGris);
  CPoint pts1[4];
  pts1[0] = Coordenada(g9,p6);
  pts1[1] = Coordenada(g18,p6);
  pts1[2] = Coordenada(g18,p7);
  pts1[3] = Coordenada(g9,p7);
  pDC->Polygon(pts1,4);
  pDC->SelectObject(&pOldBrush);

  //Gases Nobles
  pOldBrush = pDC->SelectObject(&brushAzul);
  CPoint pts2[4];
```

```
pts2[0] = Coordenada(g17,p0);
pts2[1] = Coordenada(g18,p0);
pts2[2] = Coordenada(g18,p6);
pts2[3] = Coordenada(g17,p6);
pDC->Polygon(pts2,4);
pDC->SelectObject(&pOldBrush);

//No Metales e Hidrogeno
pOldBrush = pDC->SelectObject(&brushRosa);
CPoint pts3[10];
pts3[0] = Coordenada(g13,p1);
pts3[1] = Coordenada(g13,p2);
pts3[2] = Coordenada(g14,p2);
pts3[3] = Coordenada(g14,p3);
pts3[4] = Coordenada(g15,p3);
pts3[5] = Coordenada(g15,p4);
pts3[6] = Coordenada(g16,p4);
pts3[7] = Coordenada(g16,p5);
pts3[8] = Coordenada(g17,p5);
pts3[9] = Coordenada(g17,p1);
  pDC->Polygon(pts3,10);
CPoint pts4[4];
pts4[0] = Coordenada(g0,p0);
pts4[1] = Coordenada(g1,p0);
pts4[2] = Coordenada(g1,p1);
pts4[3] = Coordenada(g0,p1);
pDC->Polygon(pts4,4);
pDC->SelectObject(&pOldBrush);

//Metaloides
pOldBrush = pDC->SelectObject(&brushAmarillo);
CPoint pts5[18];
pts5[0] = Coordenada(g13,p1);
pts5[1] = Coordenada(g13,p2);
pts5[2] = Coordenada(g14,p2);
pts5[3] = Coordenada(g14,p3);
pts5[4] = Coordenada(g15,p3);
pts5[5] = Coordenada(g15,p4);
pts5[6] = Coordenada(g16,p4);
pts5[7] = Coordenada(g16,p5);
pts5[8] = Coordenada(g17,p5);
pts5[9] = Coordenada(g17,p6);
pts5[10] = Coordenada(g15,p6);
pts5[11] = Coordenada(g15,p5);
pts5[12] = Coordenada(g14,p5);
pts5[13] = Coordenada(g14,p4);
pts5[14] = Coordenada(g13,p4);
pts5[15] = Coordenada(g13,p2);
pts5[16] = Coordenada(g12,p2);
pts5[17] = Coordenada(g12,p1);
  pDC->Polygon(pts5,18);
pDC->SelectObject(&pOldBrush);
//Metales
//Lantanidos y Actinidos
pOldBrush = pDC->SelectObject(&brushVerde);
CPoint pts6[4];
pts6[0] = Coordenada(g4,sep);
pts6[1] = Coordenada(g18,sep);
pts6[2] = Coordenada(g18,p7a);
pts6[3] = Coordenada(g4,p7a);
pDC->Polygon(pts6,4);
//Resto
CPoint pts7[14];
pts7[0] = Coordenada(g0,p1);
pts7[1] = Coordenada(g0,p7);
pts7[2] = Coordenada(g9,p7);
```

```
  pts7[3] = Coordenada(g9,p6);
  pts7[4] = Coordenada(g15,p6);
  pts7[5] = Coordenada(g15,p5);
  pts7[6] = Coordenada(g14,p5);
  pts7[7] = Coordenada(g14,p4);
  pts7[8] = Coordenada(g13,p4);
  pts7[9] = Coordenada(g13,p2);
  pts7[10] = Coordenada(g12,p2);
  pts7[11] = Coordenada(g12,p3);
  pts7[12] = Coordenada(g2,p3);
  pts7[13] = Coordenada(g2,p1);
  pDC->Polygon(pts7,14);
  pDC->SelectObject(&pOldBrush);

  pDC->SelectObject(&pOldPen);
}
```

```
// Escribe el nombre de los elementos
void CTablaElementosCtrl::DibujarNombre(CDC* pDC)
{
  char rutaFichero[300];

  switch (m_Idioma)
  {
    case 0:
      sprintf(rutaFichero,"%sElementos_Espanol.txt",m_Ruta);

      break;
    case 1:
      sprintf(rutaFichero,"%sElementos_Ingles.txt",m_Ruta);
      break;
    default:
      sprintf(rutaFichero,"%sElementos_Espanol.txt",m_Ruta);
      break;
  }
  FILE* Fichero;
  if ((Fichero = fopen(rutaFichero,"r"))==NULL)
  {
    //Cambiar
    char temp[300];
    int dev = CrearFichero(rutaFichero);
    if (dev)
    {
      CString array[] = {"No se ha podido abrir el fichero de nombres de Elementos en:"
        ,"Do not open the Elements names File in:"};

      sprintf(temp, "%s\n%s",array[m_Idioma],rutaFichero);
      MessageBox(temp,NULL,MB_ICONERROR);
      exit (-1);
    }
    else
    {
      CString array[] = {"Se ha creado el fichero de nombres en la ruta:"
        ,"The names file was created in:"};

      sprintf(temp, "%s\n%s",array[m_Idioma],rutaFichero);
      MessageBox(temp,NULL,MB_ICONINFORMATION);
      Refresh();
      Fichero = fopen(rutaFichero,"r");
    }
  }

  char strTemp[300];
  char *token;
  char seps[] = ",;\n";
  CString strG1[7],strG2[6],strG3[4],strG4[4],strG5[4],strG6[4],
```

```
      strG7[4],strG8[4],strG9[4],strG10[4],strG11[4],strG12[4],strG13[6],
      strG14[6],strG15[6],strG16[6],strG17[6],strG18[7],Lan[14],Act[14];
    int Valor;

    Valor=0;
    fgets(strTemp,200,Fichero);
    token = strtok( strTemp, seps );
    while( token != NULL )
58  {
      strG1[Valor]= token;
      // Obtener token siguiente:
      token = strtok( NULL, seps );
      Valor++;
    }
  {...}
      Valor=0;
    fgets(strTemp,200,Fichero);
    token = strtok( strTemp, seps );
68  while( token != NULL )
    {
      Act[Valor]= token;
      // Obtener token siguiente:
      token = strtok( NULL, seps );
      Valor++;
    }
    fclose(Fichero);

    CFont* pOldFuente;
78  pOldFuente = SelectFontObject (pDC,m_FuenteNombre);

      pDC->SetBkMode(TRANSPARENT);

    CPoint point;
    CSize txtSize;
    pDC->SetTextAlign(TA_LEFT);

    //Inicio de escritura de elementos
    char cadena[20];
88  int cont=0;
    for (cont=0; cont<=6; cont++)
    {
      txtSize = pDC->GetTextExtent(strG1[cont]);
      while (PT2DBLX(txtSize.cx + m_ejesRect.left) > 9)
      {
        for (int i = 0; i < strG1[cont].GetLength()-2; i++)
          cadena[i] = strG1[cont][i];
        cadena[strG1[cont].GetLength()-2] = '.';
        cadena[strG1[cont].GetLength()-1] = '\0';
98      strG1[cont] = cadena;
        txtSize = pDC->GetTextExtent(strG1[cont]);
      }
      point = Coordenada(g1-9.5,p1-10*cont);
      point.y -= txtSize.cy;
      pDC->TextOut(point.x,point.y,strG1[cont]);
    }
  {...}
      for (cont=0; cont<=13; cont++)
    {
108   txtSize = pDC->GetTextExtent(Act[cont]);
      while (PT2DBLX(txtSize.cx + m_ejesRect.left) > 9)
      {
        for (int i = 0; i < Act[cont].GetLength()-2; i++)
          cadena[i] = Act[cont][i];
        cadena[Act[cont].GetLength()-2] = '.';
        cadena[Act[cont].GetLength()-1] = '\0';
        Act[cont] = cadena;
```

```
        txtSize = pDC->GetTextExtent(Act[cont]);
      }
      point = Coordenada(g5-9.5+10*cont,p7a);
      point.y -= txtSize.cy;
      pDC->TextOut(point.x,point.y,Act[cont]);
    }
    pDC->SelectObject(pOldFuente);
  }
```

```
  // Marca el elemento al mover el rat\'{o}n
  void CTablaElementosCtrl::OnMouseMove(UINT nFlags, CPoint point)
  {
    if (m_ejesRect.PtInRect (point))
    {
      //Obtener la posici\'{o}n Real
      rx = PT2DBLX(point.x);
      ry = PT2DBLY(point.y);
      Movimiento = TRUE;
      InvalidateControl(m_ejesRect);
    } else
        SetCursor(AfxGetApp()->LoadStandardCursor(IDC_ARROW));
    COleControl::OnMouseMove(nFlags, point);
  }
```

```
  // Nos muestra un marco de color en el elemento sobre el que se situa el rat\'{o}n
  void CTablaElementosCtrl::DibujarMovRaton(CDC* pDC)
  {
    //Definimos sin linea
    CPen pen(PS_SOLID, 1, RGB(255, 0, 0));
    CPen* pOldPen = pDC->SelectObject(&pen);


    int CoorX, CoorY;
    CPoint pts[4];
    CoorX = ((int) (rx / 10)) * 10;
    if ((ry >=20) && (ry < 23))
      return;

    if (ry < 20)
      CoorY = ((int) (ry / 10)) * 10;
    else
    {
      CoorY = ((int) ((ry - 3.0000001) / 10)) * 10 + 3;
    }

    ;
    //Filtrado de zonas sin elementos
    if ((CoorY == 83) && (CoorX >= 10) && (CoorX < 170)
      || (CoorY == 73) && (CoorX >= 20) && (CoorX < 120)
      || (CoorY == 63) && (CoorX >= 20) && (CoorX < 120)
      || (CoorY == 10) && (CoorX >= 00) && (CoorX < 40)
      || (CoorY == 00) && (CoorX >= 00) && (CoorX < 40))
      return;

    pDC->MoveTo(Coordenada(CoorX, CoorY));
    pDC->LineTo(Coordenada(CoorX + 10, CoorY));
    pDC->LineTo(Coordenada(CoorX + 10, CoorY + 10));
    pDC->LineTo(Coordenada(CoorX, CoorY + 10));
    pDC->LineTo(Coordenada(CoorX, CoorY));
    pDC->SelectObject(&pOldPen);
  }
```

```
  //Devuelve el elemento seleccionado
  SHORT CTablaElementosCtrl::ObtenerElemento(void)
  {
```

```
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    int CoorX, CoorY;
    CoorX = ((int) (rx / 10)) * 10;
    if ((ry >=20) && (ry < 23))
      CoorY=100; //no existe
    if (ry < 20)
      CoorY = ((int) (ry / 10)) * 10;
    else
13  {
      CoorY = ((int) ((ry - 3.0000001) / 10)) * 10 + 3;
    }

    switch ((int)CoorX)
    {
      case g0:
        switch ((int)CoorY)
        {
          case p1:
23          m_Elemento = 1;
            break;
          case p2:
            m_Elemento = 3;
            break;
          case p3:
            m_Elemento = 11;
            break;
          case p4:
            m_Elemento = 19;
33          break;
          case p5:
            m_Elemento = 37;
            break;
          case p6:
            m_Elemento = 55;
            break;
          case p7:
            m_Elemento = 87;
            break;
43        default:
            m_Elemento = -1;
            break;
        }
        break;
  {...}
          case g17:
        switch ((int)CoorY)
        {
          case p1:
53          m_Elemento = 2;
            break;
          case p2:
            m_Elemento = 10;
            break;
          case p3:
            m_Elemento = 18;
            break;
          case p4:
            m_Elemento = 36;
63          break;
          case p5:
            m_Elemento = 54;
            break;
          case p6:
            m_Elemento = 86;
            break;
```

```
        case p7:
          m_Elemento = 118;
          break;
        case p6a:
          m_Elemento = 71;
          break;
        case p7a:
          m_Elemento = 103;
          break;
        default:
          m_Elemento = -1;
          break;
      }
      break;
      default:
        m_Elemento = -1;
        break;
  }
  //Si el valor devuelto es -1 entonces no hay nada seleccionado
  return m_Elemento;
}
```

```
// Crea el fichero de nombres de elementos si no existe
int CTablaElementosCtrl::CrearFichero(LPCSTR Ruta)
{
  FILE* Fichero;
  if ((Fichero = fopen(Ruta,"w"))==NULL)
  {
    //Cambiar
    char temp[300];

    CString array[] = {"No se ha podido crear el fichero de nombres de Elementos"
      ,"Do not create the Elements names File"};

    sprintf(temp, "%s",array[m_Idioma]);
        MessageBox(temp);
    return 1;
  }
  if (m_Idioma==0)
  {
    char nombres1[] = "Hidr\'{o}geno,Litio,Sodio,Potasio,Rubidio,Cesio,Francio;\n";
    char nombres2[] = "Berilio,Magnesio,Calcio,Estroncio,Bario,Radio;\n";
    char nombres3[] = "Escandio,Itrio,Lantano,Actinio;\n";
    char nombres4[] = "Titanio,Circonio,Hafnio,Rutherfordio;\n";
    char nombres5[] = "Vanadio,Nicobio,Tantalio,Dubnio;\n";
    char nombres6[] = "Cromo,Molibdeno,Wolframio,Seaborgio;\n";
    char nombres7[] = "Manganeso,Tecnecio,Renio,Bohrio;\n";
    char nombres8[] = "Hierro,Rutenio,Osmio,Hassio;\n";
    char nombres9[] = "Cobalto,Rodio,Irdio,Meitnerio;\n";
    char nombres10[] = "Niquel,Paladio,Platino,Dannstadio;\n";
    char nombres11[] = "Cobre,Plata,Oro,Roetgenio;\n";
    char nombres12[] = "Zinc,Cadmio,Mercurio,Uub;\n";
    char nombres13[] = "Boro,Aluminio,Galio,Indio,Talio,Uut;\n";
    char nombres14[] = "Carbono,Silicio,Germanio,Esta\~{n}o,Plomo,Uuq;\n";
    char nombres15[] = "Nitr\'{o}geno,F\'{o}sforo,Ars\'{e}nico,Antimonio,Bismuto,Uup;\n";
    char nombres16[] = "Ox\'{\i}geno,Azufre,Selenio,Teluro,Polonio,Uuh;\n";
    char nombres17[] = "Fluor,Cloro,Bromo,Yodo,Astato,Uus;\n";
    char nombres18[] = "Helio,Ne\'{o}n,Arg\'{o}n,Kript\'{o}n,Xen\'{o}n,Rad\'{o}n,Uuo;\n";
    char nombres6a[] = "Cerio,Praseodimio,Neodimio,Promecio,Samario,Europio,Gadolinio,"
            "Terbio,Disprosio,Holmio,Erbio,Tulio,Iterbio,Lutecio;\n";
    char nombres7a[] = "Torio,Proctactinio,Uranio,Neptunio,Plutonio,Amercio,Curio,"
            "Berquelio,Californio,Einstenio,Fermio,Mendelevio,Nobelio,Laurencio;";
    fprintf(Fichero,"%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s",
      nombres1, nombres2, nombres3, nombres4, nombres5, nombres6,
      nombres7, nombres8, nombres9, nombres10, nombres11, nombres12 ,nombres13,
      nombres14, nombres15, nombres16, nombres17, nombres18, nombres6a, nombres7a);
```

```
    }
    else
    {
      char nombres1i[] = "Hydrogen,Lithium,Sodium,Potassium,Rubidium,Cesium,Francium;\n";
      char nombres2i[] = "Beryllium,Magnesium,Calcium,Strontium,Barium,Radium;\n";
      char nombres3i[] = "Scandium,Itrium,Lanthanum,Actinium;\n";
      char nombres4i[] = "Titanium,Zirconium,Hafnium,Rutherfordium;\n";
      char nombres5i[] = "Vanadium,Nicobium,Tantalum,Dubnium;\n";
      char nombres6i[] = "Chromium,Molybdenum,Tungsten,Seaborgium;\n";
      char nombres7i[] = "Manganese,Technecium,Rhenium,Bohrium;\n";
      char nombres8i[] = "Iron,Rutenium,Osmium,Hassium;\n";
      char nombres9i[] = "Cobalt,Rhodium,Irdium,Meitnerium;\n";
      char nombres10i[] = "Nickel,Palladium,Platinum,Dannstadium;\n";
      char nombres11i[] = "Copper,Silver,Gold,Roetgenium;\n";
      char nombres12i[] = "Zinc,Cadmium,Mercury,Uub;\n";
      char nombres13i[] = "Boron,Aluminum,Gallium,Indium,Thallium,Uut;\n";
      char nombres14i[] = "Carbon,Silicon,Germanium,Tin,Lead,Uuq;\n";
      char nombres15i[] = "Nitrogen,Phosphourus,Arsenic,Antimony,Bismuth,Uup;\n";
      char nombres16i[] = "Oxigen,Sulfur,Selenium,Tellurium,Polonium,Uuh;\n";
      char nombres17i[] = "Fluoride,Clorine,Bromine,Iodine,Astatine,Uus;\n";
      char nombres18i[] = "Helium,Neon,Argon,Krypton,Xenon,Radon,Uuo;\n";
      char nombres6ai[] = "Cerium,Praseodymium,Neodymium,Promethium,Samarium,Europium,"
              "Gadolinium,Terbium,Dysprosium,Holmium,Erbium,Thullum,Ytterbium,Lutetium;\n";
      char nombres7ai[] = "Thorium,Proctactinium,Uranium,Neptunium,Plutonium,"
              "Amercium,Curium,Berkelium,Californium,Einstenium,Fermium,Mendelevium,"
              "Nobelium,Laurencium;\n";

      fprintf(Fichero,"%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s",
        nombres1i, nombres2i, nombres3i, nombres4i, nombres5i, nombres6i,
        nombres7i, nombres8i, nombres9i, nombres10i, nombres11i, nombres12i ,nombres13i,
        nombres14i, nombres15i, nombres16i, nombres17i, nombres18i, nombres6ai, nombres7ai);
    }

    fclose(Fichero);

    return 0;
  }
```

B.4. iisgui.exe

B.4.1. Visualización en Tres Dimensiones

Interfaz Gráfica

```
////////////////////////////////
//Se define la clase CGraphView3D
////////////////////////////////
```

```
CGraphView3D::CGraphView3D()
  : CFormView(CGraphView3D::IDD)
{
  iniciado=0;

  CPreferences *prefs = &theApp.prefs;
  int m_iLang = prefs->GetLangx();
  switch (m_iLang)
  {
    case LANG_ENG:
      desp_Idioma = 2000; //maximo id 65000
    break;
```

```
      case LANG_SPA:
      default:
        desp_Idioma = 0;
      break;
  }
}
```

```
// Controladores de mensajes de CGraphView3D
void CGraphView3D::Inicia(char * NameIn)
{
  FILE *IF2;
  IF2 = fopen(NameIn,"r");

  if(!IF2)
  {
    char temp[300];
    sprintf(temp,"%s %s",GetResString(IDS_ERROR_FICHERO), NameIn);
    MessageBox(temp, NULL, MB_ICONERROR | MB_OK);
    exit(-1);
  }

  maximo2=0.0;
  minimo2=1e50;

  fread(&N_X,sizeof(int),1,IF2);
  fread(&N_Y,sizeof(int),1,IF2);
  fread(&N_Z,sizeof(int),1,IF2);
  fread(&XMin,sizeof(double),1,IF2);
  fread(&XMax,sizeof(double),1,IF2);
  fread(&YMin,sizeof(double),1,IF2);
  fread(&YMax,sizeof(double),1,IF2);
  fread(&ZMin,sizeof(double),1,IF2);
  fread(&ZMax,sizeof(double),1,IF2);

  //Lectura de los datos del archivo, se guardan en formato lineal en DATOS
  DATOS = (double*)malloc((N_X*N_Y*N_Z+1)*sizeof(double));
  if (!DATOS)
  {
    MessageBox(GetResString(IDS_ERROR_MEMORIA),GetResString(IDS_ERROR_MEMORIA_T),
            MB_ICONERROR | MB_OK);
    exit(-3);
  }
  fread(DATOS,sizeof(double),N_X*N_Y*N_Z,IF2);
  fclose(IF2);
}
```

```
void CGraphView3D::Escribe()
{
  FILE *IF2;
  IF2 = fopen("archTemp.tmp","w+t");

  fprintf(IF2, "# Datos 3D\n");

  //Datos del tama\~{n}o la tabla
  fprintf(IF2, "# n X: %d\n",N_X);
  fprintf(IF2, "# n Y: %d\n",N_Y);
  fprintf(IF2, "# n Z: %d\n",N_Z);

  //maximos y minimos
  fprintf(IF2, "# Minimo X: %e\n",XMin);
  fprintf(IF2, "# Maximo X: %e\n",XMax);
  fprintf(IF2, "# Minimo Y: %e\n",YMin);
  fprintf(IF2, "# Maximo Y: %e\n",YMax);
  fprintf(IF2, "# Minimo Z: %e\n",ZMin);
  fprintf(IF2, "# Maximo Z: %e\n",ZMax);
```

