Generación de gráficos con TeeChart

Autor:

José Luis Freire

jlfreire@elrinconcito.com

Generación de gráficos con TeeChart

Desarrollados para Delphi y Borland C++ Builder, su integración con el entorno y relativa facilidad de manejo, convierten a estos componentes en una importante herramienta.

Cuando se instala un paquete de Delphi, herramienta a la que vamos a referirnos en esta serie de artículos, se incluyen en el entorno de desarrollo los componentes **Chart**, en la pestaña **Additional**, **DBChart** en **Data Controls** y **QRChart**, en la correspondiente a **QReport**. Ambos son los estándares de éste software elaborado por TeeMach, de Barcelona (a pesar de ello la documentación está en inglés, aunque es de agradecer que la Web se esté traduciendo) del ya muy experto en programación gráfica David Berneda.
Al adquirir la versión profesional del producto se generan otros, bajo la pestaña **TeeChart,** que a título meramente enunciativo son **ChartEditor**, **ChartPreviewer**, **Draw3D**, **ChartScrollBar**, **TeeCommander**, **ChartListBox**, **SeriesDataSet** y **TeeOpenGL**.
Hay más diferencias importantes entre una y otra versión, como son Superficies, Curvas Bezier, Polares, Barras de Error, etc. o las opciones de ejemplo como la Brújula, el Reloj o Barras con Imágenes, entre otras. No obstante, y a pesar de basarme en la 4 Profesional, serán omitidas las características particulares de la misma, a fin de que seamos capaces de seguir y comprender los ejemplos que iremos exponiendo a lo largo de estos artículos.

Desde el menú contextual

Desde la pestaña **Additional** y pinchando con el botón principal en el componente **Chart** lo trasladamos a nuestro formulario como cualquier otro objeto. Habremos añadido un componente del tipo TChart, denominado por defecto Chart1 en el Inspector de Objetos, y podemos trabajar con él tal y como estamos acostumbrados, pero en el momento en el que seleccionemos cualquier opción con un doble clic, lo que va a hacer es mostrarnos la pantalla de diseño que podemos obtener pulsando el botón secundario del ratón en cualquier punto del componente, con lo que abrimos el menú.
Aparte de la versión, el © y el **Acerca de...**, hay tres opciones que nos importan: **Edit Chart**, **Print Preview** y **Export Chart**.
La primera de ellas permite el diseño de lo que en ejecución serán nuestros gráficos. La segunda generará una previsualización de lo que hemos elaborado, y nos permite imprimir seleccionando el dispositivo, la posición del papel, los márgenes, etc. tal como muestra la Figura 1. La tercera, como se puede suponer, está referido a la exportación del gráfico, para lo que ofrece tres posibles tipos de ficheros: BMP, WMF y EMF, añadiendo un cuarto propio, el TEE, según la versión del componente que estemos utilizando.

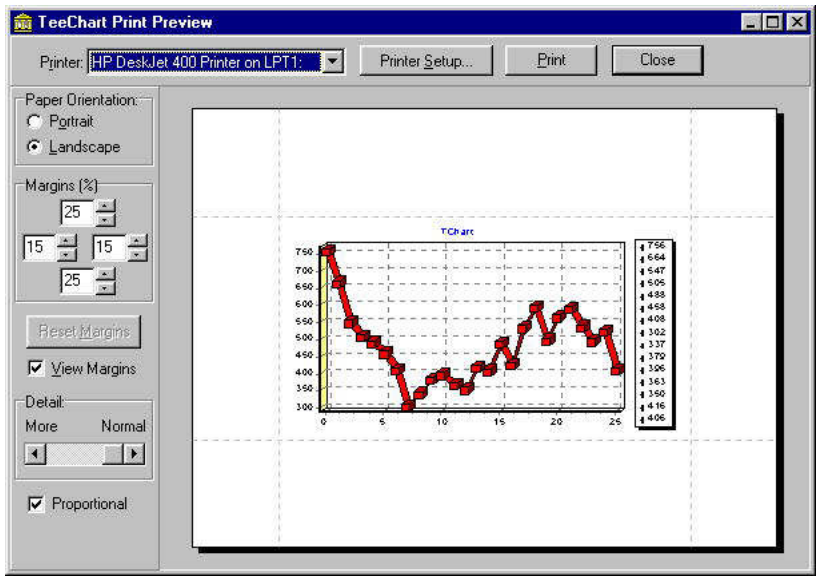

Figura 1

Es evidente que cualquiera de las opciones mencionadas puede realizarse mediante código y en ejecución. No obstante, resulta muy útil tener estas posibilidades en diseño, y vamos a referirnos a ésta forma de trabajo por ahora.
El punto fuerte es la

o elaboración de los gráficos.

Trabajemos con Edit Chart creando Series

Como hemos dicho, si abrimos la paleta **Additional** nos encontramos con el componentes **Chart**, vamos a ubicarlo en nuestra forma. Nos aparecerá simplemente un rectángulo preparado para 3D y con la leyenda "TChart" en la parte superior.
Pinchando sobre él, con el botón secundario del ratón podemos acceder a **Edit Chart**, que va a ser nuestra herramienta de trabajo. Cualquier opción que elijamos va a verse reflejada en el panel del gráfico, con unos valores en principio aleatorios.
Pues bien, la base de todo lo que realicemos va a partir de las **Series**, sobre ellas crearemos los atributos y formas de visualización, serán quienes tomen los valores para componer la gráfica. Así pues, vamos a añadir con el botón **Add** la primera de ellas. Una vez que hemos pulsado nos aparecerá la Galería con un aspecto similar al que se puede apreciar en la Figura 2, dependiendo de la versión que se esté utilizando.

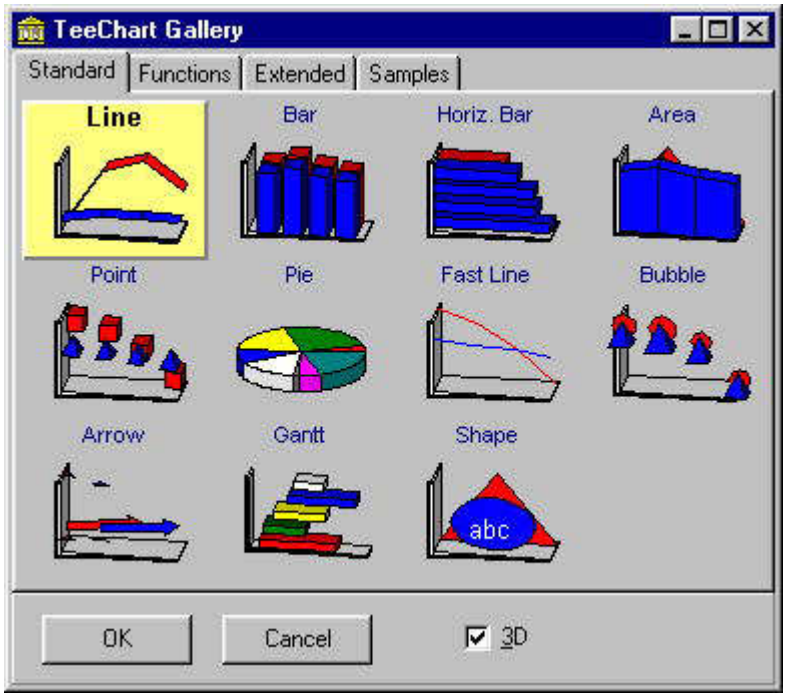

Figura 2

Por defecto, en la parte inferior derecha está activa la marca para crear en 3D. Si no es esa nuestra intención, solo tenemos que utilizar el ratón para desactivarla, con lo que todas las series perderán el fondo.
Una vez elegida la serie que nos resulte conveniente, podemos apreciar que en el panel de edición aparece ésta dibujada en miniatura en la parte izquierda, a continuación hay una marca que se utiliza para activar o no la serie. Si no vemos el formulario, podemos desplazar la ventana de edición. Nos daremos cuenta que en éste se ha creado el gráfico correspondiente a la serie seleccionada, con unos valores arbitrarios, y que desactivando esta marca la figura desaparece. Además, en el módulo de código correspondiente tenemos dos valores en la sección type, el TChart y el correspondiente a lo que hemos elegido, por ejemplo TLineSeries, TBarSeries, etc.
Las restantes opciones en la pantalla de edición son el color, que veremos como modificarlo, y el título.
En los botones de la derecha veremos que el editor nos deja añadir cuantas series queramos y del tipo que sean. Es lógico, puesto que en un mismo gráfico podemos combinar distintos valores o representar los mismos de distintas formas superpuestas e incluso tenerlos creados, aunque inactivos, hasta el momento de su uso.
El resto de las opciones son simples, borrar una serie, cambiar su título, duplicarla o sustituirla por otra de la Galería.

A cada uno de las gráficas a representar se le da el nombre de Serie, de manera que la creación de éstas es la base de inicio para el manejo de este software.

Bien, hemos creado el esqueleto de nuestro gráfico, con él podemos empezar a trabajar, veremos la forma de hacerlo después de dar un repaso a las restantes opciones del editor hasta conseguir la pantalla exacta que buscamos.

¿Editamos el gráfico o las series?

Bueno, es de suponer que ambas. Pero vamos a diferenciar.

Lo que estamos denominando "gráfico" sería el **Chart** completo, su apariencia general, cómo van a funcionar las coordenadas, qué valores pueden tomar y en qué tipo escalas, cómo será el panel, distribución de páginas, zoom, etc., y cómo no, se consigue a través de las opciones de la pestaña **Chart**.
Pero dentro de ese gráfico podemos optar desde lo más simple, que es una serie única, a representar la misma con distintas formas, como pueden ser líneas y barras superpuestas, o escalas y valores diferentes en el mismo panel, porque así nos interese para estudios comparativos, por ejemplo. Para ello debemos optar por elegir desde el color de cada una, hasta la fuente de datos, pasando por el resaltado de los puntos de inflexión y la representación de sus valores con las máscaras correspondientes, y ello, como el resto de las opciones, a través de la pestaña **Series**.
Ya tenemos la edición dividida en sus dos bases, la totalidad y la parte.