```
  //ahora escribimos todos los datos por capas
  for (int i=0; i<N_X; i++)
  {
    fprintf(IF2, "\n#␣Plano␣XY:␣Z␣=␣%d\n",i+1);
    for (int j=0; j<N_Y; j++)
    {
      for (int k=0; k<N_Z; k++)
        fprintf(IF2, "%e\t",DATOS[j*N_Y*N_Z+k*N_Z+i]);
      fprintf(IF2, "\n");
    }
  }
  fclose(IF2);
}
```

```
void CGraphView3D::ExtraeLoncha( int iPlano )
{
  //reserva de memoria para una "loncha" de datos
  if(Archivo1.Plano==0)
    DATOS2D=(double*)malloc(N_X*N_Y*sizeof(double));
    if(Archivo1.Plano==1)
    DATOS2D=(double*)malloc(N_X*N_Z*sizeof(double));
    if(Archivo1.Plano==2)
    DATOS2D=(double*)malloc(N_Y*N_Z*sizeof(double));
    if(!DATOS2D)
  {
    MessageBox(GetResString(IDS_ERROR_MEMORIA2),GetResString(IDS_ERROR_MEMORIA_T),
            MB_ICONERROR | MB_OK);
    exit (-3);
  }

  if(Archivo1.Plano==0)
    {
    N_A=N_X;
    N_B=N_Y;
    N_C=N_Z;
    AMin=XMin;
    AMax=XMax;
    BMin=YMin;
    BMax=YMax;
    CMin=ZMin;
    CMax=ZMax;
  }
    if(Archivo1.Plano==1)
    {
    N_A=N_X;
    N_B=N_Z;
    N_C=N_Y;
    AMin=XMin;
    AMax=XMax;
    BMin=ZMin;
    BMax=ZMax;
    CMin=YMin;
    CMax=YMax;
  }
    if(Archivo1.Plano==2)
    {
    N_A=N_Y;
    N_B=N_Z;
    N_C=N_X;
    AMin=YMin;
    AMax=YMax;
    BMin=ZMin;
    BMax=ZMax;
    CMin=XMin;
    CMax=XMax;
  }
```

```
    maximo=-1e50;
    minimo=1e50;

    for(int i=0;i<N_A;i++)
    for(int j=0;j<N_B;j++)
    {
      if(Archivo1.Acumulacion)
      {
        //guardado de datos acumulativos de todas las lonchas
        DATOS2D[i*N_B+j]=0.0;
        if(Archivo1.Plano==0)
          for(int k=0;k<N_C;k++)
            DATOS2D[i*N_B+j]+=DATOS[i*N_Y*N_Z+j*N_Z+k];
        if(Archivo1.Plano==1)
          for(int k=0;k<N_C;k++)
            DATOS2D[i*N_B+j]+=DATOS[i*N_Y*N_Z+k*N_Z+j];
        if(Archivo1.Plano==2)
          for(int k=0;k<N_C;k++)
            DATOS2D[i*N_B+j]+=DATOS[k*N_Y*N_Z+i*N_Z+j];
      }
      else
      {
        //guardado de datos de una loncha
        if(Archivo1.Plano==0)
        {
          DATOS2D[i*N_B+j] = DATOS[i*N_Y*N_Z+j*N_Z+iPlano];
        }
        if(Archivo1.Plano==1)
          DATOS2D[i*N_B+j] = DATOS[i*N_Y*N_Z+iPlano*N_Z+j];
        if(Archivo1.Plano==2)
          DATOS2D[i*N_B+j] = DATOS[iPlano*N_Y*N_Z+i*N_Z+j];
      }
      if(DATOS2D[i*N_B+j]>maximo)
        maximo=DATOS2D[i*N_B+j];
      if(DATOS2D[i*N_B+j]<minimo)
        minimo=DATOS2D[i*N_B+j];
    }
}
```

```
void CGraphView3D::Pinta()
{
  double x,y,z;

  if(Archivo1.Plano==0)
    {
    N_A=N_X;
    N_B=N_Y;
    N_C=N_Z;
    AMin=XMin;
    AMax=XMax;
    BMin=YMin;
    BMax=YMax;
    CMin=ZMin;
    CMax=ZMax;
  }
    if(Archivo1.Plano==1)
    {
    N_A=N_X;
    N_B=N_Z;
    N_C=N_Y;
    AMin=XMin;
    AMax=XMax;
    BMin=ZMin;
    BMax=ZMax;
    CMin=YMin;
```

```
      CMax=YMax;
    }
29    if(Archivo1.Plano==2)
      {
      N_A=N_Y;
      N_B=N_Z;
      N_C=N_X;
      AMin=YMin;
      AMax=YMax;
      BMin=ZMin;
      BMax=ZMax;
      CMin=XMin;
39    CMax=XMax;
    }

    m_Graph3D.ClearGraph();

    m_Graph3D.AddElement();
    m_Graph3D.SetTrackMode(2);  //Modo Rotar

    //Colores y tama\~{n}os de lineas y puntos
    m_Graph3D.ElementSurfaceColor(0, Archivo1.ElemColorSuperficie);
49  m_Graph3D.put_ElementLineColor(0, Archivo1.ElemColorLinea);
    m_Graph3D.put_ElementPointColor(0, Archivo1.ElemColorPunto);
    m_Graph3D.put_ElementLineWidth(0,(float)Archivo1.ElemAnchura);
    m_Graph3D.put_ElementPointSize(0,(float)Archivo1.ElemTamano);
    m_Graph3D.SetElementType(0, (short)Archivo1.GrafEstilo);

    x=y=z=0;
    //double resX = (AMax - AMin)/N_B;
    //double resZ = (BMax - BMin)/N_B;
    if (Archivo1.GrafYLogaritmico==1)
59  {
      //logaritmico (datos y ejes)
      x=AMin;
      for (int i = 0; i < N_A; i++)
      {

        z = BMin;
        for (int j = 0; j < N_B; j++)
        {
          y = log10(DATOS2D[i*N_B+j]);
69        //Cosa del autorango
          if (Archivo1.GrafAutoRango)
          {
            if (y <-10)
            {
              //y = log10 (1e-10);
              minimo = 1e-10;
            }
          }
          else
79        {
            if (y < log10(Archivo1.GrafMinimo))
            {
              //y = log10 (Archivo1.GrafMinimo);
              minimo = Archivo1.GrafMinimo;
            }
          }
          m_Graph3D.PlotXYZ (x, y, z, 0);
          z = z + (BMax - BMin)/(N_B-1);
        }
89      x = x + (AMax - AMin)/(N_A-1);
      }
    }
    else
```

```
{
  //lineal (datos y ejes)

  x=AMin;
  for (int i = 0; i < N_A; i++)
  {

    z = BMin;
    for (int j = 0; j < N_B; j++)
    {
      //x = i;
      //z = j;

      y = DATOS2D[i*N_B+j];
      if (y<0.0)
      {
        y=0.0;
        minimo=0;
      }
      m_Graph3D.PlotXYZ (x, y, z, 0);
      z = z + (BMax-BMin)/(N_B-1);
    }
    x = x + (AMax-AMin)/(N_A-1);
  }
}

if (Archivo1.GrafYLogaritmico==1)
{
  //determinaci\'{o}n del rango de pintado;
  double ExpMax;
  double ExpMin;

  if ((ceil(log10(maximo)) - floor(log10(minimo))) < 5)
    m_Graph3D.put_YGridNumber((short)(ceil(log10(maximo)) - floor(log10(minimo))));
  else
    m_Graph3D.put_YGridNumber(5);
  //cosa del Autorango
  if (Archivo1.GrafAutoRango)
  {
    ExpMax = pow(10.0, ceil(log10(maximo)));
    ExpMin = pow(10.0, floor(log10(minimo)) - (m_Graph3D.get_YGridNumber()-1));
  }
  else
  {
    //Cielo
    ExpMax = pow(10.0, ceil(log10(Archivo1.GrafMaximo)));
    //Suelo
    ExpMin = pow(10.0, floor(log10(Archivo1.GrafMinimo)) -
              (m_Graph3D.get_YGridNumber()-1));
  }
  if (ExpMin==0.0)
    ExpMin=1e-10;

  int div = (int)(log10(ExpMax)-log10(ExpMin));
  while (div%m_Graph3D.get_YGridNumber() != 0)
  {
    ExpMin=ExpMin*10;
    div = (int)(log10(ExpMax)-log10(ExpMin));
  }
  Archivo1.GrafMaximo=ExpMax;
  Archivo1.GrafMinimo=ExpMin;
  m_Graph3D.SetRange (floor(AMin), ceil(AMax), log10(ExpMin), log10(ExpMax),
          floor(BMin), ceil(BMax), Archivo1.GrafYLogaritmico);
}
else
{
```

```
    m_Graph3D.put_YGridNumber(5);
    if (Archivo1.GrafAutoRango)
    {
      Archivo1.GrafMaximo=maximo*1.001;
      Archivo1.GrafMinimo=minimo;
      m_Graph3D.SetRange (floor(AMin), ceil(AMax), minimo, maximo*1.001,
                floor(BMin) , ceil(BMax), Archivo1.GrafYLogaritmico);
    }
    else
      m_Graph3D.SetRange (floor(AMin), ceil(AMax), Archivo1.GrafMinimo,
                Archivo1.GrafMaximo/**1.001*/, floor(BMin) , ceil(BMax),
                Archivo1.GrafYLogaritmico);
  }
}
```

```
void CGraphView3D::OnBnClickedButton1()
{
  Archivo1.Grabar(ArchConf);
  ArchTemp=Archivo1;

  CGraph* grp = GetDocument()->GetGraph();
  CSimulation *sim = GetDocument()->GetSim();
  ASSERT_VALID(grp);
  ASSERT_VALID(sim);

  Graph3DProp.ArchivoPath(ArchConf,sim, grp);
  INT_PTR nRet = -1;

  Graph3DProp.desp_Idioma=desp_Idioma;
  Graph3DProp.SetDlgItemText(IDD_GRAPH3DPROP, "Hola");
  nRet = Graph3DProp.DoModal();
  switch (nRet)
  {
    case IDOK:
      Archivo1 = ArchTemp;
      Archivo1.Grabar(ArchConf);
      break;
    case IDCANCEL:
      ArchTemp = Archivo1 ;
      Archivo1.Grabar(ArchConf);
      break;
  }
}
```

```
void CGraphView3D::OnInitialUpdate()
{
  CFormView::OnInitialUpdate();

  GetParentFrame()->RecalcLayout();
  ResizeParentToFit(FALSE);
  if (!theApp.prefs.IsEnabledScrollGraph())
    //no desplazar la vista
    SetScaleToFitSize(GetTotalSize()); //disable the scroll bars

  if (GetParentFrame() && GetParentFrame()->GetSafeHwnd())
  {
    //save original frame and client rects
    GetParentFrame()->GetWindowRect(m_FrameRect);
    //set icon
    GetParentFrame()->SetIcon(theApp.LoadIcon(IDI_VIEW_GRP),FALSE);
  }

  int m_iLang = theApp.prefs.GetLangx();
  //En este caso hay que dejarlo porque manda el valor
  //pero no el desplazamiento
  //Idioma en espa\~{n}ol
```

```
    //m_Graph3D.Idioma(1);
    //Idioma en ingles
    //m_Graph3D.Idioma(0);
    m_Graph3D.Idioma(m_iLang);

    GetClientRect(m_ClientRect);

    CSpinButtonCtrl* pSpin1;
    pSpin1 = (CSpinButtonCtrl*)GetDlgItem(IDC_SCAPA);
32  // Indicamos el control asociado al Spin Control
    pSpin1->SetBuddy (GetDlgItem(IDC_ECapa));
    // Fijamos el rango del Spin Control
    pSpin1->SetRange(0,49);

    SetDlgItemText(IDC_STCAPA,GetResString(IDS_PROP_LAYER));
    SetDlgItemText(IDC_CHACUMULACION,GetResString(IDS_3D_ACUMULACION));
    SetDlgItemText(IDC_BUTTON1,GetResString(IDS_PROPERTY));
    SetDlgItemText(IDC_SVISTA,GetResString(IDS_3D_VISTA));
    SetDlgItemText(IDC_BORIGINAL,GetResString(IDS_3D_ORIGINAL)+"␣(1)");
42  SetDlgItemText(IDC_BSUPERIOR,GetResString(IDS_3D_SUPERIOR)+"␣(2)");
    SetDlgItemText(IDC_BLATERAL,GetResString(IDS_3D_LATERAL)+"␣(3)");
    SetDlgItemText(IDC_SPlano,GetResString(IDS_3D_PLANO));
    SetDlgItemText(IDC_CEJEXY,GetResString(IDS_3D_PLANO)+"␣XY");
    SetDlgItemText(IDC_CEJEXZ,GetResString(IDS_3D_PLANO)+"␣XZ");
    SetDlgItemText(IDC_CEJEYZ,GetResString(IDS_3D_PLANO)+"␣YZ");
    SetDlgItemText(IDC_BARRIBA,GetResString(IDS_3D_ARRIBA)+"␣(W)");
    SetDlgItemText(IDC_BDERECHA,GetResString(IDS_3D_DERECHA)+"␣(D)");
    SetDlgItemText(IDC_BABAJO,GetResString(IDS_3D_ABAJO)+"␣(S)");
    SetDlgItemText(IDC_BIZQUIERDA,GetResString(IDS_3D_IZQUIERDA)+"␣(A)");
52
    //Hace la llamada para que lea la estructura;
    //Pasarle el nombre de la ventana

    CGraph* grp = GetDocument()->GetGraph();
    CSimulation *sim = GetDocument()->GetSim();
    ASSERT_VALID(grp);
    ASSERT_VALID(sim);

    char temp[200];
62  char dirTemp[200];
    LPCTSTR buffArchivo=sim->GetFilePath();

    for (unsigned int i=0; i<(strlen(buffArchivo)-4); i++)
      temp[i]=buffArchivo[i];

    temp[strlen(buffArchivo)-4]=0;
    sprintf(temp,"%s\\%s.txt",temp,grp->GetName());
    sprintf(ArchConf, temp);
    Archivo1.Recuperar(ArchConf);
72  ArchTemp=Archivo1;

    SetDlgItemInt(IDC_ECapa,Archivo1.NumCapa);

    bool TiempoX, TiempoY, TiempoZ;
    if (Archivo1.FormTipoX!=3)
      TiempoX=false;
    else
      TiempoX=true;

82  if (Archivo1.FormTipoY!=3)
      TiempoY=false;
    else
      TiempoY=true;

    if (Archivo1.FormTipoZ!=3)
      TiempoZ=false;
```

```
    else
      TiempoZ=true;

92  switch (Archivo1.Plano)
    {
      case 0:
        EjeXZ.SetCheck(0);
        EjeYZ.SetCheck(0);
        EjeXY.SetCheck(1);
        //Titulos
        m_Graph3D.put_XLabel(Archivo1.GrafEjeX);
        m_Graph3D.put_YLabel(Archivo1.GrafEjeZ);
        m_Graph3D.put_ZLabel(Archivo1.GrafEjeY);
102     m_Graph3D.TipoEjes(Archivo1.FormFormatoX,
            Archivo1.FormFormatoZ, Archivo1.FormFormatoY,
            TiempoX, TiempoZ, TiempoY);
        break;
      case 1:
        EjeXZ.SetCheck(1);
        EjeYZ.SetCheck(0);
        EjeXY.SetCheck(0);
        //Titulos
        m_Graph3D.put_XLabel(Archivo1.GrafEjeX);
112     m_Graph3D.put_YLabel(Archivo1.GrafEjeY);
        m_Graph3D.put_ZLabel(Archivo1.GrafEjeZ);
        m_Graph3D.TipoEjes(Archivo1.FormFormatoX,
            Archivo1.FormFormatoY, Archivo1.FormFormatoZ,
            TiempoX, TiempoY, TiempoZ);
        break;
      case 2:
        EjeXZ.SetCheck(0);
        EjeYZ.SetCheck(1);
        EjeXY.SetCheck(0);
122     //Titulos
        m_Graph3D.put_XLabel(Archivo1.GrafEjeY);
        m_Graph3D.put_YLabel(Archivo1.GrafEjeX);
        m_Graph3D.put_ZLabel(Archivo1.GrafEjeZ);
        m_Graph3D.TipoEjes(Archivo1.FormFormatoY,
            Archivo1.FormFormatoX, Archivo1.FormFormatoZ,
            TiempoY, TiempoX, TiempoZ);
        break;
    }
    if (Archivo1.Acumulacion)
132   m_bAcumulacion.SetCheck(1);
    else
      m_bAcumulacion.SetCheck(0);
    m_Graph3D.put_BackColor(Archivo1.GrafColorFondo); //Color de fondo

    iniciado=1;
    OnSize(0,1,1);

    //Pintado del archivo
    if (StrCmp(Nom_Arch, "hola")!=0)
142 {
      for (int i=0; i<200; i++)
        Archivo1.Direccion[i] = Nom_Arch[i];

      m_Graph3D.SetCaption (Archivo1.GrafTitulo);

      sprintf(Archivo1.GrafTitulo, grp->GetName());
      m_Graph3D.SetCaption (Archivo1.GrafTitulo);

      char temp[200];
152   sprintf(temp, "Graphic3D␣%s",Archivo1.GrafTitulo);
      grp->SetName(temp);
      LPCTSTR buffArchivo=sim->GetFilePath();
```

```
      for (unsigned int i=0; i<(strlen(buffArchivo)-4); i++)
        temp[i]=buffArchivo[i];
      temp[strlen(buffArchivo)-4]=0;

      //strcpy(dirTemp, temp);

      sprintf(temp,"%s\\%s.txt",temp,grp->GetName());
      rename(ArchConf, temp);
      sprintf(ArchConf, temp);

      GetDocument()->SetTitle(sim->m_strSimName + "␣-␣" + grp->GetName());
      //m_GraphCtrl.SetCaption(strBuff); //Nombre de la gr\'{a}fica
      AfxGetMainWnd()->PostMessage(IIS_SIM_REFRESHGRAPH,(WPARAM)sim);
      ArchTemp=Archivo1;
      Archivo1.Grabar(ArchConf);
    }

    for (unsigned int i=0; i<(strlen(buffArchivo)-4); i++)
      dirTemp[i]=buffArchivo[i];
    dirTemp[strlen(buffArchivo)-4]=0;

    sprintf(dirTemp,"%s\\%s", dirTemp, Archivo1.Direccion);

    Inicia(dirTemp);
    //Divisiones
    m_Graph3D.put_XGridNumber(5); //para la base
    m_Graph3D.put_ZGridNumber(5); //para la base
    m_Graph3D.put_YGridNumber(5); //Divisiones en altura

    m_Graph3D.SetCaption (Archivo1.GrafTitulo);

    //if (m_Graph3D.get_Projection() == 0)  //0 = Conica
    m_Graph3D.put_Projection(1);    //1 = Isometrico

    ExtraeLoncha(Archivo1.NumCapa);

    Pinta();

    m_Graph3D.put_AnguloX(0,(float)Archivo1.AnguloX);
    m_Graph3D.put_AnguloY(0,(float)Archivo1.AnguloY);

  }
```

```
  void CGraphView3D::OnSize(UINT nType, int cx, int cy)
  {
    CFormView::OnSize(nType, cx, cy);

    if (nType != SIZE_MINIMIZED && cx != 0 && cy != 0 &&
          (!theApp.prefs.IsEnabledScrollGraph())&& iniciado)
    {
      CRect clientRect;
      GetClientRect(clientRect);

      //Recolocar bot\'{o}n Derecha
      CRect btnDer;
      CWnd* wnd = GetDlgItem(IDC_BDERECHA);
      wnd->GetWindowRect(btnDer);
      ScreenToClient(btnDer);
      int iWidth = btnDer.Width();
      int iHeight = btnDer.Height();
      btnDer.right = clientRect.right - 5;
      btnDer.left = btnDer.right - iWidth;
      wnd->MoveWindow(btnDer);

      //Recolocar bot\'{o}n Izquierda
```

```
23      CRect btnIzq;
        wnd = GetDlgItem(IDC_BIZQUIERDA);
        wnd->GetWindowRect(btnIzq);
        ScreenToClient(btnIzq);
        iWidth = btnIzq.Width();
        iHeight = btnIzq.Height();
        btnIzq.right = btnDer.left - 2;
        btnIzq.left = btnIzq.right - iWidth;
        wnd->MoveWindow(btnIzq);

33      //Recolocar bot\'{o}n Arriba
        CRect btnArr;
        wnd = GetDlgItem(IDC_BARRIBA);
        wnd->GetWindowRect(btnArr);
        ScreenToClient(btnArr);
        iWidth = btnArr.Width();
        iHeight = btnArr.Height();
        btnArr.right = (btnDer.right + btnIzq.left)/2 + iWidth / 2;
        btnArr.left = btnArr.right - iWidth;
        wnd->MoveWindow(btnArr);
43
        //Recolocar bot\'{o}n Abajo
        CRect btnAba;
        wnd = GetDlgItem(IDC_BABAJO);
        wnd->GetWindowRect(btnAba);
        ScreenToClient(btnAba);
        iWidth = btnAba.Width();
        iHeight = btnAba.Height();
        btnAba.right = btnArr.right;
        btnAba.left = btnAba.right - iWidth;
53      wnd->MoveWindow(btnAba);

        //Recolocar bot\'{o}n Propiedades
        CRect btnProp;
        wnd = GetDlgItem(IDC_BUTTON1);
        wnd->GetWindowRect(btnProp);
        ScreenToClient(btnProp);
        iWidth = btnProp.Width();
        iHeight = btnProp.Height();
        btnProp.right = (btnDer.right + btnIzq.left)/2 + iWidth / 2;
63      btnProp.left = btnProp.right - iWidth;
        wnd->MoveWindow(btnProp);