Vamos a manejar la opción Chart

La primera opción, **Series**, la hemos visto al probar la creación de una de ellas y no tiene más misterio. Solamente resaltar que determinadas opciones, y como es lógico, afectarán o no al resultado dependiendo del tipo de serie que hayamos elegido. No se puede pretender que el resultado de las opciones sea el mismo cuando manejamos un gráfico de barras o uno de tarta (*pie*), por poner un ejemplo, por lo que vamos a intentar ver los aspectos generales independientes de la serie elegida, salvo que existan particularidades de interés especial.
En modo de diseño vemos un gráfico que se corresponde a valores "aleatorios" que el componente introduce para que podamos trabajar. Si ejecutamos nuestro miniprograma tal y como se encuentra en estos momentos, no hallaremos más que un panel vacío. Por lo que considero de interés hacer un paréntesis de manera que podamos introducir valores y ver su funcionamiento en la práctica.
Hagamos algo simple, insertemos un botón y en el evento OnClick introduzcamos cualquier valor que se nos ocurra, para poder ir viendo resultados. El código sería este:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
   With Series1 do
   begin
     AddXY(10,20,'',clRed);
     AddXY(20,15,'',clRed);
     AddXY(18,30,'',clRed);
     AddXY(25,40,'',clRed);
     AddXY(26,52,'',clRed);
   end;
end;
```

Fácil ¿no? Le hemos dado las coordenadas de los primeros cinco puntos que se nos han ocurrido a una serie del tipo líneas, previamente creada, y simplemente los ha unido como era su obligación. Si ahora ejecutamos y pulsamos el botón se mostrarán estos puntos dentro del panel.
La segunda opción con la que nos encontramos es **General** y realmente es simple. Tan solo destacar cuatro aspectos de importancia, de los cuales dos ya se han mencionado como opciones independientes del menú contextual, y que, como todo, podremos hacerlos funcionar igualmente desde código: la posibilidad de previsualizar, (**Print Preview**), la exportación a otros ficheros, la activación del zoom y la de movimiento. Si tenemos activada la opción **Allow Zoom**, en ejecución y con el botón principal podemos trazar una cuadrícula que será la que se visualice en el panel. Si pulsamos cualquiera de los botones de **Allow Scroll**, salvo el que especifica que no haya movimiento, le estaremos indicando cómo queremos que se desplace la imagen utilizando posteriormente el botón derecho.
El desplazamiento de la imagen (*Scroll*) en el panel puede servir, llevado a código, para efectos muy interesantes en determinados tipos de gráficos. Suponiendo, por ejemplo, una serie de líneas y creando un movimiento horizontal automático, podríamos conseguir el efecto de cualquier pantalla del tipo sismograma, encefalograma, etc.

La opción Axis, es decir, la utilización de los ejes, es a mi parecer una de las más importantes y complicadas. De ella va a depender el funcionamiento general del gráfico, pues aquí será donde le vamos a decir cómo se han de comportar los elementos que creemos. Una Serie con idénticos valores varía por completo si manipulamos su representación en relación a las coordenadas.

La opción **Axis**, es decir, la utilización de los ejes, es a mi parecer una de las más

importantes y complicadas. De ella va a depender el funcionamiento general del gráfico, pues aquí será donde le vamos a decir cómo se han de comportar los elementos que creemos. Una serie con idénticos valores varía por completo si manipulamos su representación en relación a las coordenadas. El ejemplo que hemos realizado, no tendría sentido si ahora le definimos unos valores máximos y mínimos a los ejes que estén por encima o por debajo de los puntos elegidos, por poner el ejemplo más evidente, ya que nos encontraríamos con un panel en blanco.

De la cantidad de posibilidades que nos ofrece, en la mayoría de los casos su utilidad es muy evidente o intuitiva como para detallarlas, por lo que vamos a dar un repaso sin entrar a mencionar más que lo que pueda resultar confuso.

Con independencia de la opción de mostrar o no los ejes (*Show Axis*), que no hay que confundir con visualizar o no el panel, el resto está supeditado a lo que le indiquemos dependiendo de lo que tengamos pulsado, ya sea izquierda, derecha, arriba, abajo y fondo. Aunque en esencia el funcionamiento de todos ellos es el mismo, se pueden realizar combinaciones interesantes. En principio tomemos cualquiera de ellos como ejemplo, supongamos activa la izquierda del gráfico (*left*) cuyos resultados, si no superponemos series, no tendrá sobre la figura final ninguna diferencia a la elección de la zona derecha (*Right*) pues estamos trabajando el mismo eje, tan solo diferirá por el lugar de visualización de los valores correspondientes.

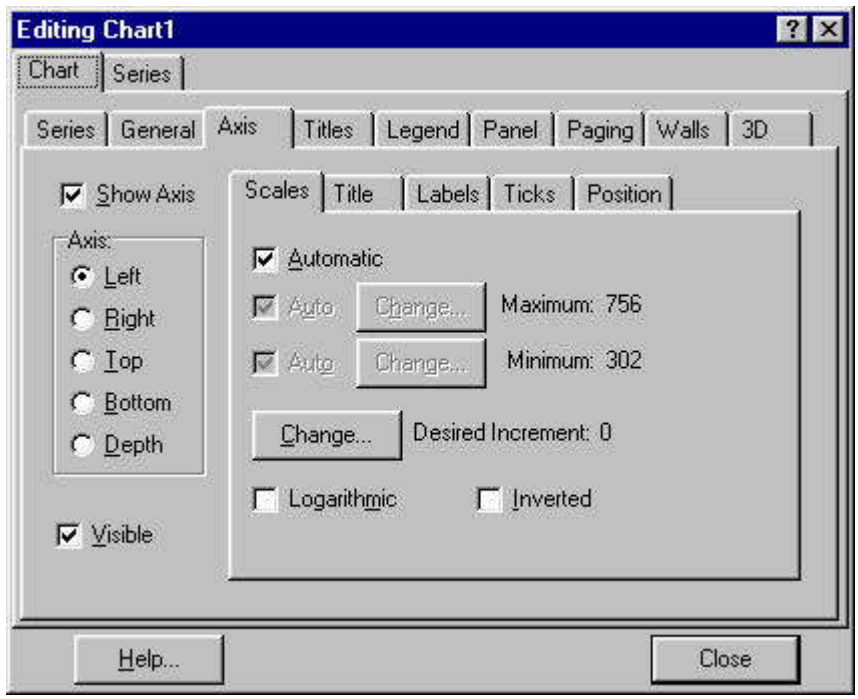

Figura 3

En este momento vamos a continuar con el panel derecho de la misma pantalla, como se ve en la figura 3.

Las escalas (*Scales*) van a fijar la visualización de los valores máximo y mínimo, así como los intervalos entre ellos. Influirá el tipo de datos con el que trabajemos, no podemos considerar de la misma forma valores absolutos que porcentuales o de fecha, pero no vamos a considerar esto por ahora. Por defecto todo es automático, y el incremento (*Desired Increment*) cero indica lo mismo.

En el ejemplo en que hemos introducido la función Addxy fijábamos unos valores de X que oscilaban entre 10 y 26, así como sus correspondientes de Y entre 15 y 52. El resultado pues es obvio, el origen de coordenadas será el punto 10,15 y los valores que se reflejen así como el dibujo de la línea terminarán en el 26,52, ¿con qué incremento?, le hemos especificado cero, así que considerará las diferencias entre los valores máximos y mínimos y hará un ajuste automático.

Si este es el resultado que buscamos, perfecto. Pero somos muy libres de considerar que no tenemos un eje de coordenadas cuyo punto de origen sea el tradicional 0,0 y necesitamos ese punto, es habitual, siempre que se utilicen gráficas comparativas, que cumplan unas características iguales, y no se parecerá a nivel visual en nada una curva si el origen es distinto, los resultados no son comparativos y normalmente el uso de los gráficos, especialmente en gestión, es este.

Bueno, pues busquemos ese punto deshaciendo el automatismo. En ese momento se activa la posibilidad de utilizar los valores que indiquemos como máximos y mínimos, o dejar automático uno, o ambos. Queremos que nuestro origen sea el punto cero, y recordemos que tenemos activada la opción

"izquierda", por lo tanto estamos dando valores al eje de las X. Fijamos el mínimo en cero ¿y el máximo?, pues dependiendo de lo que interese y lo que estemos realizando podemos fijarlo nosotros o que lo haga el propio TeeChart pulsándole como automático. Si desconociendo cual es el valor mas alto que se puede dar en un gráfico, lo especificamos, la figura se va a atener a esos valores, o bien quedará ridículamente pequeña o bien se "saldrá" del panel. No obstante, y para el caso en que se precisen gráficos comparativos, como antes decíamos, puede ser necesario, dejémoslo así por ahora, con un mínimo de cero y un máximo indefinido, que coincidirá con el valor mas alto posible que le enviemos.

Hemos fijado uno de los ejes, nos faltaría el otro, por lo tanto hacemos la misma operación pero ahora tomando, por ejemplo, la opción "abajo" (*bottom*) , y conseguiríamos el mas estándar de los gráficos.

Tan solo nos queda considerar en este aspecto el incremento a utilizar, y esto puede ser muy importante a la hora de la precisión. Con el ejemplo que hemos puesto anteriormente no es significativo, pero si debemos representar un número alto de valores, por ejemplo, el resultado de una ecuación que va tomando valores a partir de un bucle que se incrementa de uno en uno, es ilógico marcar un incremento alto en el gráfico, más coherente será acoplar ambos.

Las dos opciones que nos faltan de este panel es, que los valores sean invertidos, es decir, que el punto cero no sea el origen de coordenadas sino el extremo del eje, y la posibilidad de escalar de forma logarítmica, es decir, que en lugar de trabajar con la normal o aritmética, en que la distancia entre intervalos iguales es la misma, se puede utilizar de manera que esas longitudes sean proporcionales a su logaritmo, lo que es una forma de trabajo típico de cálculos de mayor precisión como las curvas utilizadas en estudios geológicos y similares.

El resto de las opciones son simples. La pestaña Título (*Title*) permite "etiquetar" cada eje, el denominar a qué corresponden los valores que se indican, por ejemplo, millones de pesetas en relación con los meses. En este caso, pulsando la opción "izquierda" escribiremos el "literal millones de pesetas", y abajo (*bottom*) pues, como es lógico, "meses". Lo que no nos sería posible es escribir un título de eje si no existen valores en el mismo.

Se puede apreciar, que según se pulse una opción de posicionamiento, el ángulo va modificándose, izquierda a 90º, derecha a 270º y 0º en las superior e inferior, que es la inclinación correcta del texto, si no se selecciona otra.

Las opciones siguientes de *Labels*, *Ticks* y *Position* dan lugar a modificaciones en cuanto a grosores de los ejes, visualización o no de valores, formatos, etc. que son fácilmente deducibles, por lo que no vale la pena detallarlos aquí. Tan solo comentar al respecto, que es muy simple ver el resultado de cada uno de ellos, pues modifican el gráfico de prueba según se pulsan.

Podemos encontrarnos ante situaciones en las cuales no parece que exista ninguna reacción gráfica tras activar una opción. Una de las cosas más normales en estos casos es que en realidad no se produzca, ya que todas posibilidades están activas con independencia del tipo de gráfica que se está utilizando, cuando es evidente, que las posibilidades que con las que contamos en un gráfico de Gantt pueden diferir en mucho con las que corresponden a uno de líneas o de tarta.

Terminemos con las subopciones de Chart.

Estas son también bastante simples e intuitivas.

En el título (*Titles*) se especifica la denominación del gráfico, que aparecerá en la parte superior del mismo, con las características típicas de fuente, color, fondo, alineación, etc. Tan solo hacer constar un detalle, el resultado de no titular el marco no es el mismo que el de dejar este invisible, porque en el primer caso se reserva el espacio y en el segundo no.

Exactamente funciona en la misma opción la posibilidad de poner un pié (*foot*) con iguales características.

El cuadro de valores o leyendas (*legend*) es el que aparece por defecto en la zona derecha del gráfico. No vamos a entrar en las características de fuentes, color , etc. que son suficientemente visuales. Tan solo vamos a referirnos a dos opciones, las que denominan estilo, tanto de leyenda como de texto.

Por supuesto, lo primero que podemos hacer es eliminarlo, desactivando la opción visible, pero si queda activa, en el caso de *Style Legend*, podemos elegir que en lugar de los valores por defecto que toma y que aparecen como "Automáticos" figuren las Series que estamos utilizando. Esto tendrá sentido cuando superpongamos distintas, lo que haremos en cuanto hayamos repasado las características básicas, de la misma forma que ocurre si seleccionamos que lo que aparezca sean los valores máximos de cada una de las Series.

En *Text Style*, hay que hacer dos observaciones. Una que se puede elegir entre los de la izquierda y derecha (si hubiésemos creído oportuno diferenciarlos) de manera que indiquen valores absolutos o

porcentuales, lo no quiere decir que se modifique el valor, pues es tan solo a nivel visual, y la segunda la de considerar, exclusivamente los del eje X.

En cuanto a la opción *Panel* es interesante a nivel de aspecto pero, como las anteriores, su manejo es muy intuitivo. Permite realzar o hundir, cambiar el color, tanto del propio panel como de sus bordes en colores planos o degradados, y una opción que puede resultar interesante o curiosa, la de incluir una imagen de fondo (*Back Image*) y de diferentes maneras, por ejemplo el logotipo de la empresa, bien en la totalidad del panel o ceñida a los ejes de coordenadas, en ambos casos, bien que ocupe toda el área, que se centre en ella o que se distribuya en forma de tapiz.

Con paginación (*Paging*) es muy simple conseguir que gráficos de tipo líneas o similares, que puedan tener un número muy alto de puntos o una longitud grande se puedan distribuir en distintas páginas, especificando la cantidad de ellos que queremos que aparezcan en cada página.

Las dos últimas opciones, *Walls* y *3D*, tienen como objetivo configurar las formas y los bordes en que se dibujan los ejes, tanto en profundidad, colores, etc.

Todo lo mencionado anteriormente afecta al gráfico en sí y su relación con las Series, en el caso de escalas de ejes de coordenadas, por ejemplo, pero a excepción de la que hemos utilizado como ejemplo, no nos hemos referido a ellas, ni a su presentación, ni a las relaciones existente cuando se utilizan varias, y a ello dedicaremos en adelante.

Las Series en el editor de gráficos.

De acuerdo completamente en que la descripción de lo que se puede hacer con un editor es soporífero para los programadores, no obstante resulta importante por dos motivos, uno porque es un sistema para recorrer las opciones que el paquete de trabajo ofrece, y otro por que, posteriormente, resulta muy fácil trabajar con gráficos que ya tienen todas sus características predefinidas, de manera que, una vez asignadas con el editor las que pretendemos obtener, en el momento de programación o de ejecución, según corresponda, tan solo haya que concretar los valores que tiene que reflejar, lo que es una opción válida y bien simple.

Vamos a dedicarnos a trabajar cada una de las Series de que se compone el gráfico. Recordamos para ello que no hay por qué atenerse a una sola, de hecho se pueden superponer, ya sean del mismo tipo, todas Líneas, Barras, etc. o mezclas de ellas.

El aspecto de las Series.

Para hacernos una idea hemos creado tres tipos distintos en el mismo gráfico, como se puede ver en la "figura 1", con el resultado (en este caso caótico) que se refleja en la "figura 2". El editor nos permite elegir con cual de las Series creadas vamos a trabajar, y dependiendo del tipo de que se trate nos ofrecerá las opciones correspondientes.

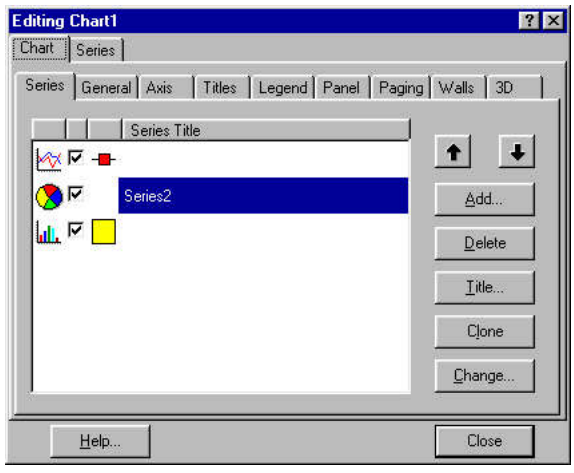

Figura 1

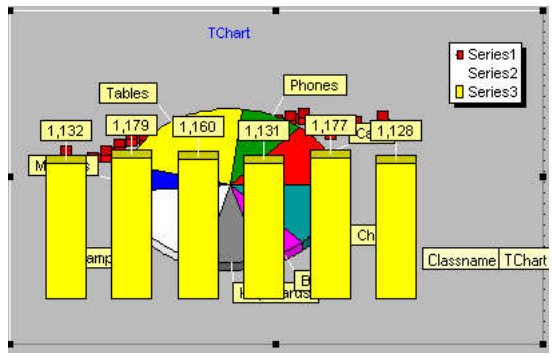

Figura 2

Vamos a utilizar para el ejemplo tres tipos básicos en programas de gestión, la de Líneas , Pie (tarta) y Barras (en este caso verticales), y para no equivocarnos cambio su nombre por defecto Series1, 2 y 3 por

Slineas, Starta y Sbarras y vamos a ver las diferencias que nos ofrece cada una.

Hagamos un inciso en este punto. Hay diferencias entre las versiones estándar que Delphi trae por defecto y las versiones profesionales. La primera que salta a la vista es que en esta última hay mas variedad. Si las **estándar** son similares, en cuanto a las funciones a mucha diferencia, pero además están las **extendidas** y las de **ejemplo**.

En el caso de las funciones, aunque varíen en cantidad, no nos afecta mucho, en definitiva la suma de dos líneas se representa por otra línea, pero en las otras dos sí las hay, no se parece en nada una Serie referida a un **gráfico de Gantt** que a una **Polar** o una curva **Bezier**. No nos vamos a referir a éstas últimas ni a las que aparecen como ejemplo, tan solo a las más típicas de las que trae el paquete de Borland.

Bien, pues aunque varían el contenido de las opciones en los tres tipos mencionados, en definitiva hay cinco posibilidades que se reflejan en las pestañas del gráficos. El formato (Format) de la serie. Lo referente a los puntos (Point) para aquellos tipos que lo permitan, pues su utilidad es incidir en los cambios de valor que adoptan especialmente las líneas. Las de carácter general, que están relacionadas tanto con el contorno como con las cajas descriptivas de los valores, es decir, fuera de los ejes de coordenadas. Los letreros (Marks) que reflejan el valor de cada punto, porción o barra, ateniéndonos a los tres ejemplos que utilizamos. Y una opción final que llaman fuente de datos (Data Source) cuya utilidad me resulta confusa, porque en la práctica será a través de nuestro programa como le indiquemos los valores que ha de tomar en cada momento.

Como cada modificación que hagamos en el editor va a verse reflejada en el gráfico, sugiero desactivar todas las Series con las que no se esté trabajando para evitar confusiones, de forma que si vamos a centrarnos de momento en Slineas, desde la opción Chart – Series desactivaríamos las otras dos.

Con gráficos de Líneas.

El formato, **Format**, en las líneas es sencillo, tienen un borde y un contenido, lo que se ve con facilidad dando al botón “Border”, y seleccionando un color cualquiera, por ejemplo azul oscuro. Damos OK y en el botón “Color” que es el contenido o el fondo, elegimos un amarillo, para que destaque. Observamos que los puntos aparecen del mismo color que el fondo. Es cuestión de ir manejando estas opciones hasta que el resultado sea el apetecido. Como observaciones interesantes cabría destacar que mientras se seleccione el ancho del borde a 1, podemos cambiar su trazo, sólido, intermitente, etc. Si se aumenta el ancho ya no, pero también superponemos el borde al contenido de la línea. “Dark 3D” oscurece el fondo, lo que se hace notar mas o menos dependiendo de los que usemos, es mas vistoso el efecto si dejamos pulsado “Color Each” puesto que se modifican los colores entre cada punto.