        //Recolocar bot\'{o}n Acumulacion
        CRect btnAcum;
        wnd = GetDlgItem(IDC_CHACUMULACION);
        wnd->GetWindowRect(btnAcum);
        ScreenToClient(btnAcum);
        iWidth = btnAcum.Width();
        iHeight = btnAcum.Height();
73      btnAcum.right = (btnDer.right + btnIzq.left)/2 + iWidth / 2;
        btnAcum.left = btnProp.right - iWidth;
        wnd->MoveWindow(btnAcum);

        //Recolocar SpinButton
        CRect SPButton;
        wnd = GetDlgItem(IDC_SCAPA);
        wnd->GetWindowRect(SPButton);
        ScreenToClient(SPButton);
        iWidth = SPButton.Width();
83      iHeight = SPButton.Height();
        SPButton.right = btnAcum.right;
        SPButton. left= SPButton.right - iWidth;
        wnd->MoveWindow(SPButton);

        //Recolocar EditControl "Capa"
```

```
CRect EditCapa;
wnd = GetDlgItem(IDC_ECapa);
wnd->GetWindowRect(EditCapa);
ScreenToClient(EditCapa);
iWidth = EditCapa.Width();
iHeight = EditCapa.Height();
EditCapa.right = SPButton.left -3;
EditCapa.left = EditCapa.right - iWidth;
wnd->MoveWindow(EditCapa);

//Recolocar StaticText "Capa"
CRect TxtCapa;
wnd = GetDlgItem(IDC_STCAPA);
wnd->GetWindowRect(TxtCapa);
ScreenToClient(TxtCapa);
iWidth = TxtCapa.Width();
iHeight = TxtCapa.Height();
TxtCapa.left = btnAcum.left;
TxtCapa.right =TxtCapa.left + iWidth;
wnd->MoveWindow(TxtCapa);

//Recolocar GroupBox "Vista"
CRect GBVista;
wnd = GetDlgItem(IDC_SVISTA);
wnd->GetWindowRect(GBVista);
ScreenToClient(GBVista);
iWidth = GBVista.Width();
iHeight = GBVista.Height();
GBVista.right = (btnDer.right + btnIzq.left)/2 + iWidth / 2;
GBVista.left =GBVista.right - iWidth;
wnd->MoveWindow(GBVista);

//Recolocar bot\'{o}n Original
CRect btnOrig;
wnd = GetDlgItem(IDC_BORIGINAL);
wnd->GetWindowRect(btnOrig);
ScreenToClient(btnOrig);
iWidth = btnOrig.Width();
iHeight = btnOrig.Height();
btnOrig.left = GBVista.left + 7;
btnOrig.right = btnOrig.left + iWidth;
wnd->MoveWindow(btnOrig);

//Recolocar bot\'{o}n Lateral
CRect btnLat;
wnd = GetDlgItem(IDC_BLATERAL);
wnd->GetWindowRect(btnLat);
ScreenToClient(btnLat);
iWidth = btnLat.Width();
iHeight = btnLat.Height();
btnLat.left = GBVista.left + 7;
btnLat.right = btnLat.left + iWidth;
wnd->MoveWindow(btnLat);

//Recolocar bot\'{o}n Superior
CRect btnSup;
wnd = GetDlgItem(IDC_BSUPERIOR);
wnd->GetWindowRect(btnSup);
ScreenToClient(btnSup);
iWidth = btnSup.Width();
iHeight = btnSup.Height();
btnSup.left = GBVista.left + 7;
btnSup.right = btnSup.left + iWidth;
wnd->MoveWindow(btnSup);

//Recolocar GroupBox "Plano"
```

```
    CRect GBPlano;
    wnd = GetDlgItem(IDC_SPlano);
    wnd->GetWindowRect(GBPlano);
    ScreenToClient(GBPlano);
    iWidth = GBPlano.Width();
    iHeight = GBPlano.Height();
    GBPlano.right = GBVista.right ;
    GBPlano.left =GBPlano.right - iWidth;
    wnd->MoveWindow(GBPlano);

    //Recolocar CheckBox "Plano XY"
    CRect CBEjeXY;
    wnd = GetDlgItem(IDC_CEJEXY);
    wnd->GetWindowRect(CBEjeXY);
    ScreenToClient(CBEjeXY);
    iWidth = CBEjeXY.Width();
    iHeight = CBEjeXY.Height();
    CBEjeXY.left = GBPlano.left + 7;
    CBEjeXY.right = CBEjeXY.left + iWidth;
    wnd->MoveWindow(CBEjeXY);

    //Recolocar CheckBox "Plano XZ"
    CRect CBEjeXZ;
    wnd = GetDlgItem(IDC_CEJEXZ);
    wnd->GetWindowRect(CBEjeXZ);
    ScreenToClient(CBEjeXZ);
    iWidth = CBEjeXZ.Width();
    iHeight = CBEjeXZ.Height();
    CBEjeXZ.left = GBPlano.left + 7;
    CBEjeXZ.right = CBEjeXZ.left + iWidth;
    wnd->MoveWindow(CBEjeXZ);

    //Recolocar CheckBox "Plano YZ"
    CRect CBEjeYZ;
    wnd = GetDlgItem(IDC_CEJEYZ);
    wnd->GetWindowRect(CBEjeYZ);
    ScreenToClient(CBEjeYZ);
    iWidth = CBEjeYZ.Width();
    iHeight = CBEjeYZ.Height();
    CBEjeYZ.left = GBPlano.left + 7;
    CBEjeYZ.right = CBEjeYZ.left + iWidth;
    wnd->MoveWindow(CBEjeYZ);

    //Recolocar el objeto ActiveX
    CRect ObjActiveX;
    wnd = GetDlgItem(IDC_NTGRAPH3D);
    wnd->GetWindowRect(ObjActiveX);
    ScreenToClient(ObjActiveX);
    iWidth = ObjActiveX.Width();
    iHeight = ObjActiveX.Height();
    ObjActiveX.left = clientRect.left;
    ObjActiveX.right = btnIzq.left - 5;
    ObjActiveX.bottom = clientRect.bottom ;
    ObjActiveX.top = clientRect.top;
    wnd->MoveWindow(ObjActiveX);
  }

}
```

```
void CGraphView3D::OnGetMinMaxInfo(MINMAXINFO* lpMMI)
{
  if (!theApp.prefs.IsEnabledScrollGraph() && !m_FrameRect.IsRectNull())
  { //prevent the view from resizing below minimun size
    lpMMI->ptMinTrackSize.x = m_ClientRect.Width();
    lpMMI->ptMinTrackSize.y = m_ClientRect.Height();
  }
```

```
}
```

```
void CGraphView3D::OnBnClickedBoriginal()
{
  m_Graph3D.put_AnguloX(0, 30);
  m_Graph3D.put_AnguloY(0, -45);
  Archivo1.AnguloX = 30;
  Archivo1.AnguloY = -45;
  Archivo1.Grabar(ArchConf);
}
```

```
void CGraphView3D::OnBnClickedBsuperior()
{
  m_Graph3D.put_AnguloX(0, 90);
  m_Graph3D.put_AnguloY(0, 0);
  Archivo1.AnguloX = 90;
  Archivo1.AnguloY = 0;
  Archivo1.Grabar(ArchConf);
}
```

```
void CGraphView3D::OnBnClickedBlateral()
{
  m_Graph3D.put_AnguloX(0, 0);
  m_Graph3D.put_AnguloY(0, 0);
  Archivo1.AnguloX = 0;
  Archivo1.AnguloY = -0;
  Archivo1.Grabar(ArchConf);
}
```

```
void CGraphView3D::OnBnClickedBarriba()
{
  m_Graph3D.put_AnguloX(0, m_Graph3D.get_AnguloX(0)+1);
  Archivo1.AnguloX = (int)m_Graph3D.get_AnguloX(0)+1;
  Archivo1.Grabar(ArchConf);
}
```

```
void CGraphView3D::OnBnClickedBabajo()
{
  m_Graph3D.put_AnguloX(0, m_Graph3D.get_AnguloX(0)-1);
  Archivo1.AnguloX = (int)m_Graph3D.get_AnguloX(0)-1;
  Archivo1.Grabar(ArchConf);
}
```

```
void CGraphView3D::OnBnClickedBderecha()
{
  m_Graph3D.put_AnguloY(0, m_Graph3D.get_AnguloY(0)-1);
  Archivo1.AnguloY = (int)m_Graph3D.get_AnguloY(0)-1;
  Archivo1.Grabar(ArchConf);
}
```

```
void CGraphView3D::OnBnClickedBizquierda()
{
  m_Graph3D.put_AnguloY(0, m_Graph3D.get_AnguloY(0)+1);
  Archivo1.AnguloY = (int)m_Graph3D.get_AnguloY(0)+1;
  Archivo1.Grabar(ArchConf);
}
```

```
void CGraphView3D::OnBnClickedChacumulacion()
{
  if (Archivo1.Acumulacion==1)
    Archivo1.Acumulacion = 0;
```

```
  else
    Archivo1.Acumulacion = 1;
  ExtraeLoncha(Archivo1.NumCapa);
  Archivo1.Grabar(ArchConf);
  Pinta();
}
```

```
void CGraphView3D::OnBnClickedCejexy()
{
  if (!EjeXY.GetCheck())
  {
    EjeXY.SetCheck(1);
    return;
  }

  EjeXZ.SetCheck(0);
  EjeYZ.SetCheck(0);

  Archivo1.Plano = 0;
  //Archivo1.NumCapa = 0;
  Archivo1.Grabar(ArchConf);
  ExtraeLoncha(Archivo1.NumCapa);

  m_Graph3D.put_XLabel(Archivo1.GrafEjeX);
  m_Graph3D.put_YLabel(Archivo1.GrafEjeZ);
  m_Graph3D.put_ZLabel(Archivo1.GrafEjeY);

  Pinta();
}
```

```
void CGraphView3D::OnBnClickedCejexz()
{
  if (!EjeXZ.GetCheck())
  {
    EjeXZ.SetCheck(1);
    return;
  }
  EjeXY.SetCheck(0);
  EjeYZ.SetCheck(0);

  Archivo1.Plano = 1;
  //Archivo1.NumCapa = 0;
  Archivo1.Grabar(ArchConf);
  ExtraeLoncha(Archivo1.NumCapa);

  m_Graph3D.put_XLabel(Archivo1.GrafEjeX);
  m_Graph3D.put_YLabel(Archivo1.GrafEjeY);
  m_Graph3D.put_ZLabel(Archivo1.GrafEjeZ);

  Pinta();
}
```

```
void CGraphView3D::OnBnClickedCejeyz()
{
  if (!EjeYZ.GetCheck())
  {
    EjeYZ.SetCheck(1);
    return;
  }

  EjeXY.SetCheck(0);
  EjeXZ.SetCheck(0);

  Archivo1.Plano = 2;
  Archivo1.Grabar(ArchConf);
```

```
  //Archivo1.NumCapa = 0;
  ExtraeLoncha(Archivo1.NumCapa);

  m_Graph3D.put_XLabel(Archivo1.GrafEjeY);
  m_Graph3D.put_YLabel(Archivo1.GrafEjeX);
  m_Graph3D.put_ZLabel(Archivo1.GrafEjeZ);

  Pinta();
}
```

```
void CGraphView3D::OnDestroy()
{
  //Archivo1.Grabar(ArchConf);

  CFormView::OnDestroy();
}
```

```
void CGraphView3D::OnDeltaposScapa(NMHDR *pNMHDR, LRESULT *pResult)
{
  LPNMUPDOWN pNMUpDown = reinterpret_cast<LPNMUPDOWN>(pNMHDR);
  if (pNMUpDown->iDelta==1) //incrementa capa
    Archivo1.NumCapa++;
  else            //decrementa capa
    Archivo1.NumCapa--;

  //comprobaci\'{o}n de limites
  if (Archivo1.NumCapa<0)
    Archivo1.NumCapa = 49;
  if (Archivo1.NumCapa>49)
    Archivo1.NumCapa = 0;

  if (m_bAcumulacion.GetCheck())  //desactivamos acumulaci\'{o}n
  {
    m_bAcumulacion.SetCheck(0);
    Archivo1.Acumulacion=0;
  }
  Archivo1.Grabar(ArchConf);
  ExtraeLoncha(Archivo1.NumCapa);
  Pinta();

  *pResult = 0;
}
```

```
void CGraphView3D::OnSetFocus(CWnd* pOldWnd)
{
  CFormView::OnSetFocus(pOldWnd);
  if (iniciado)
  {
    Archivo1.Recuperar(ArchConf);
    ArchTemp=Archivo1;

    bool TiempoX, TiempoY, TiempoZ;
    if (Archivo1.FormTipoX!=3)
      TiempoX=false;
    else
      TiempoX=true;

    if (Archivo1.FormTipoY!=3)
      TiempoY=false;
    else
      TiempoY=true;

    if (Archivo1.FormTipoZ!=3)
      TiempoZ=false;
    else
```

```
      TiempoZ=true;

    //hacer un repintado con todos los datos
    switch (Archivo1.Plano)
    {
      case 0:
        //Titulos de ejes
        m_Graph3D.put_XLabel(Archivo1.GrafEjeX);
        m_Graph3D.put_YLabel(Archivo1.GrafEjeZ);
        m_Graph3D.put_ZLabel(Archivo1.GrafEjeY);
        m_Graph3D.TipoEjes(Archivo1.FormFormatoX,
          Archivo1.FormFormatoZ, Archivo1.FormFormatoY,
          TiempoX, TiempoZ, TiempoY);
        break;
      case 1:
        //Titulos de ejes
        m_Graph3D.put_XLabel(Archivo1.GrafEjeX);
        m_Graph3D.put_YLabel(Archivo1.GrafEjeY);
        m_Graph3D.put_ZLabel(Archivo1.GrafEjeZ);
        m_Graph3D.TipoEjes(Archivo1.FormFormatoX,
          Archivo1.FormFormatoY, Archivo1.FormFormatoZ,
          TiempoX, TiempoY, TiempoZ);
        break;
      case 2:
        //Titulos de ejes
        m_Graph3D.put_XLabel(Archivo1.GrafEjeY);
        m_Graph3D.put_YLabel(Archivo1.GrafEjeX);
        m_Graph3D.put_ZLabel(Archivo1.GrafEjeZ);
        m_Graph3D.TipoEjes(Archivo1.FormFormatoY,
          Archivo1.FormFormatoX, Archivo1.FormFormatoZ,
          TiempoY, TiempoX, TiempoZ);
        break;
    }
    m_Graph3D.put_BackColor(Archivo1.GrafColorFondo); //Color de fondo

    m_Graph3D.SetCaption (Archivo1.GrafTitulo);

    //Cambio
    //renombrado del archivo y refresco de graficas
    CGraph* grp = GetDocument()->GetGraph();
    CSimulation *sim = GetDocument()->GetSim();
    ASSERT_VALID(grp);
    ASSERT_VALID(sim);

    char temp[200];
    LPCTSTR buffArchivo=sim->GetFilePath();

    sprintf(temp,"Graphic3D␣%s",Archivo1.GrafTitulo);
    grp->SetName(temp);

    for (unsigned int i=0; i<(strlen(buffArchivo)-4); i++)
      temp[i]=buffArchivo[i];
    temp[strlen(buffArchivo)-4]=0;
    sprintf(temp,"%s\\%s.txt",temp,grp->GetName());
    rename(ArchConf, temp);
    sprintf(ArchConf, temp);

    GetDocument()->SetTitle(sim->m_strSimName + "␣-␣" + grp->GetName());
    AfxGetMainWnd()->PostMessage(IIS_SIM_REFRESHGRAPH,(WPARAM)sim);

    //fin renombrado
    char dirTemp[200];
    for (unsigned int i=0; i<(strlen(buffArchivo)-4); i++)
      dirTemp[i]=buffArchivo[i];
    dirTemp[strlen(buffArchivo)-4]=0;
```

```
    sprintf(dirTemp,"%s\\%s", dirTemp, Archivo1.Direccion);

    Inicia(dirTemp);
    ExtraeLoncha(Archivo1.NumCapa);
    Pinta();
    //fin del repintado
  }
}
```

```
void CGraphView3D::OnContextMenu(CWnd* /*pWnd*/, CPoint point)
{
  // TODO: Agregue aqu\'{\i} su c\'{o}digo de controlador de mensajes
  CMenu GraphMenu;
  GraphMenu.LoadMenu(IDR_CONTEXT_GRAPH3DMENU);
  CMenu *pSubMenu = GraphMenu.GetSubMenu(0);
  CString menu_name;

  CString resString;
  resString.LoadString(::GetModuleHandle(NULL),IDS_COPY_CLIPBOARD_TEXT + desp_Idioma, NULL);
  pSubMenu->ModifyMenu(0,MF_BYPOSITION,pSubMenu->GetMenuItemID(0),resString);
  resString.LoadString(::GetModuleHandle(NULL),IDS_SAVE_BITMAP_TEXT + desp_Idioma, NULL);
  pSubMenu->ModifyMenu(1,MF_BYPOSITION,pSubMenu->GetMenuItemID(1),resString);
  resString.LoadString(::GetModuleHandle(NULL),IDS_PRINT_TEXT + desp_Idioma, NULL);
  pSubMenu->ModifyMenu(2,MF_BYPOSITION,pSubMenu->GetMenuItemID(2),resString);
  CMenu* pPopup = GraphMenu.GetSubMenu(0);
  if (AfxGetMainWnd()->IsKindOf(RUNTIME_CLASS(CBCGPMDIFrameWnd)))
  {
    CBCGPPopupMenu* pPopupMenu = new CBCGPPopupMenu;

    if (!pPopupMenu->Create(this, point.x, point.y, (HMENU)pPopup->m_hMenu, FALSE, TRUE))
      return;

    ((CBCGPMDIFrameWnd*)AfxGetMainWnd())->OnShowPopupMenu (pPopupMenu);
    UpdateDialogControls(this, FALSE);
   }
}
```

```
void CGraphView3D::OnGraph3dCopiar()
{
  m_Graph3D.CopyToClipboard();
}
```

```
void CGraphView3D::OnGraph3dGuardar()
{
  CString fileName;
  CFileDialog dlgFile(false,"bmp",fileName,OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT,
    "Bitmaps (*.bmp)|*.bmp|All Files (*.*)|*.*||",NULL,0);
  //crear buffer para el nombre del fichero
    dlgFile.m_ofn.lpstrFile = fileName.GetBuffer(_MAX_PATH);

    INT_PTR nResult = dlgFile.DoModal(); //mostrar di\'{a}logo GUARDAR
  fileName.ReleaseBuffer(); //borrar buffer

  if (nResult == IDOK)
    m_Graph3D.SaveBitMap(fileName);
}
```

Páginas de Propiedades

```
/////////////////////////////////
//Se definen las clases CGraph3D_Page1,CGraph3D_Page2, CGraph3D_Page3 y CGraph3DProp
/////////////////////////////////
```

```
/////////////////////////////////
//CGraph3DProp
/////////////////////////////////
```

```
void CGraph3DProp::OnBnClickedOk()
{
  Grabar();
  Archivo1=ArchTemp;
  Archivo1.Grabar(ArchPath);
  OnOK();
}
```

```
void CGraph3DProp::OnBnClickedCancel()
{
  ArchTemp=Archivo1;
  OnCancel();
}
```

```
void CGraph3DProp::ArchivoPath(char *nombre, CSimulation *sim1, CGraph* grp1)
{
  int i=0;
  do
  {
    ArchPath[i]=nombre[i];
    i++;
  }
  while (nombre[i]!=0);
  ArchPath[i]=0;
  sim=sim1;
  grp=grp1;
}
```

```
void CGraph3DProp::Grabar()
{
  //PAGE 1
  //Guardar titulo
  CString strBuff;
  m_tabControl.m_tabPages[0]->GetDlgItemText(IDC_ETITULO, strBuff);
  int i=-1;
  do
  {
    i++;
    ArchTemp.GrafTitulo[i] = strBuff[i];

  } while (strBuff[i] != 0);

  //Estilo
  CComboBox *estilo;
  estilo = (CComboBox*) m_tabControl.m_tabPages[0]->GetDlgItem(IDC_CESTILO);
  ArchTemp.GrafEstilo = estilo->GetCurSel();

  //Leyenda x
  m_tabControl.m_tabPages[0]->GetDlgItemText(IDC_ELABELX, strBuff);
  i=-1;
  do
  {
    i++;
    ArchTemp.GrafEjeX[i] = strBuff[i];

  } while (strBuff[i] != 0);

  //Leyenda y
  m_tabControl.m_tabPages[0]->GetDlgItemText(IDC_ELABELY, strBuff);
```

```
i=-1;
do
{
  i++;
  ArchTemp.GrafEjeY[i] = strBuff[i];

} while (strBuff[i] != 0);

//Leyenda z
m_tabControl.m_tabPages[0]->GetDlgItemText(IDC_ELABELZ, strBuff);
i=-1;
do
{
  i++;
  ArchTemp.GrafEjeZ[i] = strBuff[i];

} while (strBuff[i] != 0);

//Color fondo
CColourPicker *colorFondo;
colorFondo = (CColourPicker*) m_tabControl.m_tabPages[0]->GetDlgItem(IDC_BCOLORFONDO);
ArchTemp.GrafColorFondo = colorFondo->GetColour();

//Minimo
char str[50];
m_tabControl.m_tabPages[0]->GetDlgItemText(IDC_EMINIMO,strBuff);
i=-1;
do
{
  i++;
  str[i] = strBuff[i];

} while (strBuff[i] != 0);
ArchTemp.GrafMinimo=atof(str);

//Maximo
m_tabControl.m_tabPages[0]->GetDlgItemText(IDC_EMAXIMO,strBuff);
i=-1;
do
{
  i++;
  str[i] = strBuff[i];

} while (strBuff[i] != 0);
ArchTemp.GrafMaximo=atof(str);

//logaritmica
CButton *logaritmico;
logaritmico = (CButton*) m_tabControl.m_tabPages[0]->GetDlgItem(IDC_CHECK2);

if (logaritmico->GetCheck()==BST_CHECKED)
  ArchTemp.GrafYLogaritmico = 1;
else
  ArchTemp.GrafYLogaritmico = 0;

//Auto Rango
CButton *Autorange;
Autorange = (CButton*) m_tabControl.m_tabPages[0]->GetDlgItem(IDC_CHAUTORANGE);

if (Autorange->GetCheck()==BST_CHECKED)
  ArchTemp.GrafAutoRango = 1;
else
  ArchTemp.GrafAutoRango = 0;

//PAGE 2
//Color linea
```

```
  CColourPicker *colorLinea;
  colorLinea = (CColourPicker*) m_tabControl.m_tabPages[1]->GetDlgItem(IDC_BCOLORLINEA);
  ArchTemp.ElemColorLinea = colorLinea->GetColour();

  //Color punto
  CColourPicker *colorPunto;
  colorPunto = (CColourPicker*) m_tabControl.m_tabPages[1]->GetDlgItem(IDC_BCOLORPUNTO);
  ArchTemp.ElemColorPunto = colorPunto->GetColour();

  //Color superficie
  CColourPicker *colorSuperficie;
  colorSuperficie = (CColourPicker*) m_tabControl.m_tabPages[1]->
        GetDlgItem(IDC_BCOLORSUPERFICIE);
  ArchTemp.ElemColorSuperficie = colorSuperficie->GetColour();

  //anchura
  ArchTemp.ElemAnchura = m_tabControl.m_tabPages[1]->GetDlgItemInt(IDC_EANCHURA);

  //tama\~{n}o
  ArchTemp.ElemTamano = m_tabControl.m_tabPages[1]->GetDlgItemInt(IDC_ETAMANO);
}
```

```
/////////////////////////////////
//CGraph3D_Page1
/////////////////////////////////
```

```
BOOL CGraph3D_Page1::OnInitDialog()
{
  CDialog::OnInitDialog();
  //Puesta de los textos en el idioma correcto
  SetDlgItemText(IDC_P1_TITULO,GetResString(IDS_TITULO));
  SetDlgItemText(IDC_P1_ESTILO, GetResString(IDS_ESTILO));
  SetDlgItemText(IDC_P1_FONDO, GetResString(IDS_FONDO));
  SetDlgItemText(IDC_P1_MIN ,GetResString(IDS_MINIMO));
  SetDlgItemText(IDC_P1_MAX, GetResString(IDS_MAXIMO));
  SetDlgItemText(IDC_P1_RANGO, GetResString(IDS_RANGO));
  SetDlgItemText(IDC_P1_LEY_X, GetResString(IDS_LEY_X));
  SetDlgItemText(IDC_P1_LEY_Y, GetResString(IDS_LEY_Y));
  SetDlgItemText(IDC_P1_LEY_Z, GetResString(IDS_LEY_Z));
  SetDlgItemText(IDC_AUTORANGO, GetResString(IDS_AUTORANGO));
  SetDlgItemText(IDC_LOG, GetResString(IDS_LOG));

  //inicializaci\'{o}n de valores
  SetDlgItemText(IDC_ETITULO,ArchTemp.GrafTitulo);
  SetDlgItemText(IDC_ELABELX,ArchTemp.GrafEjeX);
  SetDlgItemText(IDC_ELABELY,ArchTemp.GrafEjeY);
  SetDlgItemText(IDC_ELABELZ,ArchTemp.GrafEjeZ);
  char str[30];
  sprintf(str, "%e", ArchTemp.GrafMinimo);
  SetDlgItemText(IDC_EMINIMO,str);
  sprintf(str, "%e", ArchTemp.GrafMaximo);
  SetDlgItemText(IDC_EMAXIMO,str);
  m_bEstilo.SetCurSel(ArchTemp.GrafEstilo);
  m_bColorFondo.SetColour(ArchTemp.GrafColorFondo);

  m_bLogaritmico.SetCheck(ArchTemp.GrafYLogaritmico);

  m_bAutoRango.SetCheck(ArchTemp.GrafAutoRango);
  ((CEdit*)GetDlgItem(IDC_EMAXIMO))->SetReadOnly(m_bAutoRango.GetCheck());
  ((CEdit*)GetDlgItem(IDC_EMINIMO))->SetReadOnly(m_bAutoRango.GetCheck());
  return FALSE;
}
```

```
void CGraph3D_Page1::OnBnClickedChautorange()
{
```

```
  ((CEdit*)GetDlgItem(IDC_EMAXIMO))->SetReadOnly(m_bAutoRango.GetCheck());
  ((CEdit*)GetDlgItem(IDC_EMINIMO))->SetReadOnly(m_bAutoRango.GetCheck());
}
```

```
void CGraph3D_Page1::OnBnClickedCheck2()
{
  if (m_bLogaritmico.GetCheck())
  {
    CString strBuff;
    char str[50];
    GetDlgItemText(IDC_EMINIMO,strBuff);
    int i=-1;
    do
    {
      i++;
      str[i] = strBuff[i];

    } while (strBuff[i] != 0);
    double Minimo=atof(str);
    if ( Minimo <=0)
    {
      SetDlgItemInt(IDC_EMINIMO,1);
    }
  }
}
```

```
////////////////////////////////
//CGraph3D_Page2
////////////////////////////////
```

```
BOOL CGraph3D_Page2::OnInitDialog()
{
  CDialog::OnInitDialog();

  CSpinButtonCtrl* pSpin1;
  CSpinButtonCtrl* pSpin2;
  pSpin1 = (CSpinButtonCtrl*)GetDlgItem(IDC_SANCHURA);
  // establecemos la ventana relacionada
  pSpin1->SetBuddy (GetDlgItem(IDC_EANCHURA));
  // establecemos los valores m\'{a}ximo y m\'{\i}nimo
  pSpin1->SetRange(0,5);

  pSpin2 = (CSpinButtonCtrl*)GetDlgItem(IDC_STAMANO);
  // establecemos la ventana relacionada
  pSpin2->SetBuddy (GetDlgItem(IDC_ETAMANO));
  // establecemos los valores m\'{a}ximo y m\'{\i}nimo
  pSpin2->SetRange(0,6);

  //Puesta de los textos en el idioma correcto
  SetDlgItemText(IDC_COLOR_LINEA,GetResString(IDS_COLOR_LINEA));
  SetDlgItemText(IDC_ANCHURA, GetResString(IDS_ANCHURA));
  SetDlgItemText(IDC_COLOR_SUPERFICIE, GetResString(IDS_COLOR_SUP));
  SetDlgItemText(IDC_COLOR_PUNTO ,GetResString(IDS_COLOR_PUNTO));
  SetDlgItemText(IDC_TAMANO, GetResString(IDS_TAMANO));

  //inicializacion de valores a los del archivo
  m_bColorPunto.SetColour(ArchTemp.ElemColorPunto);
  m_bColorLinea.SetColour(ArchTemp.ElemColorLinea);
  m_bColorSuperficie.SetColour(ArchTemp.ElemColorSuperficie);

  SetDlgItemInt(IDC_EANCHURA,ArchTemp.ElemAnchura);
  SetDlgItemInt(IDC_ETAMANO,ArchTemp.ElemTamano);

  return FALSE;
}
```

```
/////////////////////////////////
//CGraph3D_Page3
/////////////////////////////////
```

```
BOOL CGraph3D_Page3::OnInitDialog()
{
  //Puesta de los textos en el idioma correcto
  SetDlgItemText(IDC_P3_PLANTILLA,GetResString(IDS_PLANTILLA));
  SetDlgItemText(IDC_P3_EJE, GetResString(IDS_EJE));
  SetDlgItemText(IDC_P3_TIPO, GetResString(IDS_TIPO));
  SetDlgItemText(IDC_P3_FORMATO, GetResString(IDS_FORMATO));

  CDialog::OnInitDialog();
  SetDlgItemText(IDC_EFORMATO,ArchTemp.FormFormatoX);
  m_bEje.SetCurSel(0);
  m_bTipo.SetCurSel(ArchTemp.FormTipoX);
  OnCbnSelchangeCtipo();
  return FALSE;
}
```

```
void CGraph3D_Page3::OnCbnSelchangeCeje()
{
  switch (m_bEje.GetCurSel())
  {
    case 0:
      m_bTipo.SetCurSel(ArchTemp.FormTipoX);
      SetDlgItemText(IDC_EFORMATO,ArchTemp.FormFormatoX);
      break;
    case 1:
      m_bTipo.SetCurSel(ArchTemp.FormTipoY);
      SetDlgItemText(IDC_EFORMATO,ArchTemp.FormFormatoY);
      break;
    case 2:
      m_bTipo.SetCurSel(ArchTemp.FormTipoZ);
      SetDlgItemText(IDC_EFORMATO,ArchTemp.FormFormatoZ);
      break;
  }
  OnCbnSelchangeCtipo();
}
```

```
void CGraph3D_Page3::OnCbnSelchangeCtipo()
{
  switch (m_bEje.GetCurSel())
  {
    case 0:
      ArchTemp.FormTipoX = m_bTipo.GetCurSel();
    break;
      case 1:
      ArchTemp.FormTipoY = m_bTipo.GetCurSel();
      break;
    case 2:
      ArchTemp.FormTipoZ = m_bTipo.GetCurSel();
      break;
  }
  switch(m_bTipo.GetCurSel())
  {
    case 0:
      //SetAxisTimeFormat("FALSE");
      m_bPlantilla.ResetContent();
      m_bPlantilla.AddString(".");
            m_bPlantilla.AddString(".#");
      m_bPlantilla.AddString(".##");
      m_bPlantilla.AddString(".###");
      m_bPlantilla.AddString(".####");
```

```
      break;
    case 1:
      //SetAxisTimeFormat("FALSE");
      m_bPlantilla.ResetContent();
      m_bPlantilla.AddString("e");
            m_bPlantilla.AddString(".#e");
      m_bPlantilla.AddString(".##e");
      m_bPlantilla.AddString(".###e");
      m_bPlantilla.AddString(".####e");
      m_bPlantilla.AddString("E");
            m_bPlantilla.AddString(".#E");
      m_bPlantilla.AddString(".##E");
      m_bPlantilla.AddString(".###E");
      m_bPlantilla.AddString(".####E");
      break;
    case 2:
      //SetAxisTimeFormat("FALSE");
      m_bPlantilla.ResetContent();
      m_bPlantilla.AddString("V");
            m_bPlantilla.AddString("A");
      m_bPlantilla.AddString("Hz");
      m_bPlantilla.AddString("g");
      m_bPlantilla.AddString("Deg");
      m_bPlantilla.AddString(".␣%");
      m_bPlantilla.AddString(".#␣%");
            m_bPlantilla.AddString(".##␣%");
      m_bPlantilla.AddString("$␣.");
      m_bPlantilla.AddString("$␣.#");
      m_bPlantilla.AddString("$␣.##");
      break;
    case 3:
      m_bPlantilla.ResetContent();
      m_bPlantilla.AddString("ddd/mmm/yyy");
      m_bPlantilla.AddString("dd/mm/yy");
      m_bPlantilla.AddString("d/m/y");
      m_bPlantilla.AddString("m/y");
      m_bPlantilla.AddString("d/m");
      break;
    case 4:
      m_bPlantilla.ResetContent();
      m_bPlantilla.AddString("h:m:s");
      m_bPlantilla.AddString("hh:m");
      m_bPlantilla.AddString("h:m");
      m_bPlantilla.AddString("m:s");
      break;
  }
}
```

```
void CGraph3D_Page3::OnLbnSelchangeLplantilla()
{
  switch(m_bTipo.GetCurSel())
  {
    case 0:
      if(m_bPlantilla.GetCaretIndex()==0)
        SetDlgItemText(IDC_EFORMATO, "%g");
      else if(m_bPlantilla.GetCaretIndex()==1)
        SetDlgItemText(IDC_EFORMATO, "%.1f");
      else if(m_bPlantilla.GetCaretIndex()==2)
        SetDlgItemText(IDC_EFORMATO, "%.2f");
      else if(m_bPlantilla.GetCaretIndex()==3)
        SetDlgItemText(IDC_EFORMATO, "%.3f");
      else if(m_bPlantilla.GetCaretIndex()==4)
        SetDlgItemText(IDC_EFORMATO, "%.4f");
      break;
    case 1:
      if(m_bPlantilla.GetCaretIndex()==0)
```

```
19          SetDlgItemText(IDC_EFORMATO, "%.e");
          else if(m_bPlantilla.GetCaretIndex()==1)
            SetDlgItemText(IDC_EFORMATO, "%.1e");
          else if(m_bPlantilla.GetCaretIndex()==2)
            SetDlgItemText(IDC_EFORMATO, "%.2e");
          else if(m_bPlantilla.GetCaretIndex()==3)
            SetDlgItemText(IDC_EFORMATO, "%.3e");
          else if(m_bPlantilla.GetCaretIndex()==4)
            SetDlgItemText(IDC_EFORMATO, "%.4e");
          else if(m_bPlantilla.GetCaretIndex()==5)
29          SetDlgItemText(IDC_EFORMATO, "%.E");
          else if(m_bPlantilla.GetCaretIndex()==6)
            SetDlgItemText(IDC_EFORMATO, "%.1E");
          else if(m_bPlantilla.GetCaretIndex()==7)
            SetDlgItemText(IDC_EFORMATO, "%.2E");
          else if(m_bPlantilla.GetCaretIndex()==8)
            SetDlgItemText(IDC_EFORMATO, "%.3E");
          else if(m_bPlantilla.GetCaretIndex()==9)
            SetDlgItemText(IDC_EFORMATO, "%.4E");
          break;
39      case 2:
          if(m_bPlantilla.GetCaretIndex()==0)
            SetDlgItemText(IDC_EFORMATO, "%g␣V");
          else if(m_bPlantilla.GetCaretIndex()==1)
            SetDlgItemText(IDC_EFORMATO, "%g␣A");
          else if(m_bPlantilla.GetCaretIndex()==2)
            SetDlgItemText(IDC_EFORMATO, "%g␣Hz");
          else if(m_bPlantilla.GetCaretIndex()==3)
            SetDlgItemText(IDC_EFORMATO, "%g␣g");
          else if(m_bPlantilla.GetCaretIndex()==4)
49          SetDlgItemText(IDC_EFORMATO, "%g␣Deg");
          else if(m_bPlantilla.GetCaretIndex()==5)
            SetDlgItemText(IDC_EFORMATO, "%g␣%%");
          else if(m_bPlantilla.GetCaretIndex()==6)
            SetDlgItemText(IDC_EFORMATO, "%.1f␣%%");
          else if(m_bPlantilla.GetCaretIndex()==7)
            SetDlgItemText(IDC_EFORMATO, "%.2f␣%%");
          else if(m_bPlantilla.GetCaretIndex()==8)
            SetDlgItemText(IDC_EFORMATO, "$␣%g");
          else if(m_bPlantilla.GetCaretIndex()==9)
59          SetDlgItemText(IDC_EFORMATO, "$␣%.1f");
          else if(m_bPlantilla.GetCaretIndex()==10)
            SetDlgItemText(IDC_EFORMATO, "$␣%.2f");
          break;
        case 3:
          if(m_bPlantilla.GetCaretIndex()==0)
            SetDlgItemText(IDC_EFORMATO, "%d␣%a/%B/%Y");
          else if(m_bPlantilla.GetCaretIndex()==1)
            SetDlgItemText(IDC_EFORMATO, "%d/%b/%Y");
                else if(m_bPlantilla.GetCaretIndex()==2)
69          SetDlgItemText(IDC_EFORMATO, "%d/%m/%Y");
                else if(m_bPlantilla.GetCaretIndex()==3)
            SetDlgItemText(IDC_EFORMATO, "%b/%Y");
                else if(m_bPlantilla.GetCaretIndex()==4)
            SetDlgItemText(IDC_EFORMATO, "%d/%B");
          break;
        case 4:
          if(m_bPlantilla.GetCaretIndex()==0)
            SetDlgItemText(IDC_EFORMATO, "%H:%M:%S");
          else if(m_bPlantilla.GetCaretIndex()==1)
79          SetDlgItemText(IDC_EFORMATO, "%H␣%p/%M");
          else if(m_bPlantilla.GetCaretIndex()==2)
            SetDlgItemText(IDC_EFORMATO, "%H:%M");
                else if(m_bPlantilla.GetCaretIndex()==3)
            SetDlgItemText(IDC_EFORMATO, "%M:%S");
          break;
```

```
  }

  CString strBuff;
  int i;
  switch (m_bEje.GetCurSel())
  {
    case 0:
      GetDlgItemText(IDC_EFORMATO, strBuff);
      i=-1;
      do
      {
        i++;
        ArchTemp.FormFormatoX[i] = strBuff[i];

      } while (strBuff[i] != 0);
      break;

    case 1:
      GetDlgItemText(IDC_EFORMATO, strBuff);
      i=-1;
      do
      {
        i++;
        ArchTemp.FormFormatoY[i] = strBuff[i];

      } while (strBuff[i] != 0);
      break;
    case 2:
      GetDlgItemText(IDC_EFORMATO, strBuff);
      i=-1;
      do
      {
        i++;
        ArchTemp.FormFormatoZ[i] = strBuff[i];

      } while (strBuff[i] != 0);
      break;
  }
}
```

Guardado de Propiedades

```
//////////////////////////////////
//Se definen procedimientos en Graph3dGrabar.h
//////////////////////////////////
```

```
struct  ARCHIVO
{
  char Direccion[200];
  //Pincipal
  int NumCapa;
  int Acumulacion;
  int AnguloX;
  int AnguloY;
  int Plano;

  //PAGE_1
  int GrafEstilo;
  char GrafTitulo[50];
  char GrafEjeX[30];
  char GrafEjeY[30];
  char GrafEjeZ[30];
```

```
  COLORREF GrafColorFondo;
  double GrafMinimo;
  double GrafMaximo;
  int GrafYLogaritmico;
  int GrafAutoRango;

  //PAGE_2
  COLORREF ElemColorLinea;
  int ElemAnchura;
  COLORREF ElemColorPunto;
  int ElemTamano;
  COLORREF ElemColorSuperficie;

  //PAGE_3
  int FormTipoX;
  int FormTipoY;
  int FormTipoZ;
  char FormFormatoX[10];
  char FormFormatoY[10];
  char FormFormatoZ[10];

  //Procedimientos
  void Grabar(LPCSTR strPath);
  void Recuperar(LPCSTR strPath);
  void Inicializa();
} ;
```

```
ARCHIVO Archivo1;
ARCHIVO ArchTemp;
```

```
void ARCHIVO::Grabar(LPCSTR strPath)
{
  FILE *IF2;
  IF2 = fopen(strPath,"w"); //Abierto el archivo y vacio
  if(!IF2)
    MessageBox(0,GetResString(IDS_ERROR_GUARDAR), GetResString(IDS_ERROR_ARCHIVO),
            MB_ICONERROR | MB_OK);

  char str1[200];
  strcpy(str1,"DIRECCION = ");
  strcat(str1,Direccion);
  strcat(str1,"\n");
  fprintf(IF2, str1); //GrafEjeY[20];
  //Principal

  fprintf(IF2, "NUMCAPA = %d\n",NumCapa);
  fprintf(IF2, "ACUMULACION = %d\n", Acumulacion);
  fprintf(IF2, "ANGULOX = %d\n",AnguloX);
  fprintf(IF2, "ANGULOY = %d\n",AnguloY);
  switch (Plano)
  {
    case 0:
      fprintf(IF2, "PLANO = XY\n");
      break;
    case 1:
      fprintf(IF2, "PLANO = ZX\n");
      break;
    case 2:
      fprintf(IF2, "PLANO = YZ\n");
      break;
  }

  //PAGE_1
  switch (GrafEstilo)
  {
    case 0:
```

```
      fprintf(IF2, "ESTILO␣=␣LINEA\n");
      break;
    case 1:
      fprintf(IF2, "ESTILO␣=␣PUNTO\n");
      break;
    case 2:
      fprintf(IF2, "ESTILO␣=␣COMBINADO\n");
      break;
    case 3:
    default:
      fprintf(IF2, "ESTILO␣=␣SUPERFICIE\n");
      break;
  }

  char str[60];
  strcpy(str,"TITULO␣=␣");
  strcat(str,GrafTitulo);
  strcat(str,"\n");
  fprintf(IF2, str);  //GrafTitulo[50];
  strcpy(str,"EJEX␣=␣");
  strcat(str,GrafEjeX);
  strcat(str,"\n");
  fprintf(IF2, str);  //GrafEjeX[20];
  strcpy(str,"EJEY␣=␣");
  strcat(str,GrafEjeY);
  strcat(str,"\n");
  fprintf(IF2, str);  //GrafEjeY[20];
  strcpy(str,"EJEZ␣=␣");
  strcat(str,GrafEjeZ);
  strcat(str,"\n");
  fprintf(IF2, str);  //GrafEjeZ[20];
  fprintf(IF2, "COLORFONDO␣=␣%d␣%d␣%d\n",GetRValue(GrafColorFondo),
        GetGValue(GrafColorFondo), GetBValue(GrafColorFondo));
  fprintf(IF2, "MINIMO␣=␣%e\n",GrafMinimo);
  fprintf(IF2, "MAXIMO␣=␣%e\n",GrafMaximo);
  fprintf(IF2, "YLOGARITMICO␣=␣%d\n",GrafYLogaritmico);
  fprintf(IF2, "AUTORANGO␣=␣%d\n",GrafAutoRango);