El convertir en modo de la línea en “Stairs” o escalera es claro, desaparecen las inclinadas y se sustituyen por su figura correspondiente en verticales y horizontales, en un sentido u otro dependiendo que sean invertidas o no.

El fondo es modificable mediante la opción “Pattern”, por defecto es sólido pero permite desde eliminar por completo las líneas junto con su borde a dejar éste y cambiar el contenido por diagonales, horizontales, etc.

En relación a los puntos, **Point**, la mejor forma es ir variándolos viendo el resultado, es puramente modificar la estética, podemos desde eliminarlos a cambiarlos el color (por defecto el de la línea, pero modificable con “background”) o la forma con “Style”.

La opción **General** tiene sus variaciones importantes. Permite desaparecer el cuadro de valores y cambiar la forma del cursor en ejecución, pero también dar un formato, como si utilizásemos el Format de Delphi para decidir como se van a visualizar en los ejes, en formato de números enteros, reales o porcentuales.

El detalle de modificar la situación de arriba o abajo, izquierda o derecha, puede ser pura estética o no, ya que también tenemos la posibilidad de ponerlo en ambos, lo que resulta interesante junto con la opción **Axis** de **Chart** cuando son dos gráficos los que utilizamos y con distintos valores, o cuando se decide reflejar a la vez la escala normal y la escala logarítmica.

Por último simplemente el poder dar valores de fecha/hora a los ejes, tanto en horizontales como en verticales.

El activar **Marks** tan solo significa que en cada punto ponemos un “letrero” indicando su valor en el gráfico. Puede ser desde simple a bastante complejo e ilegible, como especificar en cada uno una etiqueta conteniendo su porcentaje en relación al total, etc., son las opciones que pueden verse bajo “Style”. El

"Format" de esta opción se refiere a la etiqueta en sí y el "Arrow" a la forma de unión con el punto. En cualquier caso, vuelven a ser efectos estéticos.

La fuente de datos (**Data Source**) no veo su importancia en fase de diseño, sino de ejecución, que será cuando demos los valores a través de las funciones del TeeChart.

Y lo que cambia al utilizar Pie o Barras.

Si cambiamos de Serie y utilizamos STarta, la diferencia fundamental va ha estar en **Format**, como es lógico no tiene las mismas características que las Líneas.

Las diferencias más significativas son, en cuanto al sistema de visualización, las opciones **Custom Pie Radius**. Si se elimina el automatismo, lo que se hace es modificar los valores del radio de la circunferencia, bien en un sentido vertical u horizontal, con lo que si ambos no coinciden, la figura resultante no sería una circunferencia.

Con **Explode Biggest**, se pueden "desgajar" valores, en una distancia equivalente al que se expecifique, como se ve en la "figura 3".

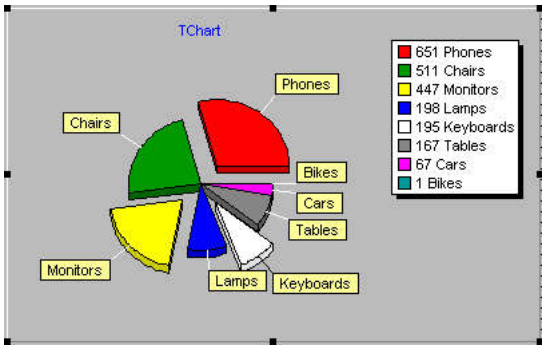

Figura 3

Groups Slices contiene las opciones "None", "Below %" y Below Value". Con las dos últimas se restringen valores, bien en relación con el porcentaje que les corresponda sobre el total, o bien las que no superen una cantidad mínima que se estipule. Con "None" se visualiza la totalidad.

Si utilizamos gráficos de barras, en nuestro caso la serie Sbarras, nos ocurre lo mismo que con la anterior, tan solo se modifica **Format** para adaptarse a las nuevas condiciones.

El **Style** es claro, no tiene por qué tratarse de rectángulos, y de hecho podemos modificar a cilindros u otras formas de barras distintas.

Todas las opciones son de forma de visualizar en un sentido estético, menos la opción **Multiple Bar**. Los gráficos de barras son los típicos para representar distintas series en un mismo Chart. La forma en que se relacionan se ofrecen en estas opciones. Por defecto "None", las barras se superponen, Con "Side" quedan paralelas, "Stacked" las superpone y "Stacked 100%" hace lo mismo pero obliga a hacer visibles las marcas (**Marks**) pues las fija a un mismo tamaño con independencia de su valor.

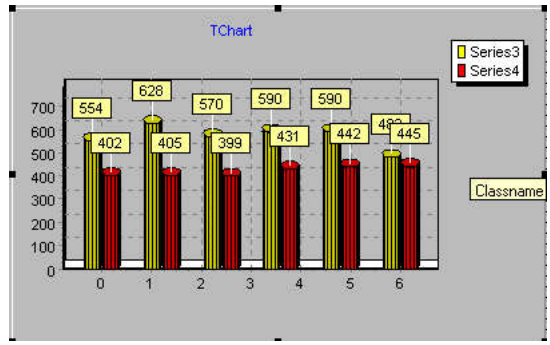

Figura 4

He creado una cuarta Serie para reflejar barras unidas por la propiedad "Side" tal como aparece en la "figura 4".

Con esto hemos hecho una introducción a las posibilidades que nos ofrece TeeChart desde el Editor. A partir de ahora o bien lo utilizaremos combinado con código fuente o no lo veremos en absoluto, como nos sea más cómodo. Espero que haya servido para ofrecer un acercamiento a las muchas posibilidades de la herramienta.

Como punto final veamos un ejemplo de tres Series, con valores aleatorios, para compobar la combinación en un mismo Chart ("Figura 5")

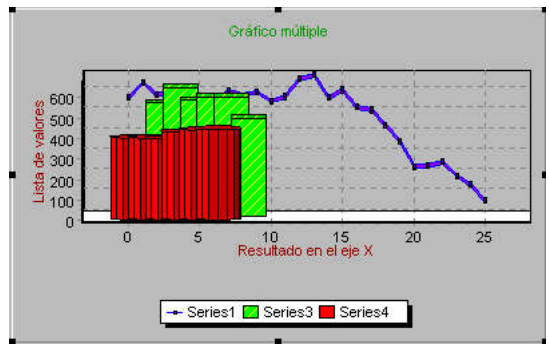

Figura 5

Generación de gráficos con TeeChart

Diseño de Charts y Series desde código.

Comencemos por el principio.

Después de haber visto qué podemos hacer, ahora nos corresponde hacerlo, crear gráficos reales. Probablemente será información complementaria a una serie de datos, de manera que se pueda tener una idea de un simple vistazo.

Bueno, vamos a "pegar" el componente *TChart* desde la paleta Adicional ajustando el tamaño en el formulario. A partir de aquí, es prácticamente igual utilizar el Inspector de Objetos para conseguir las características deseadas o el Edit Chart, pulsando el botón secundario.

En cualquiera de los casos lo que primero nos interesa es qué pretendemos. ¿Vamos a crear un gráfico manipulable por el usuario a nivel de tipos de Series? ¿Vamos a representar, sin más, una lista de valores?

Supongamos, en primer lugar, que estamos interesados en mostrar una gráfica correspondiente a unas variables obtenidas por el procedimiento que se nos ocurra, resultado de cálculos, introducción de datos por el usuario, lectura de determinados campos, cualquier cosa, pero necesitamos valores para poder hacer algo, y, puestos a suponer, estos están contenidos en una matriz.

A continuación pensamos qué tipo de Series nos interesa. Está claro que intentamos representar algo dentro de unos ejes de coordenadas, es decir, dos dimensiones, pero solo tenemos una lista de valores. Es decir, que hemos de utilizarla para complementar los dos ejes. Así que en ese aspecto nos daría igual utilizar Líneas, o Barras, parecen las más representativas para estas características. ¿Elegimos barras verticales? Con la propiedad *Series List* se despliega nuevamente el modo de edición y elegimos añadir barras. Al cerrar aparecen en pantalla unos valores que no nos interesan para nada. No importa, estamos en diseño y en ejecución desaparecen.

Vamos a dejarlo así, que hemos obtenido la base con poco trabajo, y según lo veamos al introducir los valores verdaderos lo vamos modificando.

Mejor poner un botón para manejar el gráfico con el evento *OnClick.*

Tipos de Series posibles en un Chart.

Aunque hay diferencias entre las posibilidades del TeeChart Pro y el estándar que trae el paquete de Delphi, están principalmente en la posibilidad de utilización de las *Series* extendidas (Bezier, Candle, Contour, ErrorBar, Point3D, Polar, Radar, 3D Surface y Volume) y en las funciones (no obstante, estas pueden implementarse con cálculos matemáticos). En cuanto a las básicas no hay diferencias.

Cuando hagamos ahora una introducción a la representación de valores nos daremos cuenta que determinadas series pueden ser tratadas con un número determinado de variables, salvo en un par de casos en los cuales una de ellas se puede omitir y el componente la obtendrá según los valores dados a la primera, y por lo tanto necesitaremos utilizar unas u otras funciones. En general, en el "cuadro 1" se reflejan los parámetros que se han de pasar según se trate de unas u otras, con independencia del *Xlabel* que es común.

Tipo	**Variables**	
Line	2	X, Y, Xlabel
Fast Line	2	X, Y, Xlabel
Bar	2	X, Y, Xlabel
Horiz Bar	2	X, Y, Xlabel
Area	2	X, Y, Xlabel
Point	2	X, Y, Xlabel
Pie	1	Valor, Xlabel
Arrow	4	X0, Y0, Xlabel, X1, Y1
Bubble	3	X, Y, Xlabel, Radio
Gantt	3	Valor1, Valor2, Distancia, labels
Shape	4	X0, Y0, X1, Y1

Cuadro 1

Un ejemplo introductorio de programación con Chart y Series.

Para buscar un ejemplo supongamos que nuestra matriz contiene los beneficios de la empresa durante el ejercicio.

Vamos a hacer la primera prueba.