  //PAGE_2
  fprintf(IF2, "COLORLINEA␣=␣%d␣%d␣%d\n",GetRValue(ElemColorLinea),
        GetGValue(ElemColorLinea), GetBValue(ElemColorLinea));
  fprintf(IF2, "ANCHURA␣=␣%d\n", ElemAnchura);
  fprintf(IF2, "COLORPUNTO␣=␣%d␣%d␣%d\n",GetRValue(ElemColorPunto),
        GetGValue(ElemColorPunto), GetBValue(ElemColorPunto));
  fprintf(IF2, "TAMANO␣=␣%d\n", ElemTamano);
  fprintf(IF2, "COLORSUPERFICIE␣=␣%d␣%d␣%d\n",GetRValue(ElemColorSuperficie),
        GetGValue(ElemColorSuperficie), GetBValue(ElemColorSuperficie));

  //PAGE_3
  switch (FormTipoX)
  {
    case 0:
      fprintf(IF2, "TIPOX␣=␣NUMERO\n");
      break;
    case 1:
      fprintf(IF2, "TIPOX␣=␣EXPONENCIAL\n");
      break;
    case 2:
      fprintf(IF2, "TIPOX␣=␣SIMBOLICO\n");
      break;
    case 3:
      fprintf(IF2, "TIPOX␣=␣FECHA\n");
      break;
    default:
      fprintf(IF2, "TIPOX␣=␣TIEMPO\n");
  }
```

```
    switch (FormTipoY)
    {
      case 0:
        fprintf(IF2, "TIPOY␣=␣NUMERO\n");
108     break;
      case 1:
        fprintf(IF2, "TIPOY␣=␣EXPONENCIAL\n");
        break;
      case 2:
        fprintf(IF2, "TIPOY␣=␣SIMBOLICO\n");
        break;
      case 3:
        fprintf(IF2, "TIPOY␣=␣FECHA\n");
        break;
118   default:
        fprintf(IF2, "TIPOY␣=␣TIEMPO\n");
    }
    switch (FormTipoZ)
    {
      case 0:
        fprintf(IF2, "TIPOZ␣=␣NUMERO\n");
        break;
      case 1:
        fprintf(IF2, "TIPOZ␣=␣EXPONENCIAL\n");
128     break;
      case 2:
        fprintf(IF2, "TIPOZ␣=␣SIMBOLICO\n");
        break;
      case 3:
        fprintf(IF2, "TIPOZ␣=␣FECHA\n");
        break;
      default:
        fprintf(IF2, "TIPOZ␣=␣TIEMPO\n");
    }
138 strcpy(str,"FORMATOX␣=␣");
    strcat(str,FormFormatoX);
    strcat(str,"\n");
    fputs(str,IF2);

    strcpy(str,"FORMATOY␣=␣");
    strcat(str,FormFormatoY);
    strcat(str,"\n");
    fputs(str,IF2);

148 strcpy(str,"FORMATOZ␣=␣");
    strcat(str,FormFormatoZ);
    strcat(str,"\n");
    fputs(str,IF2);

    fclose(IF2);

  }
```

```
  void ARCHIVO::Recuperar(LPCSTR strPath/*, LPCSTR Titulo*/)
  {
    FILE *IF2;
    IF2 = fopen(strPath,"r"); //Abierto el archivo para lectura
  5 if(!IF2)
    {
      IF2 = fopen(strPath,"w"); //No hay archivo de configuraci\'{o}n, creamos uno
      if(!IF2)
        MessageBox(0,GetResString(IDS_ERROR_CARGAR), GetResString(IDS_ERROR_ARCHIVO),
                   MB_ICONERROR | MB_OK);
      else Inicializa();
```

```
    fclose(IF2);
    Grabar(strPath);
    return;
  }
  char str1[60];
  char str2[60];
  char str3[60];
  //char str4[200];

  int i=-1;

  fscanf(IF2, "%s%s",str1,str2);
  fread(&Direccion[0], sizeof(char),1, IF2);
  do
  {
    i++;
    fread(&Direccion[i], sizeof(char),1, IF2);
  } while (Direccion[i]!='\n');
  Direccion[i]=0;

  //Principal
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  NumCapa=atoi(str3);
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  Acumulacion=atoi(str3);
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  AnguloX=atoi(str3);
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  AnguloY=atoi(str3);
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  if (str3[0]=='X')
    Plano = 0;
  if (str3[0]=='Y')
    Plano = 2;
  if (str3[0]=='Z')
    Plano = 1;


  //PAGE_1
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  if (str3[0]=='L')
    GrafEstilo=0;
  if (str3[0]=='P')
    GrafEstilo=1;
  if (str3[0]=='C')
    GrafEstilo=2;
  if (str3[0]=='S')
    GrafEstilo=3;

  fscanf(IF2, "%s%s",str1,str2);  //GrafTitulo
  i=-1;

  fread(&GrafTitulo[0], sizeof(char),1, IF2);
  do
  {
    i++;
    fread(&GrafTitulo[i], sizeof(char),1, IF2);
  } while (GrafTitulo[i]!='\n');
  GrafTitulo[i]=0;

  fscanf(IF2, "%s%s",str1,str2);  //GrafEjeX
  i=-1;
  fread(&GrafEjeX[0], sizeof(char),1, IF2);
  do
  {
    i++;
```

```
    fread(&GrafEjeX[i], sizeof(char),1, IF2);
  } while (GrafEjeX[i]!='\n');
  GrafEjeX[i]=0;

  fscanf(IF2, "%s%s",str1,str2);  //GrafEjeY
  i=-1;
  fread(&GrafEjeY[0], sizeof(char),1, IF2);
  do
  {
    i++;
    fread(&GrafEjeY[i], sizeof(char),1, IF2);
  } while (GrafEjeY[i]!='\n');
  GrafEjeY[i]=0;

  fscanf(IF2, "%s%s",str1,str2);  //GrafEjeZ
  i=-1;
  fread(&GrafEjeZ[0], sizeof(char),1, IF2);
  do
  {
    i++;
    fread(&GrafEjeZ[i], sizeof(char),1, IF2);
  } while (GrafEjeZ[i]!='\n');
  GrafEjeZ[i]=0;

  fscanf(IF2, "%s%s%s%s%s",str1,str1,str1, str2, str3);
    GrafColorFondo=RGB(atoi(str1),atoi(str2),atoi(str3));

  fscanf(IF2, "%s%s%s",str1,str2,str3);
  GrafMinimo=atof(str3);
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  GrafMaximo=atof(str3);
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  GrafYLogaritmico=atoi(str3);
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  GrafAutoRango=atoi(str3);

  //PAGE_2
  fscanf(IF2, "%s%s%s%s%s",str1,str1,str1, str2, str3);
    ElemColorLinea=RGB(atoi(str1),atoi(str2),atoi(str3));
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  ElemAnchura=atoi(str3);
  fscanf(IF2, "%s%s%s%s%s",str1,str1,str1, str2, str3);
    ElemColorPunto=RGB(atoi(str1),atoi(str2),atoi(str3));
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  ElemTamano=atoi(str3);
  fscanf(IF2, "%s%s%s%s%s",str1,str1,str1, str2, str3);
    ElemColorSuperficie=RGB(atoi(str1),atoi(str2),atoi(str3));

  //PAGE_3
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  if (str3[0]=='N')
    FormTipoX=0;
  if (str3[0]=='E')
    FormTipoX=1;
  if (str3[0]=='S')
    FormTipoX=2;
  if (str3[0]=='F')
    FormTipoX=3;
  if (str3[0]=='T')
    FormTipoX=4; //FormTipoX
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  if (str3[0]=='N')
    FormTipoY=0;
  if (str3[0]=='E')
    FormTipoY=1;
  if (str3[0]=='S')
```

```
    FormTipoY=2;
  if (str3[0]=='F')
    FormTipoY=3;
  if (str3[0]=='T')
    FormTipoY=4; //FormTipoY
  fscanf(IF2, "%s%s%s",str1,str2,str3);
  if (str3[0]=='N')
    FormTipoZ=0;
  if (str3[0]=='E')
    FormTipoZ=1;
  if (str3[0]=='S')
    FormTipoZ=2;
  if (str3[0]=='F')
    FormTipoZ=3;
  if (str3[0]=='T')
    FormTipoZ=4; //FormTipoZ
  fscanf(IF2, "%s%s",str1,str2);  //GrafEjeZ
  i=-1;
  fread(&FormFormatoX[0], sizeof(char),1, IF2);
  do
  {
    i++;
    fread(&FormFormatoX[i], sizeof(char),1, IF2);
  } while (FormFormatoX[i]!='\n');
  FormFormatoX[i]=0;

  fscanf(IF2, "%s%s",str1,str2);  //GrafEjeZ
  i=-1;
  fread(&FormFormatoY[0], sizeof(char),1, IF2);
  do
  {
    i++;
    fread(&FormFormatoY[i], sizeof(char),1, IF2);
  } while (FormFormatoY[i]!='\n');
  FormFormatoY[i]=0;

  fscanf(IF2, "%s%s",str1,str2);  //GrafEjeZ
  i=-1;
  fread(&FormFormatoZ[0], sizeof(char),1, IF2);
  do
  {
    i++;
    fread(&FormFormatoZ[i], sizeof(char),1, IF2);
  } while (FormFormatoZ[i]!='\n');
  FormFormatoZ[i]=0;


  ArchTemp=Archivo1;

  fclose(IF2);
}
```

```
void ARCHIVO::Inicializa()
{
  strcpy(Direccion,"C:\\");
  //Pincipal
  NumCapa = 0;
  AnguloX = 30;
  AnguloY = -45;
  Plano = 0;

  //PAGE_1
  GrafEstilo=3;
  strcpy(GrafTitulo,"Visualizaci\'{o}n␣3D");
  strcpy(GrafEjeX, "x");
  strcpy(GrafEjeY, "y");
```

```
  strcpy(GrafEjeZ, "z");
  GrafColorFondo = RGB(255,255,255);
  GrafMinimo=0;
  GrafMaximo=1e30;
  GrafYLogaritmico=0;
  GrafAutoRango=1;

  //PAGE_2
  ElemColorLinea = RGB(255,0,0);
  ElemAnchura=2;
  ElemColorPunto = RGB(0,255,0);
  ElemTamano=1;
  ElemColorSuperficie = RGB(255,255,255);
  Acumulacion=0;

  //PAGE_3
  FormTipoX=0;
  FormTipoY=0;
  FormTipoZ=1;
  strcpy(FormFormatoX,"%g");
  strcpy(FormFormatoY,"%g");
  strcpy(FormFormatoZ,"%.1e");

  ArchTemp=Archivo1;
}
```

B.4.2. Nuevos Parámetros

```
////////////////////////////////
//Definida nueva clase en NuevosParametros.h y NuevosParametros.cpp
////////////////////////////////
```

```
////////////////////////////////
//Definicion de la nueva clase NuevosParametros.h
////////////////////////////////
// Cuadro de di\'{a}logo de CNuevosParametros
class CNuevosParametros : public CDialog
{
  DECLARE_DYNAMIC(CNuevosParametros)

public:
  CNuevosParametros(CWnd* pParent = NULL);   // Constructor est\'{a}ndar
  virtual ~CNuevosParametros();

// Datos del cuadro de di\'{a}logo
  enum { IDD = IDD_NUEVOS };

protected:
  virtual void DoDataExchange(CDataExchange* pDX);    // Compatibilidad con DDX o DDV

  DECLARE_MESSAGE_MAP()
public:
  CComboBox m_Tipo;

  CString m_bNombre;
  CString m_bAyuda;
  int m_bTipo;
  double m_bMinF, m_bMaxF;
  int m_bMinI, m_bMaxI;
  CString m_bValorTexto;
  int m_bValorInt;
  double m_bValorFloat;
```

```
private:
  BOOL OnInitDialog();

public:
  int GetMinimoInt();
  int GetMaximoInt();
  int GetValorInt();
  double GetMinimoFloat();
  double GetMaximoFloat();
  double GetValorFloat();
  int GetTipo();
  LPCSTR GetNombre();
  LPCSTR GetAyuda();
  LPCSTR GetValorTexto();
  afx_msg void OnCbnSelchangeCTipo();
  afx_msg void OnEnKillfocusEMin();
  afx_msg void OnEnKillfocusEMax();
  afx_msg void OnBnClickedOk();
  afx_msg void OnEnKillfocusEValor();
  afx_msg void OnEnKillfocusENombre();
  afx_msg void OnEnKillfocusEAyuda();
};
```

```
/////////////////////////////////
//NuevosParametros.cpp
/////////////////////////////////
BEGIN_MESSAGE_MAP(CNuevosParametros, CDialog)
  ON_CBN_SELCHANGE(IDC_C_TIPO, OnCbnSelchangeCTipo)
  ON_EN_KILLFOCUS(IDC_E_MIN, OnEnKillfocusEMin)
  ON_EN_KILLFOCUS(IDC_E_MAX, OnEnKillfocusEMax)
  ON_EN_KILLFOCUS(IDC_E_VALOR, OnEnKillfocusEValor)
  ON_BN_CLICKED(IDOK, OnBnClickedOk)
  ON_EN_KILLFOCUS(IDC_E_NOMBRE, OnEnKillfocusENombre)
  ON_EN_KILLFOCUS(IDC_E_AYUDA, OnEnKillfocusEAyuda)
END_MESSAGE_MAP()
```

```
// Controladores de mensajes de CNuevosParametros
BOOL CNuevosParametros::OnInitDialog()
{
  CDialog::OnInitDialog();

    SetWindowText(GetResString(IDS_NUEVOS_PARAMETROS));
  SetDlgItemText(IDC_S_NOMBRE,GetResString(IDS_PROP_SIMNAME));
  SetDlgItemText(IDC_S_AYUDA, GetResString(IDS_GEN_DESCRIPTION));
  SetDlgItemText(IDC_S_TIPO, GetResString(IDS_PROP_POTENTIAL_TYPE));
  SetDlgItemText(IDC_S_VALOR, GetResString(IDS_VALOR_DEFECTO));
  SetDlgItemText(IDC_S_MIN, GetResString(IDS_MINIMO));
  SetDlgItemText(IDC_S_MAX, GetResString(IDS_MAXIMO));

  m_Tipo.SetCurSel(0);
  m_bTipo=0;
  ((CEdit*)GetDlgItem(IDC_E_MAX))->SetReadOnly(1);
  ((CEdit*)GetDlgItem(IDC_E_MIN))->SetReadOnly(1);
    SetDlgItemText(IDC_E_VALOR,GetResString(IDS_VALOR_DEFECTO_BOOL));

  int m_iLang = theApp.prefs.GetLangx();
  switch (m_iLang)
  {
    case LANG_ENG:
      m_bValorTexto = "True";
    break;
    case LANG_SPA:
      m_bValorTexto = "Verdadero";
    default:
    break;
```

```
    }

    m_bValorInt = 0;
      m_bValorFloat = 0.0;
    SetDlgItemInt(IDC_E_MIN,0);
    m_bMinI = 0;
    m_bMinF = 0.0;
    SetDlgItemInt(IDC_E_MAX,0);
38  m_bMaxI = 0;
    m_bMaxF = 0.0;

    return FALSE;
  }
```

```
  void CNuevosParametros::OnCbnSelchangeCTipo()
  {
    m_bTipo=m_Tipo.GetCurSel();
    int m_iLang = theApp.prefs.GetLangx();
    switch (m_bTipo)
    {
    case 0:
 8    m_bTipo=0;
      ((CEdit*)GetDlgItem(IDC_E_MAX))->SetReadOnly(1);
      ((CEdit*)GetDlgItem(IDC_E_MIN))->SetReadOnly(1);
      SetDlgItemText(IDC_E_VALOR,GetResString(IDS_VALOR_DEFECTO_BOOL));


      switch (m_iLang)
      {
        case LANG_ENG:
          m_bValorTexto = "True";
18      break;
        case LANG_SPA:
          m_bValorTexto = "Verdadero";
        default:
        break;
      }
      break;
    case 1:
      m_bTipo=1;
      ((CEdit*)GetDlgItem(IDC_E_MAX))->SetReadOnly(0);
28    ((CEdit*)GetDlgItem(IDC_E_MIN))->SetReadOnly(0);
      SetDlgItemText(IDC_E_VALOR,GetResString(IDS_VALOR_DEFECTO_INT));
      m_bValorInt = 0;
      SetDlgItemText(IDC_E_MIN,GetResString(IDS_VALOR_DEFECTO_INT));
      m_bMinI = 0;
      SetDlgItemText(IDC_E_MAX,GetResString(IDS_VALOR_DEFECTO_INT));
      m_bMaxI = 0;
      break;
    case 2:
      m_bTipo=2;
38    ((CEdit*)GetDlgItem(IDC_E_MAX))->SetReadOnly(0);
      ((CEdit*)GetDlgItem(IDC_E_MIN))->SetReadOnly(0);
      SetDlgItemText(IDC_E_VALOR,GetResString(IDS_VALOR_DEFECTO_FLOAT));
      m_bValorFloat=0.0;
      SetDlgItemText(IDC_E_MIN,GetResString(IDS_VALOR_DEFECTO_FLOAT));
      m_bMinF = 0.0;
      SetDlgItemText(IDC_E_MAX,GetResString(IDS_VALOR_DEFECTO_FLOAT));
      m_bMaxF = 0.0;
      break;
    case 3:
48    m_bTipo=3;
      ((CEdit*)GetDlgItem(IDC_E_MAX))->SetReadOnly(1);
      ((CEdit*)GetDlgItem(IDC_E_MIN))->SetReadOnly(1);
      SetDlgItemText(IDC_E_VALOR,NULL);
      m_bValorTexto="";
```

```
  default:
    break;
  }
}
```

```
void CNuevosParametros::OnEnKillfocusEMin()
{
  CString Valor;
  GetDlgItemText(IDC_E_MIN,Valor);
  char str[50];
  int i=-1;
  do
  {
    i++;
    str[i] = Valor[i];

  } while (Valor[i] != 0);
  switch (m_Tipo.GetCurSel())
  {
  case 1:
    if (m_bMaxI >= atoi(str))
    {
      m_bMinI = atoi(str);
      SetDlgItemInt(IDC_E_MIN,m_bMinI);
    }
    else
    {
      MessageBox(GetResString(IDS_MAX_MENOR_MIN),NULL,MB_ICONEXCLAMATION);
      SetDlgItemInt(IDC_E_MIN,m_bMinI);
    }
    break;
  case 2:
    if (m_bMaxF >= atof(str))
    {
      m_bMinF = atof(str);
      sprintf(str,"%g",m_bMinF);
      SetDlgItemText(IDC_E_MIN,str);
    }
    else
    {
      MessageBox(GetResString(IDS_MAX_MENOR_MIN),NULL,MB_ICONEXCLAMATION);
      sprintf(str,"%g",m_bMinF);
      SetDlgItemText(IDC_E_MIN,str);
    }
    break;
  }
}
```

```
void CNuevosParametros::OnEnKillfocusEMax()
{
  CString Valor;
  GetDlgItemText(IDC_E_MAX,Valor);
  char str[50];
  int i=-1;
  do
  {
    i++;
    str[i] = Valor[i];

  } while (Valor[i] != 0);
  switch (m_Tipo.GetCurSel())
  {
  case 1:
    if (m_bMinI <= atoi(str))
    {
      m_bMaxI = atoi(str);
```

```
        SetDlgItemInt(IDC_E_MAX,m_bMaxI);
      }
      else
      {
        MessageBox(GetResString(IDS_MIN_MAYOR_MAX),NULL,MB_ICONEXCLAMATION);
        SetDlgItemInt(IDC_E_MAX,m_bMaxI);
      }
      break;
    case 2:
      if (m_bMinF <= atof(str))
      {
        m_bMaxF = atof(str);
        sprintf(str,"%g",m_bMaxF);
        SetDlgItemText(IDC_E_MAX,str);
      }
      else
      {
        MessageBox(GetResString(IDS_MIN_MAYOR_MAX),NULL,MB_ICONEXCLAMATION);
        sprintf(str,"%g",m_bMaxF);
        SetDlgItemText(IDC_E_MAX,str);
      }
      break;
    }
  }
```

```
  void CNuevosParametros::OnBnClickedOk()
  {
    CString Valor;
    if (GetDlgItemText(IDC_E_NOMBRE,Valor)
      && GetDlgItemText(IDC_E_AYUDA,Valor)
      && GetDlgItemText(IDC_E_VALOR,Valor))
      OnOK();
    else
      MessageBox(GetResString(IDS_INCOMPLETO),NULL,MB_ICONEXCLAMATION);
  }
```

```
  void CNuevosParametros::OnEnKillfocusEValor()
  {
    CString Valor;
    int tempI;
    double tempF;
    GetDlgItemText(IDC_E_VALOR,Valor);
    char str[50];
    int i=-1;
    do
    {
      i++;
      str[i] = Valor[i];

    } while (Valor[i] != 0);

    switch (m_Tipo.GetCurSel())
    {
      case 0:
        if (Valor != GetResString(IDS_VERDADERO)
          && Valor != GetResString(IDS_FALSO))
        {
           MessageBox(GetResString(IDS_VALOR_INCORRECTO_BOOL),NULL,MB_ICONEXCLAMATION);
          SetDlgItemText(IDC_E_VALOR,m_bValorTexto);
        }
        else
        {
          m_bValorTexto = Valor;
          SetDlgItemText(IDC_E_VALOR,m_bValorTexto);
        }
        break;
```

```
    case 1:
      tempI = atoi(str);
      if (m_bMinI > tempI
        || m_bMaxI < tempI)
      {
        MessageBox(GetResString(IDS_VALOR_INCORRECTO_NUM),NULL,MB_ICONEXCLAMATION);
        SetDlgItemInt(IDC_E_VALOR,m_bValorInt);
      }
      else
      {
        m_bValorInt = tempI;
        SetDlgItemInt(IDC_E_VALOR,m_bValorInt);
      }
      break;
    case 2:
      tempF = atof(str);
      if (m_bMinF > tempF
        || m_bMaxF < tempF)
      {
        MessageBox(GetResString(IDS_VALOR_INCORRECTO_NUM),NULL,MB_ICONEXCLAMATION);
        sprintf(str, "%g",m_bValorFloat);
        SetDlgItemText(IDC_E_VALOR,str);
      }
      else
      {
        m_bValorFloat = tempF;
        sprintf(str, "%g",m_bValorFloat);
        SetDlgItemText(IDC_E_VALOR,str);
      }
      break;
    case 3:
      m_bValorTexto = Valor;
      SetDlgItemText(IDC_E_VALOR,m_bValorTexto);
      break;
  }

}
```

```
void CNuevosParametros::OnEnKillfocusENombre()
{
  GetDlgItemText(IDC_E_NOMBRE,m_bNombre);
  SetDlgItemText(IDC_E_NOMBRE,m_bNombre);
}
```

```
void CNuevosParametros::OnEnKillfocusEAyuda()
{
  GetDlgItemText(IDC_E_AYUDA,m_bAyuda);
  SetDlgItemText(IDC_E_AYUDA,m_bAyuda);
}
```

B.4.3. Idiomas

```
//////////////////////////////////
//Seleccion del idioma de la aplicaci\'{o}n
//Esta seccion se repite varias veces por todo el codigo
//////////////////////////////////
int desp_Idioma = 0;
CPreferences *prefs = &theApp.prefs;
int m_iLang = prefs->GetLangx();
switch (m_iLang)
{
```

```
    case LANG_ENG:
      desp_Idioma = 2000; //maximo id 65000
    break;
    case LANG_SPA:
      desp_Idioma = 0;
    default:
    break;
  }
```

```
  ////////////////////////////////
  //SimTreeCtrl.cpp
  //Textos de los submenus
  ////////////////////////////////
  void CSimTreeCtrl::OnContextMenu(CWnd* /*pWnd*/, CPoint point)
  {
    POINT point2 = point; //punto en coordernadas de pantalla

    ScreenToClient(&point2); //convertir punto a coordenadas de cliente
    SelectItem(HitTest(point2));

    CMenu SimMenu;
    SimMenu.LoadMenu(IDR_SIMLIST_MENU);

  [...]