```
var
  Form1: TForm1;
  aBeneficios:Array[1..12] of   Integer=(320,430,560,400,450,510,580,645,600,530,440,280);

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var x:byte;
begin
  With Series1 do
    for x:=1 to 12 do
      Add(aBeneficios[x],'',clBlue);
end;
```

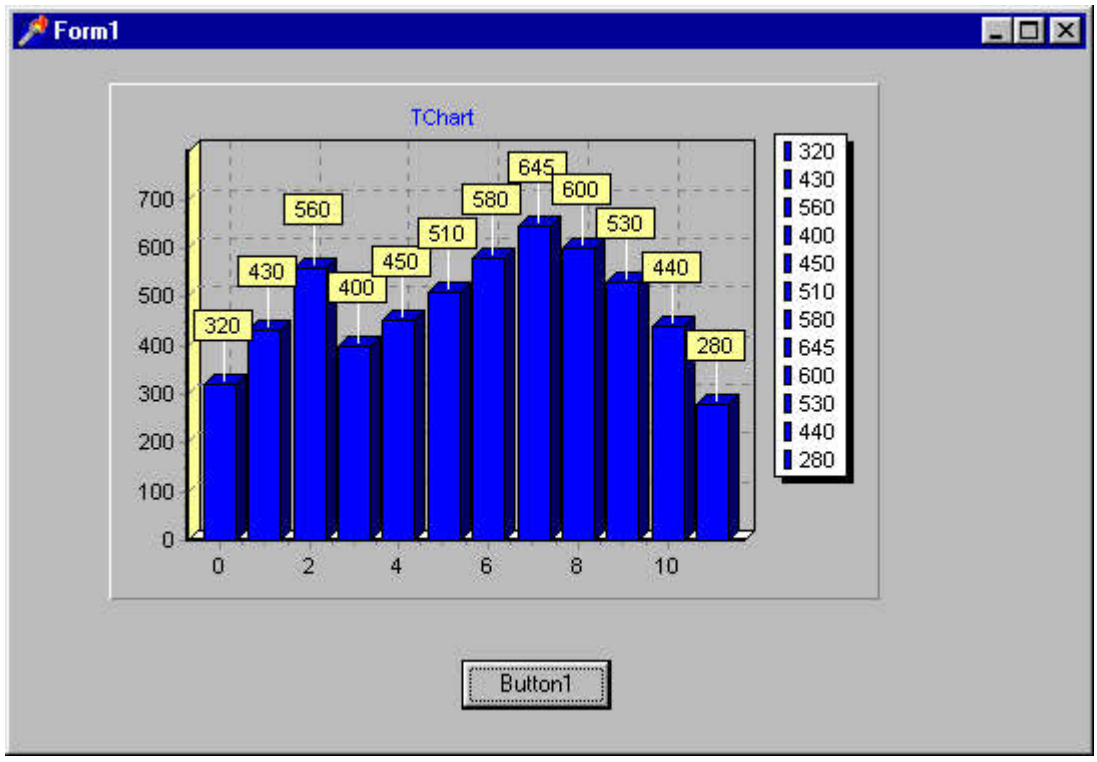

Figura 1

Y como siempre ocurre, no nos gusta. No es legible, un gráfico tiene que servir para ser analizado en un instante, la retina debe de captar los datos de forma inmediata. No nos dice nada el cuadro de valores de situado a la derecha, no sabemos de qué van los ejes ni qué se esta representando.

Todo eso hay que arreglarlo.

Como dijimos en los anteriores números, son dos opciones diferentes las que afectan al *Chart* y las de las *Series*, y como tal las vamos a considerar.

Cambiemos la zona del Chart:

```
With Chart1 do
  begin
// Borramos el texto que hubiera
    Title.Text.Clear;
// Añadimos el nuestro
    Title.text.add('Resultados del 1.999');
  end;
end;
```

Y los carteles tampoco son atractivos, vimos que pertenecen a Series, luego las quitamos. ¡Ah! Y nos acabamos de dar cuenta que los meses empiezan en 0 y terminan en 11 y eso no es posible. No obstante, si damos un valor mínimo de uno, la barra cero no se dibuja, por lo que tendremos que indicar que lo que nos interesa en el eje inferior no son valores, sino texto. Para ello podemos utilizar el que ya tenemos convirtiendo la variable del bucle For o crear una matriz distinta con el nombre de los meses. Habida cuenta que dependiendo del tamaño que se haya dado al gráfico el que se solapen los nombres es muy fácil, optamos por su numeración.

El resultado es la "**figura 2**" y el código que la acompaña.

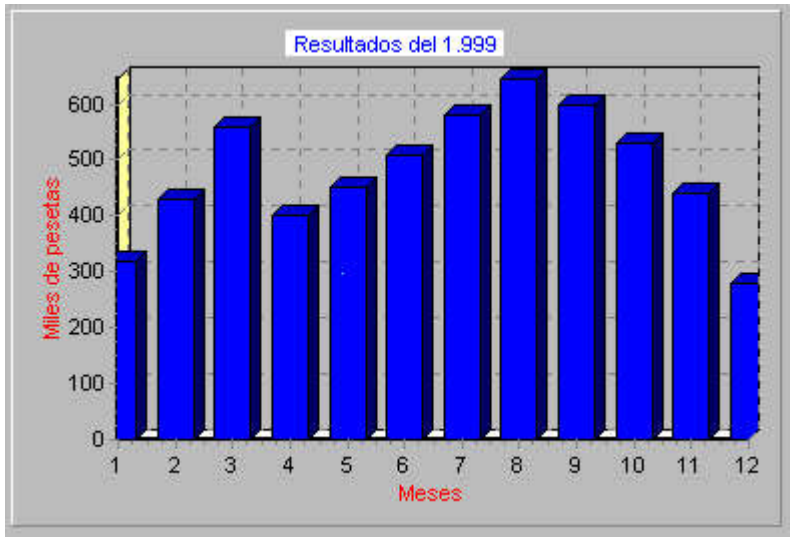

Figura 2