      CString resString;
    resString.LoadString(::GetModuleHandle(NULL),IDS_PROP_TEXT + desp_Idioma, NULL);
    pSubMenu->ModifyMenu(0,MF_BYPOSITION,pSubMenu->GetMenuItemID(0),resString);
    resString.LoadString(::GetModuleHandle(NULL),IDS_INICIAR_TEXT + desp_Idioma, NULL);
    pSubMenu->ModifyMenu(1,MF_BYPOSITION,pSubMenu->GetMenuItemID(1),resString);
    resString.LoadString(::GetModuleHandle(NULL),IDS_DETENER_TEXT + desp_Idioma, NULL);
    pSubMenu->ModifyMenu(2,MF_BYPOSITION,pSubMenu->GetMenuItemID(2),resString);
    resString.LoadString(::GetModuleHandle(NULL),IDS_ELIMINAR_SIM_TEXT + desp_Idioma, NULL);
    pSubMenu->ModifyMenu(4,MF_BYPOSITION,pSubMenu->GetMenuItemID(4),resString);
    resString.LoadString(::GetModuleHandle(NULL),IDS_NUEVA_TEXT + desp_Idioma, NULL);
    pSubMenu->ModifyMenu(5,MF_BYPOSITION,pSubMenu->GetMenuItemID(5),resString);
    resString.LoadString(::GetModuleHandle(NULL),IDS_GUARDAR_COMO_TEXT + desp_Idioma, NULL);
    pSubMenu->ModifyMenu(6,MF_BYPOSITION,pSubMenu->GetMenuItemID(6),resString);
    resString.LoadString(::GetModuleHandle(NULL),IDS_DUPLICAR + desp_Idioma, NULL);
    pSubMenu->ModifyMenu(8,MF_BYPOSITION,pSubMenu->GetMenuItemID(8),resString);
    resString.LoadString(::GetModuleHandle(NULL),IDS_DEL_RESULT_TEXT + desp_Idioma, NULL);
    pSubMenu->ModifyMenu(10,MF_BYPOSITION,pSubMenu->GetMenuItemID(10),resString);
    resString.LoadString(::GetModuleHandle(NULL),IDS_DEL_GRAFICA_TEXT + desp_Idioma, NULL);
    pSubMenu->ModifyMenu(11,MF_BYPOSITION,pSubMenu->GetMenuItemID(11),resString);

      CMenu* pPopup = SimMenu.GetSubMenu(0);
    if (AfxGetMainWnd()->IsKindOf(RUNTIME_CLASS(CBCGPMDIFrameWnd)))
    {
      CBCGPPopupMenu* pPopupMenu = new CBCGPPopupMenu;

      if (!pPopupMenu->Create(this, point.x, point.y, (HMENU)pPopup->m_hMenu, FALSE, TRUE))
        return;

      ((CBCGPMDIFrameWnd*)AfxGetMainWnd())->OnShowPopupMenu (pPopupMenu);
      UpdateDialogControls(this, FALSE);
    }
  }
```

```
  ////////////////////////////////
  //Mensajes de texto de la aplicacion
  //repartidos por todo el codigo
  ////////////////////////////////
  buffer.Format(GetResString(IDS_ERROR_FILEACCESS),ex.m_strFileName,szError);
      AfxMessageBox(buffer,MB_ICONERROR|MB_OK);
```

```
////////////////////////////////
//Ejemplo de inicializacion de textos de botones, cuadros, etc.
////////////////////////////////
// Controladores de mensajes de CGraph3D_Page1
BOOL CGraph3D_Page1::OnInitDialog()
{
  CDialog::OnInitDialog();

  //Puesta de los textos en el idioma correcto
  SetDlgItemText(IDC_P1_TITULO,GetResString(IDS_TITULO));
  SetDlgItemText(IDC_P1_ESTILO, GetResString(IDS_ESTILO));
  SetDlgItemText(IDC_P1_FONDO, GetResString(IDS_FONDO));
  SetDlgItemText(IDC_P1_MIN ,GetResString(IDS_MINIMO));
  SetDlgItemText(IDC_P1_MAX, GetResString(IDS_MAXIMO));
  SetDlgItemText(IDC_P1_RANGO, GetResString(IDS_RANGO));
  SetDlgItemText(IDC_P1_LEY_X, GetResString(IDS_LEY_X));
  SetDlgItemText(IDC_P1_LEY_Y, GetResString(IDS_LEY_Y));
  SetDlgItemText(IDC_P1_LEY_Z, GetResString(IDS_LEY_Z));
  SetDlgItemText(IDC_AUTORANGO, GetResString(IDS_AUTORANGO));
  SetDlgItemText(IDC_LOG, GetResString(IDS_LOG));

    [...]

  return FALSE;
}
```

B.4.4. Tabla de Elementos

```
////////////////////////////////
//elem_table.cpp
////////////////////////////////
//Al iniciar la tabla de elementos
BOOL CElemTable::OnInitDialog()
{
    CPreferences *prefs = &theApp.prefs;
  int m_iLang = prefs->GetLangx();
  switch (m_iLang)
  {
    case LANG_ENG:
      m_ElemTable.SetIdioma(1); //Ingl\'{e}s
    break;
    case LANG_SPA:
      m_ElemTable.SetIdioma(0); //Espa\~{n}ol
    default:
      m_ElemTable.SetIdioma(0); //Espa\~{n}ol
    break;
  }
    //ruta del fichero de los nombre de los elementos
  CString cad = GETPATH(DATA_PATH);
  m_ElemTable.SetRuta(cad);
}
```

```
void CElemTable::OnContextMenu(CWnd* /*pWnd*/, CPoint point)
{
  //llamada a la funci\'{o}n TablaElementos
  UINT uiCmd = m_ElemTable.GetElemento();
  if (uiCmd != -1)
  {
    //  ASSERT (IsValidIndex(uiCmd));
    *pElem = m_lstelems.GetAt(uiCmd);   //(**metodo referencia**)
    OnOK();
```

```
  }

}
```

B.4.5. Batch

```
////////////////////////////////
//Definici\'{o}n de los hilos en ficheros SimRunBatch.h y SimRunBatch.cpp
//Funciones mas importantes
////////////////////////////////

int CShellBatchThread::Run()
{
  SConfig scfg = *m_pScfg;
  CString strBuff;
  scfg.logfilename = m_pSimRun->m_strLogSSHPath;
  try
  {
    HANDLE handles[] = {*m_EvReadyToExec,*m_EvKillThread};
    DWORD dwResult = WaitForMultipleObjects(2,handles,FALSE,INFINITE);
    switch (dwResult)
    {
      case WAIT_OBJECT_0: //ready to exec
        break;
      case WAIT_OBJECT_0+1: //kill thread
      default:
        ASSERT(dwResult == WAIT_OBJECT_0+1); //no deber\'{\i}a ocurrir nunca
        m_EvKillThread->SetEvent();
        break;
    }
    if (dwResult == WAIT_OBJECT_0)
    {
      strBuff = "cd␣";
      strBuff += m_strRemotePath;
      ssh.Connect(scfg);
      ssh.SendCommand(strBuff);
      handles[0] = m_EvEndCommand;

      CString cmd;
      //Execute script
      if (!m_pSimRun->m_strScript.IsEmpty())
      {
        cmd = ExtractFileName(m_pSimRun->m_strScript);
        if (m_pScfg->shell_type == UNIX_SHELL)
          cmd = ".␣" + cmd;
        else if (m_pScfg->shell_type == CSH_SHELL)
          cmd = "source␣" + cmd;
        ssh.SendCommand(cmd);
      }
      //Extracts executable name from the executable path+name
      cmd = ExtractFileName(m_pSimRun->m_Version.m_strNameBin);
      if (m_pScfg->shell_type != WINDOWS_SHELL)
        cmd = "./" + cmd; //execute even in missconfigured unix systems
      //Add handler for OnDataReceived to log the simulator output
      ssh.__AddEventHandler_CSSHELL_OnDataReceived(this,OnDataReceived);
      //Execute the simulator
      strBuff = ((CSimulation*)m_pSimRun->m_hSim)->GetFileName();
      cmd += "␣" + strBuff + ".in";
      //Redirect stderr to a file
      cmd += "␣2>" + strBuff + "_stderr";
      cmd = "nohup␣" + cmd + "␣&";
      //Sent the command
```

```
      m_EvEndCommand.ResetEvent();
      ssh.SendCommand(cmd,0);
      dwResult = WaitForMultipleObjects(2,handles,FALSE,INFINITE);
      switch (dwResult)
      {
        case WAIT_OBJECT_0: //command finished
          break;
        case WAIT_OBJECT_0+1: //kill thread
        default:
          ASSERT(dwResult == WAIT_OBJECT_0+1); //no deber\'{\i}a ocurrir nunca
          m_EvKillThread->SetEvent();
          ssh.AbortCommand(true); //disconnect also
          break;
      }
    }
  }
  catch(CSSHException *ex)
  {
    m_EvKillThread->SetEvent();
    AfxMessageBox(ex->GetError(),MB_ICONERROR|MB_OK);
    ex->Delete();
  }
  m_EvSimFinished->SetEvent();
    theApp.m_pMainWnd->PostMessage(IIS_SIM_REFRESHOUTPUT,(WPARAM)m_pSimRun->m_hSim,(LPARAM)ID_OUTPUT_OUT);
  ssh.Disconnect();

  return ExitInstance();
}
```

```
int CSFTPBatchThread::Run()
{
  SConfig scfg = *m_pScfg;

  m_bFinishedOK = false;
  //establecer fichero de log
  scfg.logfilename = m_pSimRun->m_strLogSFTPPath;

  try
  {
    //conectar
    sftp.Connect(scfg);
    //obtener el directorio HOME
    sftp.GetRemotePath(m_strHomePath.GetBuffer(MAX_PATH),MAX_PATH);
    m_strHomePath.ReleaseBuffer();
    m_strHomePath.TrimRight("\\/");
    m_strHomePath.AppendChar('/');
    //subir tablas, simulador y fichero .in
    if (!IsEvSignaled(m_EvKillThread))
      UploadSimulator();
    //desconectar
    sftp.Disconnect();
  }
  catch(CSSHException *ex) //control de excepciones
  {
    //parar el temporizador y mostrar msg de error
    KillTimer(m_pMainWnd->GetSafeHwnd(),(UINT_PTR)&m_EvTimer);
    KillTimer(m_pMainWnd->GetSafeHwnd(),(UINT_PTR)&m_EvStdErrTimer);
    m_EvKillThread->SetEvent();
    AfxMessageBox(ex->GetError(),MB_ICONERROR|MB_OK);
    ex->Delete();
  }
  //Parar el temporizador y establecer evento de fin de simulaci\'{o}n
  KillTimer(m_pMainWnd->GetSafeHwnd(),(UINT_PTR)&m_EvTimer);
  KillTimer(m_pMainWnd->GetSafeHwnd(),(UINT_PTR)&m_EvStdErrTimer);
  m_pSimRun->OnFinish(m_bFinishedOK);
```

```
  return ExitInstance();
}
```

```
int CSFTPActualizarThread::Run()
{
  SConfig scfg = *m_pScfg;

  m_bFinishedOK = false;
  //establecer fichero de log
  scfg.logfilename = m_pSimRun->m_strLogSFTPPath;

  try
  {
    //conectar
    sftp.Connect(scfg);
    //obtener el directorio HOME
    sftp.GetRemotePath(m_strHomePath.GetBuffer(MAX_PATH),MAX_PATH);
    m_strHomePath.ReleaseBuffer();
    m_strHomePath.TrimRight("\\/");
    m_strHomePath.AppendChar('/');

    DownloadResults();
    //desconectar
    sftp.Disconnect();
  }
  catch(CSSHException *ex) //control de excepciones
  {
    //parar el temporizador y mostrar msg de error
    KillTimer(m_pMainWnd->GetSafeHwnd(),(UINT_PTR)&m_EvTimer);
    KillTimer(m_pMainWnd->GetSafeHwnd(),(UINT_PTR)&m_EvStdErrTimer);
    m_EvKillThread->SetEvent();
    AfxMessageBox(ex->GetError(),MB_ICONERROR|MB_OK);
    ex->Delete();
  }
  //Parar el temporizador y establecer evento de fin de simulaci\'{o}n
  KillTimer(m_pMainWnd->GetSafeHwnd(),(UINT_PTR)&m_EvTimer);
  KillTimer(m_pMainWnd->GetSafeHwnd(),(UINT_PTR)&m_EvStdErrTimer);

  return ExitInstance();
}
```

B.4.6. Control de Versiones

```
//////////////////////////////////
//Son cambios menores sobre el c\'{o}digo existente
//VersionList.cpp
//////////////////////////////////
void CVersion::Serialize(CArchive& ar)
{
  CObject::Serialize(ar);
  char Directorio [300];
  int i;
  //Con esto conseguimos el directorio del ejecutable
  GetModuleFileName(GetModuleHandle(NULL),Directorio,300);
  for (i=strlen(Directorio);Directorio[i]!='\\';i--)
    Directorio[i]=0;
  CString Cad = Directorio;
  Cad.Replace("iisgui.exe",""); //eliminamos el ejecutable

  if (ar.IsStoring())
  {
    CString temp1 = m_strNameBin.MakeUpper();
```

```
      CString temp2 = m_strScript.MakeUpper();
      temp1.Replace(Cad.MakeUpper(),"");
      temp2.Replace(Cad.MakeUpper(),"");

      m_strNameBin = temp1.MakeLower();
      m_strScript = temp2.MakeLower();

      ar << m_strName << m_strDescription << m_strVersionNumber << m_strArchitecture
        << m_strNameBin << m_strScript;
    }
    else
    {
      ar >> m_strName >> m_strDescription >> m_strVersionNumber >> m_strArchitecture
        >> m_strNameBin >> m_strScript;

    }
    m_strNameBin = Cad.MakeLower() + m_strNameBin;
    m_strScript = Cad.MakeLower() + m_strScript;

  }
```

```
bool CVersionList::Compare(CVersion* const version1, CVersion* const version2)
{
  if (!version1->m_strArchitecture.CompareNoCase(version2->m_strArchitecture))
    if (!version1->m_strDescription.CompareNoCase(version2->m_strDescription))
      if (!version1->m_strName.CompareNoCase(version2->m_strName))
        if (!version1->m_strNameBin.CompareNoCase(version2->m_strNameBin))
          if (!version1->m_strVersionNumber.CompareNoCase(version2->m_strVersionNumber))
            if (!version1->m_strScript.CompareNoCase(version2->m_strScript))
              return true;
  return false;
}
```

```
//////////////////////////////////
//Para abrir la p\'{a}gina correspondiente
//MainFrm.cpp
//////////////////////////////////
void CMainFrame::OnHerramientasControldeversiones()
{
  CPrefsPSht2 prefsDlg2;
  if (prefsDlg2.DoModal()==IDOK)
    theApp.prefs.SavePreferences();
}
```

B.4.7. Lista de Simulaciones

```
//////////////////////////////////
//CSimListCtrl.cpp
//////////////////////////////////

void CSimListCtrl::AddColumn()
{
  //Tenemos 6 fijas y el resto dependientes de variables
    InsertColumn(0,GetResString(IDS_PROP_SIMNAME),LVCFMT_LEFT,110,0);
  InsertColumn(1,GetResString(IDS_STATUS),LVCFMT_LEFT,80,1);
  InsertColumn(2,GetResString(IDS_SIMINIT_HOST),LVCFMT_LEFT,110,2);
  InsertColumn(3,GetResString(IDS_SIMINIT_VERSION),LVCFMT_LEFT,110,3);
  InsertColumn(4,GetResString(IDS_SIMINIT_STARTTIME),LVCFMT_LEFT,105,4);
  InsertColumn(5,GetResString(IDS_SIMINIT_FINISHTIME),LVCFMT_LEFT,105,5);
  if (theApp.prefs.simtable.m_act_WinWidth)
    InsertColumn(6,GetResString(IDS_PROP_WINWIDTH),LVCFMT_CENTER,100,6);
```

```
    if (theApp.prefs.simtable.m_act_WinHeight)
      InsertColumn(7,GetResString(IDS_PROP_WINHEIGHT),LVCFMT_CENTER,100,7);
    if (theApp.prefs.simtable.m_act_ImplantArea)
      InsertColumn(8,GetResString(IDS_PROP_IMPLAREA),LVCFMT_CENTER,110,8);
20  if (theApp.prefs.simtable.m_act_ErrorTha)
      InsertColumn(9,GetResString(IDS_PROP_ERRTHA),LVCFMT_CENTER,70,9);
    if (theApp.prefs.simtable.m_act_ErrorPhi)
      InsertColumn(10,GetResString(IDS_PROP_ERRPHI),LVCFMT_CENTER,70,10);
    if (theApp.prefs.simtable.m_act_EjeProf)
      InsertColumn(11,GetResString(IDS_PROP_DEPTHAXIS),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_EjeVert)
      InsertColumn(12,GetResString(IDS_PROP_VERTICALAXIS),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_Capa)
      InsertColumn(13,GetResString(IDS_PROP_LAYER),LVCFMT_CENTER,100,10);
30  if (theApp.prefs.simtable.m_act_Energia)
      InsertColumn(14,GetResString(IDS_PROP_SUBIMPLANT_ENERGY),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_Divergencia)
      InsertColumn(15,GetResString(IDS_PROP_SUBIMPLANT_DIVERGENCE),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_NumIones)
      InsertColumn(16,GetResString(IDS_PROP_SUBIMPLANT_IONNUM),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_Dosis)
      InsertColumn(17,GetResString(IDS_PROP_SUBIMPLANT_DOSE),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_Temperatura)
      InsertColumn(18,GetResString(IDS_PROP_SUBIMPLANT_TEMP),LVCFMT_CENTER,100,10);
40  if (theApp.prefs.simtable.m_act_Ruido)
      InsertColumn(19,GetResString(IDS_PROP_SUBIMPLANT_NOISE),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_Recombinacion)
      InsertColumn(20,GetResString(IDS_PROP_DAMAGE_RECOMB),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_KinPease)
      InsertColumn(21,GetResString(IDS_PROP_DAMAGE_KINCHIN),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_DenCritica)
      InsertColumn(22,GetResString(IDS_PROP_DAMAGE_DENSITY),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_EnergDesp)
      InsertColumn(23,GetResString(IDS_PROP_DAMAGE_ENERDESP),LVCFMT_CENTER,100,10);
50  if (theApp.prefs.simtable.m_act_EnergCorte)
      InsertColumn(24,GetResString(IDS_PROP_DAMAGE_ENERCUT),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_DanPrevio)
      InsertColumn(25,GetResString(IDS_PROP_DAMAGE_PREVDAMAGE),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_ModFrenado)
      InsertColumn(26,GetResString(IDS_PROP_POTENTIAL_MODEL),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_CurvaIon)
      InsertColumn(27,GetResString(IDS_PROP_POTENTIAL_CURVE),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_DisperE)
      InsertColumn(28,GetResString(IDS_PROP_ADVANCED_ENERGYDISP),LVCFMT_CENTER,100,10);
60  if (theApp.prefs.simtable.m_act_Sigma)
      InsertColumn(29,GetResString(IDS_PROP_ADVANCED_SIGMA),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_Porcentaje)
      InsertColumn(30,GetResString(IDS_PROP_ADVANCED_PERCENT),LVCFMT_CENTER,100,10);
      if (theApp.prefs.simtable.m_act_DivergHaz)
      InsertColumn(31,GetResString(IDS_PROP_ADVANCED_HAZDIV),LVCFMT_CENTER,100,10);
      if (theApp.prefs.simtable.m_act_DependTemp)
      InsertColumn(32,GetResString(IDS_PROP_ADVANCED_TEMP),LVCFMT_CENTER,100,10);
      if (theApp.prefs.simtable.m_act_NivelSimulacion)
      InsertColumn(33,GetResString(IDS_PROP_ADVANCED_SIMLEVEL),LVCFMT_CENTER,100,10);
70    if (theApp.prefs.simtable.m_act_Aleatoriza)
      InsertColumn(34,GetResString(IDS_PROP_ADVANCED_RANDOMIZE),LVCFMT_CENTER,100,10);
      if (theApp.prefs.simtable.m_act_Semillas)
      InsertColumn(35,GetResString(IDS_PROP_ADVANCED_SEED),LVCFMT_CENTER,100,10);
      if (theApp.prefs.simtable.m_act_PosPartida)
      InsertColumn(36,GetResString(IDS_PROP_ADVANCED_STARTPOS),LVCFMT_CENTER,100,10);
      if (theApp.prefs.simtable.m_act_UmbralReducRuido)
      InsertColumn(37,GetResString(IDS_PROP_ADVANCED_NOISEUMBRAL),LVCFMT_CENTER,100,10);
      if (theApp.prefs.simtable.m_act_IntervaloReducRuido)
      InsertColumn(38,GetResString(IDS_PROP_ADVANCED_NOISEINTERVAL),LVCFMT_CENTER,100,10);
80    if (theApp.prefs.simtable.m_act_TipoReducRuido)
      InsertColumn(39,GetResString(IDS_PROP_ADVANCED_NOISETYPE),LVCFMT_CENTER,100,10);
```

```
    if (theApp.prefs.simtable.m_act_UmbralERRSup)
    InsertColumn(40,GetResString(IDS_PROP_ADVANCED_ENERGYUMBRAL),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_PorcentajeProfRRSup)
    InsertColumn(41,GetResString(IDS_PROP_ADVANCED_PERCENTPROF),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_RadIntegracion)
    InsertColumn(42,GetResString(IDS_PROP_ADVANCED2_RADIUS),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_EUmbral)
    InsertColumn(43,GetResString(IDS_PROP_ADVANCED2_ENERGY),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_LimGHI)
    InsertColumn(44,GetResString(IDS_PROP_ADVANCED2_GHI),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_DisSim1)
    InsertColumn(45,GetResString(IDS_PROP_ADVANCED2_DISTANCE)+"1",LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_DisSim2)
    InsertColumn(46,GetResString(IDS_PROP_ADVANCED2_DISTANCE)+"2",LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_MaxIteraciones)
    InsertColumn(47,GetResString(IDS_PROP_ADVANCED2_MAXITER),LVCFMT_CENTER,100,10);
    if (theApp.prefs.simtable.m_act_MaxProfundidad)
    InsertColumn(48,GetResString(IDS_PROP_ADVANCED2_MAXDEPTH),LVCFMT_CENTER,100,10);
}
```

```
void CSimListCtrl::RefreshSim(CSimulation* sim)
{
  //recorremos todas las simulaciones actualizando los datos
  POSITION pos;
  if (theApp.lstsims.GetSimCount())
  {
    pos = theApp.lstsims.GetHeadPosition();
    while (pos)
    {
      sim = theApp.lstsims.GetNext(pos);

      LVFINDINFO find;
      find.flags = LVFI_PARAM;
      find.lParam = (LPARAM)sim;
      sint32 itemnr = FindItem(&find);
      CLayer* layer;
      POSITION pos;
      CSubImp* subimp;

      if (itemnr == (-1))
        return;

      CString buffer;
      CHeaderCtrl *pHeaderCtrl = GetHeaderCtrl();
      int iCount = pHeaderCtrl->GetItemCount();

      bool cambio=theApp.prefs.simtable.m_Change;

      if(theApp.prefs.simtable.m_Change || (iCount==0))
        cambio=true;
      if (cambio)
        //eliminamos las columnas
        for (int iCurrent=0;iCurrent < iCount;iCurrent++)
          DeleteColumn(0); //siempre 0 porque es la 1  columna la que eliminamos

      int contador=0;;
      //insertamos las columnas que queramos
      if (cambio)
        AddColumn();
      //name
      buffer = sim->m_strSimName;
      //set icon
      uint8 image;
      switch (sim->m_RunProps->GetSimState())
      {
        case RS_RUNNING:
```

```
            image = 1;
            break;
          case RS_FINISHED:
            image = 2;
            break;
          case RS_BATCH:
            image = 1;
            break;
          case RS_STOPPED:
          case RS_NONE:
          default:
            image = 0;
            break;
        }
        SetItem(itemnr,0,LVIF_IMAGE,0,image,0,0,0);
        SetItemText(itemnr,contador,buffer);
        contador++;
        // end name
        //state
        buffer = sim->m_RunProps->GetSimStateCaption();
        SetItemText(itemnr,contador,buffer);
        contador++;
        //end state
        //host
        buffer = sim->m_RunProps->GetSimHost();
        SetItemText(itemnr,contador,buffer);
        contador++;
        //end host
        //version
        buffer = sim->m_RunProps->m_Version.m_strName;
        SetItemText(itemnr,contador,buffer);
        contador++;
        //end version
        //start time
        buffer = (sim->m_RunProps->m_StartTime == 0) ? ""
                  : sim->m_RunProps->m_StartTime.Format("%c");
        SetItemText(itemnr,contador,buffer);
        contador++;
        //end start time
        //finish time
        buffer = (sim->m_RunProps->m_FinishTime == 0) ? ""
                  : sim->m_RunProps->m_ElapsedTime.Format("%Dd%Hh%Mm%Ss");
        SetItemText(itemnr,contador,buffer);
        contador++;
        //end finish time
        //window width
        if (theApp.prefs.simtable.m_act_WinWidth)
        {
          buffer.Format("%g␣{\AA}",sim->m_dWinWidth);
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        //end window width
        //window height
        if (theApp.prefs.simtable.m_act_WinHeight)
        {
          buffer.Format("%g␣{\AA}",sim->m_dWinHeight);
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        //end window height
        //impalnt area
        if (theApp.prefs.simtable.m_act_ImplantArea)
        {
          buffer.Format("%g␣{\AA} ",sim->m_dImplantArea);
          SetItemText(itemnr,contador,buffer);
```

```
        contador++;
      }
      //end implant Area
      //Error Tha
      if (theApp.prefs.simtable.m_act_ErrorTha)
      {
        buffer.Format("%g␣{\AA} ",sim->m_dErrTha);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      //end Error Tha
      //Error Phi
      if (theApp.prefs.simtable.m_act_ErrorPhi)
      {
        buffer.Format("%g␣{\AA} ",sim->m_dErrPhi);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      //end Error Phi
      if (theApp.prefs.simtable.m_act_EjeProf)
      {
         buffer.Format("(%d,%d,%d)",sim->m_iEjeProfX,sim->m_iEjeProfY,sim->m_iEjeProfZ);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_EjeVert)
      {
         buffer.Format("(%d,%d,%d)",sim->m_iEjeVertX,sim->m_iEjeVertY,sim->m_iEjeVertZ);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_Capa)
      {
        if (sim->m_lstLayerList.GetCount())
        {
          buffer.Format("");
          pos = sim->m_lstLayerList.GetHeadPosition();
          layer = sim->m_lstLayerList.GetNext(pos);
          buffer.Format(buffer + "%s",layer->m_strLayerName);
          while (pos)
          {
            layer = sim->m_lstLayerList.GetNext(pos);
            buffer.Format(buffer + ",␣%s",layer->m_strLayerName);
          }
        }
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_Energia)
      {
        if (sim->m_lstSubImpList.GetCount())
        {
          buffer.Format("");
          POSITION pos = sim->m_lstSubImpList.GetHeadPosition();
          subimp = sim->m_lstSubImpList.GetNext(pos);
          buffer.Format(buffer + "%g",subimp->m_dEnergy);
          while (pos)
          {
            subimp = sim->m_lstSubImpList.GetNext(pos);
            buffer.Format(buffer + ",␣%g",subimp->m_dEnergy);
          }
        }
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_Divergencia)
```

```
        {
180       if (sim->m_lstSubImpList.GetCount())
          {
            buffer.Format("");
            POSITION pos = sim->m_lstSubImpList.GetHeadPosition();
            subimp = sim->m_lstSubImpList.GetNext(pos);
            buffer.Format(buffer + "%g",subimp->m_dDivergence);
            while (pos)
            {
              subimp = sim->m_lstSubImpList.GetNext(pos);
              buffer.Format(buffer + ", %g",subimp->m_dDivergence);
190         }
          }
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_NumIones)
        {
          if (sim->m_lstSubImpList.GetCount())
          {
            buffer.Format("");
200         POSITION pos = sim->m_lstSubImpList.GetHeadPosition();
            subimp = sim->m_lstSubImpList.GetNext(pos);
            buffer.Format(buffer + "%li",subimp->m_lNumIons);
            while (pos)
            {
              subimp = sim->m_lstSubImpList.GetNext(pos);
              buffer.Format(buffer + ", %li",subimp->m_lNumIons);
            }
          }
          SetItemText(itemnr,contador,buffer);
210       contador++;
        }
        if (theApp.prefs.simtable.m_act_Dosis)
        {
          if (sim->m_lstSubImpList.GetCount())
          {
            buffer.Format("");
            POSITION pos = sim->m_lstSubImpList.GetHeadPosition();
            subimp = sim->m_lstSubImpList.GetNext(pos);
            buffer.Format(buffer + "%g",subimp->m_dDose);
220         while (pos)
            {
              subimp = sim->m_lstSubImpList.GetNext(pos);
              buffer.Format(buffer + ", %g",subimp->m_dDose);
            }
          }
          buffer.Format(buffer + " (cm^-2)");
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
230     if (theApp.prefs.simtable.m_act_Temperatura)
        {
          if (sim->m_lstSubImpList.GetCount())
          {
            buffer.Format("");
            POSITION pos = sim->m_lstSubImpList.GetHeadPosition();
            subimp = sim->m_lstSubImpList.GetNext(pos);
            buffer.Format(buffer + "%g",subimp->m_dTemperature);
            while (pos)
            {
240           subimp = sim->m_lstSubImpList.GetNext(pos);
              buffer.Format(buffer + ", %g",subimp->m_dTemperature);
            }
          }
          buffer.Format(buffer + " K");
```

```
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_Ruido)
      {
250     if (sim->m_lstSubImpList.GetCount())
        {
          buffer.Format("");
          POSITION pos = sim->m_lstSubImpList.GetHeadPosition();
          subimp = sim->m_lstSubImpList.GetNext(pos);
          buffer.Format(buffer + "%g",subimp->m_iNoiseLevel);
          while (pos)
          {
            subimp = sim->m_lstSubImpList.GetNext(pos);
            buffer.Format(buffer + ", %i",subimp->m_iNoiseLevel);
260       }
        }
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_Recombinacion)
      {
        buffer.Format("%g",sim->m_dRecombFact);
        SetItemText(itemnr,contador,buffer);
        contador++;
270   }
      if (theApp.prefs.simtable.m_act_KinPease)
      {
        buffer.Format("%g",sim->m_dKinchin);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_DenCritica)
      {
        buffer.Format("%g",sim->m_dAmorDensity);
280     SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_EnergDesp)
      {
        buffer.Format("%g",sim->m_dDespEnergy);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_EnergCorte)
290   {
        buffer.Format("%g",sim->m_dCutEnergy);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_DanPrevio)
      {
        buffer.Format("%s",sim->m_strPrevDamage);
        SetItemText(itemnr,contador,buffer);
        contador++;
300   }
      if (theApp.prefs.simtable.m_act_ModFrenado)
      {
        switch (sim->m_iBreakModel)
        {
          case NONE:
            buffer.Format("%s",GetResString(IDS_PROP_OPT_NONE));
            break;
          case BRANDT_KITTAGAWA_MODEL:
            buffer.Format("%s",GetResString(IDS_PROP_OPT_KITTAGAWA));
310         break;
```

```
            case OUR_MODEL:
              buffer.Format("%s",GetResString(IDS_PROP_OPT_OURMODEL));
              break;
          }
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_CurvaIon)
        {
320       switch (sim->m_iIonCurve)
          {
            case BK_IONIZATION:
              buffer.Format("%s",GetResString(IDS_PROP_OPT_BKION));
              break;
            case CGJ_IONIZATION:
              buffer.Format("%s",GetResString(IDS_PROP_OPT_CGJION));
              break;
            case MP_IONIZATION:
              buffer.Format("%s",GetResString(IDS_PROP_OPT_MPION));
330           break;
          }
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_DisperE)
        {
          buffer.Format("%i",sim->m_iEnerDisp);
          SetItemText(itemnr,contador,buffer);
          contador++;
340     }
        if (theApp.prefs.simtable.m_act_Sigma)
        {
          buffer.Format("%g",sim->m_dSigma);
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_Porcentaje)
        {
          buffer.Format("%g",sim->m_dPercentage);
350       SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_DivergHaz)
        {
          buffer.Format("%i",sim->m_iHazDiverg);
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_DependTemp)
360     {
          buffer.Format("%i",sim->m_bTempDependence);
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_NivelSimulacion)
        {
          switch (sim->m_iSimLevel)
          {
            case 0:
370           buffer.Format("%s",GetResString(IDS_PROP_OPT_FOLLOWCASCADES));
              break;
            case 1:
              buffer.Format("%s",GetResString(IDS_PROP_OPT_FOLLOWMAINPATH));
              break;
          }
          SetItemText(itemnr,contador,buffer);
```

```
        contador++;
      }
      if (theApp.prefs.simtable.m_act_Aleatoriza)
      {
        buffer.Format("%i",sim->m_bRandomize);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_Semillas)
      {
        buffer.Format("(%i, %i)",sim->m_iSeed1, sim->m_iSeed2);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_PosPartida)
      {
        buffer.Format("(%g, %g, %g) {\AA}",sim->m_dStartPosX,
                    sim->m_dStartPosY,sim->m_dStartPosZ);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_UmbralReducRuido)
      {
        buffer.Format("%i",sim->m_iNoiseThreshold);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_IntervaloReducRuido)
      {
        buffer.Format("%i",sim->m_iNoiseInterval);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_TipoReducRuido)
      {
        buffer.Format("%i",sim->m_iNoiseReductType);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_UmbralERRSup)
      {
        buffer.Format("%g",sim->m_dNoiseThresholdSurf);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_PorcentajeProfRRSup)
      {
        buffer.Format("%g",sim->m_dNoisePercenSurf);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_RadIntegracion)
      {
        buffer.Format("%g {\AA}",sim->m_dInterRadio);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_EUmbral)
      {
        buffer.Format("%g eV",sim->m_dThresholdEnergy);
        SetItemText(itemnr,contador,buffer);
        contador++;
      }
      if (theApp.prefs.simtable.m_act_LimGHI)
      {
        buffer.Format("%g",sim->m_dGHILimit);
```

```
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_DisSim1)
        {
          buffer.Format("%g",sim->m_dSimultDist1);
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_DisSim2)
        {
          buffer.Format("%g",sim->m_dSimultDist2);
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_MaxIteraciones)
        {
          buffer.Format("%i",sim->m_iMaxIter);
          SetItemText(itemnr,contador,buffer);
          contador++;
        }
        if (theApp.prefs.simtable.m_act_MaxProfundidad)
        {
          buffer.Format("%g␣{\AA}",sim->m_dMaxDepth);
          SetItemText(itemnr,contador,buffer);
          contador++;
        }

        theApp.prefs.simtable.m_Change = false;

        /////////////////
        //Se ordena tambi\'{e}n el \'{a}rbol de simulaciones
        int sortItem = theApp.prefs.simtable.GetColumnSortItem();
        bool sortAscending = theApp.prefs.simtable.GetColumnSortAscending();
        AfxGetMainWnd()->PostMessage(IIS_SIM_SORT,(WPARAM)SortProc,
                  (LPARAM)sortItem + (sortAscending ? 0:100));

      }
    }
  }
```

```
//CSimTreeCtrl.cpp

void CSimTreeCtrl::SetBitmaps(UINT StatusId)
{
  // create Imagelists
  // 16x16 Pixel, Blanco color trasparente
  m_StatusImageList.Create(
    IDB_CHECKBOX, 13, 3, RGB (255,255,255));

  // and set the imagelists
  SetImageList(&m_StatusImageList, TVSIL_STATE );
}
```

```
void CSimTreeCtrl::changeItemState(HTREEITEM hItem)
{
  UINT OldState;

  CSimulation* sim;


  if (hItem!=NULL)
    {
    // ... Calcula el nuevo estado usando un BitMask
    OldState = GetItemState(hItem, TVIS_STATEIMAGEMASK);
```

```
    switch (int(OldState>>12))
    {
      case 1: //Caso imagen general
        SetItemState(hItem,INDEXTOSTATEIMAGEMASK (0), TVIS_STATEIMAGEMASK);
        //MessageBox("Pongo un 0");
        break;

      case 2: //caso imagen checkada
        SetItemState(hItem,INDEXTOSTATEIMAGEMASK (2), TVIS_STATEIMAGEMASK);
        theApp.prefs.simtable.marcados--;
        ActualizarLista(hItem);
        theApp.prefs.simtable.m_Change=true;

        //hacemos que refresque la lista de simulaciones
        AfxGetMainWnd()->SendMessage(IIS_SIM_CHANGE_LIST,0,(LPARAM)sim);

        break;

      default:  //caso no checkado (Case 12)
        SetItemState(hItem,INDEXTOSTATEIMAGEMASK (1), TVIS_STATEIMAGEMASK);
        theApp.prefs.simtable.marcados++;
        ActualizarLista(hItem);
        theApp.prefs.simtable.m_Change=true;

        //hacemos que refresque la lista de simulaciones
        AfxGetMainWnd()->SendMessage(IIS_SIM_CHANGE_LIST,0,(LPARAM)sim);

        break;
    }
  }
}
```

```
void CSimTreeCtrl::OnNMLclick(NMHDR* pNMHDR, LRESULT* pResult)
{

  //NM_TREEVIEW* pNMTreeView = (NM_TREEVIEW*)pNMHDR;
  CPoint pt;
  CRect rt;
  GetCursorPos(&pt);
  ScreenToClient(&pt);
  UINT flag;
  HTREEITEM hItem = HitTest(pt, &flag);
  if ((flag & TVHT_ONITEMSTATEICON) != TVHT_ONITEMSTATEICON)
    return;
  //Aqui se cambia la imagen segun correponda
  changeItemState(hItem);
}
```

```
void CSimTreeCtrl::ActualizarLista(HTREEITEM hItem)
{
  //esto funciona para poner las variables al valor correcto

  CString nomItem = GetItemText(hItem);
  //CSimulation* sim = (CSimulation*)GetItemData(GetRootItemEx(GetSelectedItem()));

  if (theApp.prefs.simtable.marcados > MAX_MARCADOS)
  {
    //Caso imagen no marcada, el resto de casos no me interesan
    SetItemState(hItem,INDEXTOSTATEIMAGEMASK (2), TVIS_STATEIMAGEMASK);
    theApp.prefs.simtable.marcados--;
    MessageBox(GetResString(IDS_MAX_MARCADOS),NULL,MB_ICONWARNING);
    return;
  }
  //hacer esto para todas las opciones posibles
  //Principal
  if (StrStr(nomItem, GetResString(IDS_PROP_WINWIDTH)))
```