```
procedure TForm1.Button1Click(Sender: TObject);
var x:byte;
begin
  With Chart1 do
  begin
// Borramos el texto que hubiera
    Title.Text.Clear;
// Añadimos el nuestro con color de fondo
    Title.Color:=clWhite;
    Title.text.add('Resultados del 1.999');
// Eliminamos el cuadro de valores
    Legend.Visible:=False;
```

```
// Ponemos título a los ejes con color en font
   LeftAxis.Title.Caption:='Miles de pesetas';
   LeftAxis.Title.Font.Color:=clRed;
   BottomAxis.Title.Caption:='Meses';
   BottomAxis.Title.Font.Color:=clRed;
// Cambiamos los valores automáticos
// respetando que el índice de las columnas empieza en 0
   BottomAxis.Automatic:=False;
   BottomAxis.Maximum:=11;
   BottomAxis.Minimum:=0;
//  Preparamos el eje inferior para que tome los
//  valores de Texto que le pasaremos con la función
//  Add al dibujar las series.
   Chart1.BottomAxis.LabelStyle:=talText;

  end;
  With Series1 do
  begin
// Eliminamos los carteles sobre las barras
   Marks.Visible:=False;
   for x:=1 to 12 do
// Incluimos el valor de X como texto de meses
     Add(aBeneficios[x],inttoStr(x),clBlue);
  end;
end;
```

Lo que se puede dibujar con una variable y cómo hacerlo.

El ejemplo anterior ha sido un intento de acercar el diseño de las *Series* de forma comprensible y muy ambigua. En éste apartado supondremos que el Chart o contenedor lo tenemos ya adaptado a las necesidades y vamos a referirnos a lo que se puede hacer con los gráficos desde código, su creación, gráficos múltiples (tan útiles a niveles comparativos) y sobre todo las funciones propias a utilizar para designar sus valores.

La expresión de adaptar a las necesidades, significa como aparece en la "**figura 3**", llamémosle, tal como su madre le trajo al mundo. Podíamos optar también por hacer un Create de un formulario con el Chart correspondiente, pero la ley del mínimo esfuerzo impera, en tanto no haya nada que nos obligue a lo contrario.

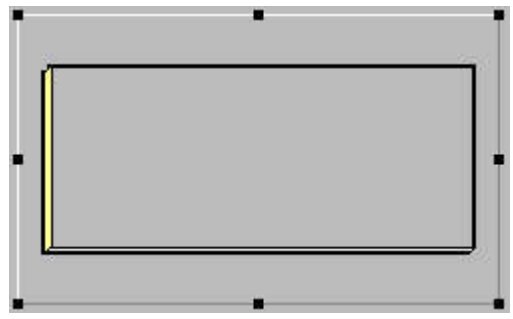

Figura 3

En cuanto a las *Series* o a las distintas propiedades del marco, ya es distinto, habida cuenta de que dentro del mismo contenedor e incluso con los mismos datos, el usuario puede elegir por un sistema de visualización u otro.

Para ver esto, vamos a crear de forma dinámica lo que antes habíamos hecho a través del editor. Usemos, para variar, Líneas en lugar de Barras. ¡Ah! Y que no se nos olvide incluir en uses Series.pas, que en este caso ya no lo hará de forma automática.

La creación de una Serie en ejecución es simple, tan solo se trata de crear una variable del tipo que nos convenga, en este caso:

```
var
  SerieLineas:TLineSeries;
```

Y a continuación, con el constructor y la propiedad *ParentChart* está resuelto

```
SerieLineas:=TLineSeries.Create(Self);
SerieLineas.ParentChart:=Chart1;
```

Si necesitásemos cambiar el modelo de *Serie* nos sería suficiente con borrar la anterior y seleccionar la nueva:

```
SerieLineas.Clear;
```

¿Y si buscamos la coexistencia de distintas en un mismo gráfico?. Pues no sería ningún problema, con crear las dos y dibujarlas tenemos suficiente.

```
var
  SerieLineas:TLineSeries;
  SerieBarras:TBarSeries;
  x:byte;
begin

// Creamos las serie y asociamos con su Chart
  SerieLineas:=TLineSeries.Create(Self);
  SerieLineas.ParentChart:=Chart1;

  SerieBarras:=TBarSeries.Create(Self);
  SerieBarras.ParentChart:=Chart1;

// y simplemente dibujamos ambas
With SerieLineas do
  begin
    Marks.Visible:=False;
    for x:=1 to 12 do
      Add(aBeneficios[x],inttoStr(x),clBlue);
  end;
  With SerieBarras do
  begin
    Marks.Visible:=False;
    for x:=1 to 12 do
      Add(aBeneficios[x],inttoStr(x),clGreen);
  end;
```

Muy bien, pero esto no nos basta. Queremos hallar la media de los meses. Pero solo tenemos una serie, mientras que en la versión profesional tenemos tres funciones *Average* distintas, en la estándar tan solo una. Se plantea entonces dar valores al *Periodo*. Con un valor 0, no tenemos linea, no existe, con valor = 1 los puntos que toma de referencia son los mismos que hay, con valor = 2 la mitad, etc. De manera que si tenemos en nuestro ejemplo de beneficios 12 puntos y queremos que la media sea una línea recta, para ver el resultado ascendente o descendente del ejercicio, usaríamos un período = 6. Como prefiero examinar la tendencia en lugar de la media absoluta, voy a decirle que es igual a 4, con los que obtendré una línea con cuatro puntos de inflexión.
Y, que no se nos olvide, para utilizar funciones hay que incluir en uses el pas *TeeFunci*.

En el "**listado 3**" podemos ver como se consigue el gráfico que aparece en la "**figura 4**".

```
implementation
var
  aBeneficios:Array[1..12] of integer=(320,430,560,400,450,510,580,645,600,530,440,280);
{$R *.DFM}
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  SerieLineas:TLineSeries;
  SerieBarras:TBarSeries;
  SerieMedia:TLineSeries;
  x:byte;
begin
// Creamos la serie y la asociamos con su Chart
  SerieLineas:=TLineSeries.Create(Self);
  SerieLineas.ParentChart:=Chart1;

  SerieBarras:=TBarSeries.Create(Self);
//  SerieBarras.ParentChart:=Chart1;
//  Reemplazo esta linea por la función AddSeries

  SerieMedia:=TLineSeries.Create(Self);

// Aquí se configura la SeriaMedia como la función que es
  SerieMedia.SetFunction(TAverageteeFunction.Create(Self));

// Damos las características deseadas al Chart
  With Chart1 do
  Begin

// Incluimos la función AddSeries
    AddSeries(SerieBarras);
    AddSeries(SerieMedia);

    Title.Color:=clAqua;
    Title.text.add('Resultados del 1.999');
    Title.text.add('de la Empresa JLF S.A.');
    Legend.Visible:=False;
    LeftAxis.Title.Caption:='Miles de pesetas';
    LeftAxis.Title.Font.Color:=clRed;
    BottomAxis.Title.Caption:='Meses';
    BottomAxis.Title.Font.Color:=clRed;
    BottomAxis.Automatic:=False;
    BottomAxis.Maximum:=11;
    BottomAxis.Minimum:=0;
    Chart1.BottomAxis.LabelStyle:=talText;
  end;

  With SerieLineas do
  begin
    Marks.Visible:=False;
    for x:=1 to 12 do
      Add(aBeneficios[x],inttoStr(x),clBlue);
  end;
  With SerieBarras do
  begin
    Marks.Visible:=False;
    for x:=1 to 12 do
      Add(aBeneficios[x],inttoStr(x),clGreen);
  end;
```