```
{
  theApp.prefs.simtable.m_act_WinWidth = !theApp.prefs.simtable.m_act_WinWidth;
  return;
}
if (StrStr(nomItem, GetResString(IDS_PROP_WINHEIGHT)))
{
  theApp.prefs.simtable.m_act_WinHeight = !theApp.prefs.simtable.m_act_WinHeight;
  return;
}
if (StrStr(nomItem, GetResString(IDS_PROP_IMPLAREA)))
{
  theApp.prefs.simtable.m_act_ImplantArea = !theApp.prefs.simtable.m_act_ImplantArea;
  return;
}
if (StrStr(nomItem, GetResString(IDS_PROP_ERRTHA)))
{
  theApp.prefs.simtable.m_act_ErrorTha = !theApp.prefs.simtable.m_act_ErrorTha;
  return;
}
if (StrStr(nomItem, GetResString(IDS_PROP_ERRPHI)))
{
  theApp.prefs.simtable.m_act_ErrorPhi = !theApp.prefs.simtable.m_act_ErrorPhi;
  return;
}
if (StrStr(nomItem, GetResString(IDS_PROP_DEPTHAXIS)))
{
  theApp.prefs.simtable.m_act_EjeProf = !theApp.prefs.simtable.m_act_EjeProf;
  return;
}
if (StrStr(nomItem, GetResString(IDS_PROP_VERTICALAXIS)))
{
  theApp.prefs.simtable.m_act_EjeVert = !theApp.prefs.simtable.m_act_EjeVert;
  return;
}
//Material blanco
if (StrStr(nomItem, GetResString(IDS_PROP_LAYER)))
{
  theApp.prefs.simtable.m_act_Capa = !theApp.prefs.simtable.m_act_Capa;
  CSimulation * sim = GetSelectedSim();
  RefreshSim(sim);
  return;
}
//Implantes
//La primera y la seguna pertenecen a Avanzado, pero se pone aqui para que la distinga
if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_ENERGYDISP)))
{
  theApp.prefs.simtable.m_act_DisperE = !theApp.prefs.simtable.m_act_DisperE;
  return;
}
if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_ENERGYUMBRAL)))
{
  theApp.prefs.simtable.m_act_UmbralERRSup = !theApp.prefs.simtable.m_act_UmbralERRSup;
  return;
}
//pertenece a Avanzado2, pero se pone aqui para que la distinga
if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED2_ENERGY)))
{
  theApp.prefs.simtable.m_act_EUmbral = !theApp.prefs.simtable.m_act_EUmbral;
  return;
}
if (StrStr(nomItem, GetResString(IDS_PROP_SUBIMPLANT_ENERGY)))
{
  theApp.prefs.simtable.m_act_Energia = !theApp.prefs.simtable.m_act_Energia;
  CSimulation * sim = GetSelectedSim();
  RefreshSim(sim);
  return;
```

```
85  }
    //pertenece a Avanzado, pero se pone aqui para que la distinga
    if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_HAZDIV)))
    {
      theApp.prefs.simtable.m_act_DivergHaz = !theApp.prefs.simtable.m_act_DivergHaz;
      return;
    }
    if (StrStr(nomItem, GetResString(IDS_PROP_SUBIMPLANT_DIVERGENCE)))
    {
      theApp.prefs.simtable.m_act_Divergencia = !theApp.prefs.simtable.m_act_Divergencia;
95    CSimulation * sim = GetSelectedSim();
      RefreshSim(sim);
      return;
    }
    if (StrStr(nomItem, GetResString(IDS_PROP_SUBIMPLANT_IONNUM)))
    {
      theApp.prefs.simtable.m_act_NumIones = !theApp.prefs.simtable.m_act_NumIones;
      CSimulation * sim = GetSelectedSim();
      RefreshSim(sim);
      return;
105 }
    if (StrStr(nomItem, GetResString(IDS_PROP_SUBIMPLANT_DOSE)))
    {
      theApp.prefs.simtable.m_act_Dosis = !theApp.prefs.simtable.m_act_Dosis;
      CSimulation * sim = GetSelectedSim();
      RefreshSim(sim);
      return;
    }
    //pertenece a Avanzado, pero se pone aqui para que la distinga
    if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_TEMP)))
115 {
      theApp.prefs.simtable.m_act_DependTemp = !theApp.prefs.simtable.m_act_DependTemp;
      return;
    }
    if (StrStr(nomItem, GetResString(IDS_PROP_SUBIMPLANT_TEMP)))
    {
      theApp.prefs.simtable.m_act_Temperatura = !theApp.prefs.simtable.m_act_Temperatura;
      CSimulation * sim = GetSelectedSim();
      RefreshSim(sim);
      return;
125 }
    //pertenece a Avanzado, pero se pone aqui para que la distinga
    if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_NOISEUMBRAL)))
    {
      theApp.prefs.simtable.m_act_UmbralReducRuido =
              !theApp.prefs.simtable.m_act_UmbralReducRuido;
      return;
    }
    //pertenece a Avanzado, pero se pone aqui para que la distinga
    if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_NOISEINTERVAL)))
135 {
      theApp.prefs.simtable.m_act_IntervaloReducRuido =
              !theApp.prefs.simtable.m_act_IntervaloReducRuido;
      return;
    }
    //pertenece a Avanzado, pero se pone aqui para que la distinga
    if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_NOISETYPE)))
    {
      theApp.prefs.simtable.m_act_TipoReducRuido =
              !theApp.prefs.simtable.m_act_TipoReducRuido;
145   return;
    }
    if (StrStr(nomItem, GetResString(IDS_PROP_SUBIMPLANT_NOISE)))
    {
      theApp.prefs.simtable.m_act_Ruido = !theApp.prefs.simtable.m_act_Ruido;
      CSimulation * sim = GetSelectedSim();
```

```
      RefreshSim(sim);
      return;
    }
    //Da\~{n}ado
155 if (StrStr(nomItem, GetResString(IDS_PROP_DAMAGE_RECOMB)))
    {
      theApp.prefs.simtable.m_act_Recombinacion =
              !theApp.prefs.simtable.m_act_Recombinacion;
      return;
    }
    if (StrStr(nomItem, GetResString(IDS_PROP_DAMAGE_KINCHIN)))
    {
      theApp.prefs.simtable.m_act_KinPease = !theApp.prefs.simtable.m_act_KinPease;
      return;
165 }
    if (StrStr(nomItem, GetResString(IDS_PROP_DAMAGE_DENSITY)))
    {
      theApp.prefs.simtable.m_act_DenCritica = !theApp.prefs.simtable.m_act_DenCritica;
      return;
    }
    if (StrStr(nomItem, GetResString(IDS_PROP_DAMAGE_ENERDESP)))
    {
      theApp.prefs.simtable.m_act_EnergDesp = !theApp.prefs.simtable.m_act_EnergDesp;
      return;
175 }
    if (StrStr(nomItem, GetResString(IDS_PROP_DAMAGE_ENERCUT)))
    {
      theApp.prefs.simtable.m_act_EnergCorte = !theApp.prefs.simtable.m_act_EnergCorte;
      return;
    }
    if (StrStr(nomItem, GetResString(IDS_PROP_DAMAGE_PREVDAMAGE)))
    {
      theApp.prefs.simtable.m_act_DanPrevio = !theApp.prefs.simtable.m_act_DanPrevio;
      return;
185 }
    //Frenado
    if (StrStr(nomItem, GetResString(IDS_PROP_POTENTIAL_MODEL)))
    {
      theApp.prefs.simtable.m_act_ModFrenado = !theApp.prefs.simtable.m_act_ModFrenado;
      return;
    }
    if (StrStr(nomItem, GetResString(IDS_PROP_POTENTIAL_CURVE)))
    {
      theApp.prefs.simtable.m_act_CurvaIon = !theApp.prefs.simtable.m_act_CurvaIon;
195   return;
    }
    //avanzado
    if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_SIGMA)))
    {
      theApp.prefs.simtable.m_act_Sigma = !theApp.prefs.simtable.m_act_Sigma;
      return;
    }
    //lleva este orden porque es la misma palabra
    if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_PERCENTPROF)))
205 {
      theApp.prefs.simtable.m_act_PorcentajeProfRRSup =
              !theApp.prefs.simtable.m_act_PorcentajeProfRRSup;
      return;
    }
    if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_PERCENT)))
    {
      theApp.prefs.simtable.m_act_Porcentaje = !theApp.prefs.simtable.m_act_Porcentaje;
      return;
    }
215 if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_SIMLEVEL)))
    {
```

```
    theApp.prefs.simtable.m_act_NivelSimulacion =
            !theApp.prefs.simtable.m_act_NivelSimulacion;
    return;
  }
  if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_RANDOMIZE)))
  {
    theApp.prefs.simtable.m_act_Aleatoriza = !theApp.prefs.simtable.m_act_Aleatoriza;
    return;
  }
  if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_SEED)))
  {
    theApp.prefs.simtable.m_act_Semillas = !theApp.prefs.simtable.m_act_Semillas;
    return;
  }
  if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED_STARTPOS)))
  {
    theApp.prefs.simtable.m_act_PosPartida = !theApp.prefs.simtable.m_act_PosPartida;
    return;
  }

  //avanzado2
  if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED2_RADIUS)))
  {
    theApp.prefs.simtable.m_act_RadIntegracion =
            !theApp.prefs.simtable.m_act_RadIntegracion;
    return;
  }
  if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED2_GHI)))
  {
    theApp.prefs.simtable.m_act_LimGHI = !theApp.prefs.simtable.m_act_LimGHI;
    return;
  }
  if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED2_DISTANCE)+"1"))
  {
    theApp.prefs.simtable.m_act_DisSim1 = !theApp.prefs.simtable.m_act_DisSim1;
    return;
  }
  if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED2_DISTANCE)+"2"))
  {
    theApp.prefs.simtable.m_act_DisSim2 = !theApp.prefs.simtable.m_act_DisSim2;
    return;
  }
  if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED2_MAXITER)))
  {
    theApp.prefs.simtable.m_act_MaxIteraciones =
            !theApp.prefs.simtable.m_act_MaxIteraciones;
    return;
  }
  if (StrStr(nomItem, GetResString(IDS_PROP_ADVANCED2_MAXDEPTH)))
  {
    theApp.prefs.simtable.m_act_MaxProfundidad =
            !theApp.prefs.simtable.m_act_MaxProfundidad;
    return;
  }
  MessageBox(GetResString(IDS_ERROR_NADA),"Error");
}
```

```
void CSimTreeCtrl::OnFocus(NMHDR *pNMHDR, LRESULT *pResult)
{
  CSimulation * sim = GetSelectedSim();
  RefreshSim(sim);
}
```

```
// Actualiza los datos de la simulaci\'{o}n
void CSimTreeCtrl::RefreshSim(CSimulation* sim, DWORD dwFlags)
{
```

```
[...]
//Para todos los elementos del arbol
        if (theApp.prefs.simtable.m_act_WinWidth)
      {
        theApp.prefs.simtable.marcados++;
        InsertItem(LVIF_TEXT | LVIF_PARAM | TVIF_STATE | TVIF_IMAGE |
                   TVIF_SELECTEDIMAGE, buffer,0,0,INDEXTOSTATEIMAGEMASK (2),
                   TVIS_STATEIMAGEMASK,ID_PROP_GENE_,hItem,TVI_LAST);
      }
      else
        InsertItem(LVIF_TEXT | LVIF_PARAM | TVIF_STATE | TVIF_IMAGE |
                   TVIF_SELECTEDIMAGE,buffer,0,0,INDEXTOSTATEIMAGEMASK (3),
                   TVIS_STATEIMAGEMASK,ID_PROP_GENE_,hItem,TVI_LAST);

      buffer.Format(GetResString(IDS_PROP_WINHEIGHT)+": %g {\AA}",sim->m_dWinHeight);
[...]
}
```

B.4.8. Otra Mejoras

Autoguardado

```
//////////////////////////////////
//Definicion de la variable de tiempo
//Preferences.h
//////////////////////////////////
  //Obtener intervalo de autoguardado en las simulaciones
  long GetStdAutoSave(){return m_lStdAutoSave;}
  //Obtener intervalo de refresco de stdout en las simulaciones
  void SetStdAutoSave(long lAutoSave){m_lStdAutoSave = lAutoSave;}
    long m_lStdAutoSave;  //Time to AutoSave
```

```
//Preferences.cpp

void CPreferences::LoadPreferences(void)
{
{...}
  //Cargar valor de Autoguardado
  m_lStdAutoSave = theApp.GetProfileInt(PREFS_REG,"StdAutoSave",10);
}
```

```
void CPreferences::SavePreferences(void)
{
{...}
  //Guardar refresco de stdout en las simulaciones
  theApp.WriteProfileInt(PREFS_REG,"StdAutoSave",m_lStdAutoSave);
}
```

```
//Clase que define el hilo de autoguardado
//MainFrm.cpp y MainFrm.h

//Definida nueva clase CAutoSaveThread

//Accion a realizar cuando se pasa el tiempo de autoguardado
LRESULT CMainFrame::OnSimAutoSave(WPARAM wParam,LPARAM)
{
  CSimulation* sim;
  //XAT - WorkSpaceManager
  if ((sim = m_WSManager.GetSelectedSim()) != NULL)
```

```
    sim->SaveSim(NULL,false);
  theApp.prefs.SavePreferences();
  return 0;
}
```

```
//Cargar el valor en la ventana de opciones de la aplicacion
//PrefsPPg3.cpp

void CPrefsPPg3::LoadSettings()
{...
    m_strStdAutoSave.Format("\%lu",prefs->GetStdAutoSave());
}
```

Menús

```
//////////////////////////////////
//Algunas de las funciones de los nuevos menus
//MainFrm.cpp y MainFrm.h
//////////////////////////////////

void CMainFrame::OnSimulacionActualizar()
{
  //Solo funciona si se selecciona la simulacion de la rejilla
  //y sino elije la que muestra cosas
  CSimulation* sim = m_WSManager.GetSelectedSim();
  if (sim)
    AfxGetMainWnd()->SendMessage(IIS_SIM_BATCH,(WPARAM)sim,0);
  else
  {
    POSITION pos = theApp.lstsims.GetHeadPosition();
    if (pos!=NULL)
    {
      sim = theApp.lstsims.GetNext(pos);
      AfxGetMainWnd()->SendMessage(IIS_SIM_BATCH,(WPARAM)sim,0);
    }
  }
}
```

```
LRESULT CMainFrame::OnSimBatch(WPARAM wParam,LPARAM /*lParam*/)
{
  CSimulation* sim = (CSimulation*)wParam;
  ASSERT (sim);
  if (sim->m_RunProps->Batch())
    m_WSManager.RefreshSim(sim,SIMTREE_REFRESH_STATE);
  return true;
}
```

Rutas

```
//////////////////////////////////
//Ejemplo de consecucion de la ruta del ejecutable
//Simulation.cpp
//////////////////////////////////

void CSimulation::Serialize(CArchive\& ar)
{
char Directorio [300];
```

```
  int i;
  //Con esto conseguimos el directorio del ejecutable en Directorio
  GetModuleFileName(GetModuleHandle(NULL),Directorio,300);
  for (i=strlen(Directorio);Directorio[i]!='';i--)
    Directorio[i]=0;
}
```

B.4.9. Ayuda

```
////////////////////////////////
//iisgui.cpp
////////////////////////////////

//Change the help file name to "iisgui_en.chm"
strHelpFile = m_pszHelpFilePath;
strHelpFile.MakeLower();
strHelpFile.Replace(".chm","_en.chm");
free((void*)m_pszHelpFilePath);
m_pszHelpFilePath=_tcsdup(strHelpFile);
```

B.4.10. Guardado de Ventanas

```
////////////////////////////////
//MainFrm.cpp
////////////////////////////////
//Guardado de las ventanas en el fichero
void CMainFrame::OnClose()
{
    [...]
  FILE *Ventanas;
  Ventanas = fopen("ventanas.gui","w"); //Abierto el archivo y vacio
  if(!Ventanas)
    MessageBox(GetResString(IDS_ERROR_GUARDAR), GetResString(IDS_ERROR_ARCHIVO),
            MB_ICONERROR | MB_OK);

  CDocument* pDoc = NULL;
  CSimulation* sim;
  POSITION pos = theApp.lstsims.GetHeadPosition();
  while (pos!=NULL)
  {
    sim = theApp.lstsims.GetNext(pos);
    while (NOTNULL(pDoc = FindDocument(sim,pDoc)))
      fprintf(Ventanas,"%s\t;;;%s\n",pDoc->GetPathName(), pDoc->GetTitle());
  }
  fclose(Ventanas);
    [...]
}
```

```
////////////////////////////////
//iigui.cpp
////////////////////////////////
    [...]
    //Aqui hacemos la llamada a la funci\'{o}n de la carga de ventanas
    AfxGetMainWnd()->PostMessage(IIS_BUSCA_ITEM);
    [...]
```

```
////////////////////////////////
//SimTreeCtrl.cpp
////////////////////////////////
void CSimTreeCtrl::BuscaItem(CSimulation* sim, LPCSTR nombreItem)
{
  ASSERT(sim);
  CString buffer;

  TVITEM* item = FindSim(sim); //BORRAR ITEM cuando no se necesite m\'{a}s
  if (item == NULL)
  {
    return;
  }

  HTREEITEM hItem, hTemp;
  TVITEM pItem;

  SetItemText(item->hItem, sim->m_strSimName);

  if (NOTNULL(hItem = FindItem(item->hItem,ID_PROP_GRAPHS3D)))
  {
    hTemp = GetChildItem(hItem);
    while (NOTNULL (hTemp))
    {
      if (StrStr(GetItemText(hTemp),nombreItem))
      {

        pItem.mask = TVIF_PARAM | TVIF_HANDLE;
        pItem.hItem = hTemp;
        GetItem(&pItem);

        strcpy(Nom_Arch,"hola");
        AfxGetMainWnd()->PostMessage(IIS_SIM_SHOWGRAPH3D,(WPARAM)sim,
                    (LPARAM)pItem.lParam);
        return;
      }
      hTemp = GetNextItem(hTemp,TVGN_NEXT);
    }
  }

  if (NOTNULL(hItem = FindItem(item->hItem,ID_PROP_GRAPHS)))
  {
    hTemp = GetChildItem(hItem);
    while (NOTNULL (hTemp))
    {
      if (StrStr(GetItemText(hTemp),nombreItem))
      {
        pItem.mask = TVIF_PARAM | TVIF_HANDLE;
        pItem.hItem = hTemp;
        GetItem(&pItem);

        AfxGetMainWnd()->PostMessage(IIS_SIM_SHOWGRAPH,(WPARAM)sim,(
                    LPARAM)pItem.lParam);
        return;
      }
      hTemp = GetNextItem(hTemp,TVGN_NEXT);
    }
  }
}
```

Apéndice C

Manual de Usuario

Bibliografía

[1] *Code Guru.*
`http://www.codeguru.com`.

[2] *Code Project.*
`http://www.codeproject.com`.

[3] J. Arias. *Simulación de la implantación iónica en cristales incluyendo baja energía y acumulación de dañado.* PhD thesis, Universidad de Valladolid, 1995.

[4] Nicolás Serrano Barcena. OpenGL Reference Card. *2nd Edition.*

[5] Jason Beres. *Teach Yourself Visual Studio .NET 2003 in 21 Days.*

[6] Álvaro Fernández Díez. *Proyecto Fin de Carrera. Ion Implan Simulator.* June 2004.

[7] Bruce Eckel. *Thinking in C++*, volume 1. January 2000.

[8] Bruce Eckel. *Thinking in C++*, volume 2. January 2000.

[9] R. D. Goldberg, J. S. Williams, and R. G. Elliman. Amorphization of silicon by elevated temperature ion irradiation. *Nuclear Instruments and Methods in Physics Research B*, 106:242–247, 1995.

[10] J. M. Hernández-Mangas. IIS simulation tokens.
`http://bellota.ele.uva.es/~jesman/Papers/cap_tokens.pdf`.

[11] J. M. Hernández-Mangas, J. Lázaro, L. Enríquez, L. Bailón, J. Barbolla, and M. Jaraíz. Statistical 3d damage accumulation model for ion implant simulators. *Nuclear Instruments and Methods in Physics Research B*, 202:138–142, 2003.

[12] J. M. Hernández-Mangas, J. Lázaro, M. Jaraíz, J. Barbolla, and L. Bailón. Dose division algorithm: improvements of damage accumulation on ion implant simulators. *Conferencia de Dispositivos Electrónicos*, 2003.

[13] J.M. Hernández-Mangas, J. Arias, L. Bailón, M. Jaraíz, and J. Barbolla. Improving the prediction capabilities of binary collision approximation ion implant. *Journal of Applied Physics*, 91(2):658–667, 2002.

[14] G. Hobler, A. Simionescu, L. Palmetshofer, C. Tian, and G. Stingeder. Boron channeling implantations in silicon: modeling of electronic stopping and damage accumulation. *Journal of Applied Physics*, 77(8):3697–3703, April 1995.

[15] BDM Federal inc. *Ion beam processing technologies - Sector study.* `http://www.dtic.mil/natibo/docs/ibp-2.pdf`, 1996.

[16] A. F. Komarov, F. F. Komarov, P. Żukowski, C. Karwat, and A. A. Kamarou. Simulation of two-beam ion implantation in the multilayer and multicomponent targets. *Vacuum*, 63:495–499, 2001.

[17] J. Lázaro, J. M. Hernández-Mangas, L. Bailón, J. Arias, and M. Jaraíz. Cumulative damage model in atomistic ion implant simulation. *Conferencia de Dispositivos Electrónicos*, 2001.

[18] Jeff Molefee. *Open GL. Windows Tutorial.*

[19] P. Sigmund. On the number of atoms displaced by implanted ions or energetic recoil atoms. *Applied Physics Letters*, 14:114–117, 1969.

[20] Nikolai Teofilov. 2D Graph ActiveX Control. August 2003.

[21] Nikolai Teofilov. NTGraph 3D ActiveX Control. Jule 2003.

[22] S. Tian, S. -H. Yang, S. Morris, K. Parab, A. F. Tasch, D. Kamenitsa, R. Reece, B. Freer, R. B. Simonton, and C. Magee. The effect of dose rate on ion implanted impurity profiles in silicon. *Nuclear Instruments and Methods in Physics Research B*, 112:144–147, 1996.