```
// Asignamos a la función la fuente de datos y
// creamos el período.
  SerieMedia.DataSources.Add(SerieBarras);
  SerieMedia.FunctionType.Period:=4;
end;
```

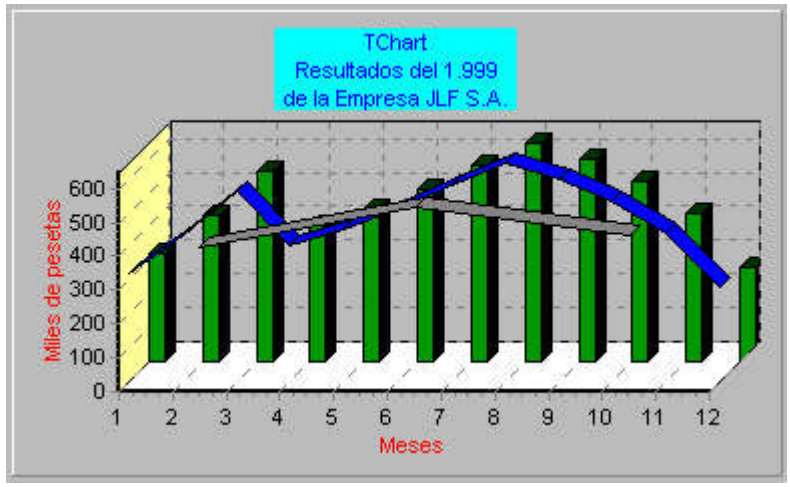

Figura 4

Y una observación al mismo: aparecen tres Series creadas, y he cambiado el fuente para que se vea que una de ellas aparece añadida al *Chart* a través del ParentChart, en tanto que para las otras dos he utilizado, la función AddSeries() solamente para que se vea que el resultado es similar.

Una clase especial: Tartas.

Y hablando de tipos gráficos de una sola variable, es decir, los que podemos hacer funcionar sin dar valores a los dos ejes de coordenadas, no hay mas remedio que incluir la más típica de todas: los gráficos de "tarta" o *Pie*.

Como realmente su funcionamiento es sencillo, vamos a pasarlo por alto con un ejemplo simple. Generando un gráfico de nuestros conocidos beneficios. Solo haremos pequeñas variaciones para atenernos siempre a una lectura eficaz, en este caso sí vamos a dejar los cuadros de valores, e incluso vamos a potenciarlos creando una matriz con los meses del año. Y en el ejemplo que tratamos, vamos a diferencia el mes en que mayor cantidad de beneficios se han obtenido. También omitiremos en la función Add cualquier sugerencia al color, personalmente me da lo mismo que agosto se dibuje de un color u otro, quien sea fanático de las mezclas necesitará crear una matriz de colores, personalmente le dejo a TeeChart que los escoja.

Eso sí, tengo un especial interés por resaltar aquel mes del año que ha producido mayores beneficios, y para ello nada mejor que entresacarlo del conjunto.

En el "**listado 4**" podemos apreciar lo que da lugar a la "**figura 5**"

```
implementation
var
  aBeneficios:Array[1..12] of
integer=(320,430,560,400,450,510,580,645,600,530,440,280);
  aMeses:array[1..12] of
string=('Enero','Febrero','Marzo','Abril','Mayo','Julio','Julio','Agosto','Septiembre','Octubre',
'Noviembre','Diciembre');
  {$R *.DFM}
```

```
procedure TForm1.Button1Click(Sender: TObject);
var
  SerieTarta:TPieSeries;
  x:byte;

begin

  SerieTarta:=TPieSeries.Create(Self);
  SerieTarta.ParentChart:=Chart1;
  Chart1.Title.Text.Clear;
  Chart1.Title.Text.add('Resultados del 1.999');
  With SerieTarta do
  begin
    for x:=1 to 12 do
      Add(aBeneficios[x],aMeses[x]);
    SerieTarta.ExplodeBiggest:=20;
  end;
end;
```

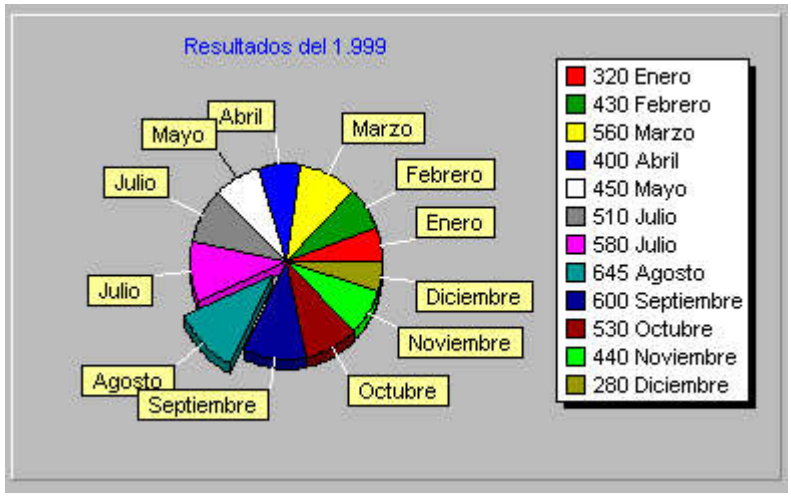

Figura 5

Y una observación al mismo: aparecen tres Series creadas, y he cambiado el fuente para que se vea que una de ellas aparece añadida al *Chart* a través del ParentChart, en tanto que para las otras dos he utilizado, la función AddSeries() solamente para que se vea que el resultado es similar.

Gráficos comparativos con más de una serie.

Bien, nuestro ejemplo sobre beneficios es correcto, obtenemos una media en la cual le damos los períodos que estimamos, pero ¿y si de lo que se trata es de comparar dos ejercicios y obtener la media de ambos?

Dar un valor de período distinto no nos va a ser de utilidad en este caso, al tomar más de una fuente de datos y no obstante, tenemos que dar la instrucción si queremos que se dibuje.

Para este ejemplo creamos dos matrices a las que incorporamos los valores de los 12 meses como viene siendo habitual, y consideramos que son las correspondientes a dos años distintos.

Por lo demás, la única diferencia llamativa es la asignación a la función Average de las *Series* que nos interesan, puesto que se acumulan de una en una para llegar al resultado. De manera que si tuviesen un número de ellas mayor, podríamos elegir por este sistema las que nos interesan, pero tanto para esta como para cualquier otra función de las disponibles.

En este caso introduzco por primera vez marcas para conseguir los resultados, pero tan solo lo hago con

la tercera *Serie* resultante de las otras dos. Especifico el color del "cartel":

Marks.BackColor:=clWhite;

La longitud de la línea que va a separar el valor de el gráfico en sí:

Marks.ArrowLength:=20;

Y le especifico que lo que quiero obtener es precisamente los valores que está tomando.

Marks.Style:=smsValue;

¿Por qué en esta que hemos llamado SerieMedia hacemos este proceso y no en las anteriores? Pues por pura conveniencia. Para calcular las dos de beneficios nos hemos valido de unas matrices y por lo tanto, en este caso, de valores conocidos, mientras que ahora no sabemos cuales son éstos, salvo que procedamos a su cálculo en cada punto.
En el "**listado 5**" se puede ver el código, así como el resultado gráfico en la "**figura 6**"

```
var
  Form1: TForm1;
  aBeneficio98:Array[1..12] of integer=(320,430,560,400,450,510,580,645,600,530,440,280);
  aBeneficio99:Array[1..12] of integer=(380,480,560,480,590,530,750,700,645,500,430,290);
implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
SerieBar1:TBarSeries;
SerieBar2:TBarSeries;
SerieMedia:TLineSeries;
x:byte;
begin

// Creación
  SerieBar1:=TBarSeries.Create(Self);
  SerieBar2:=TBarSeries.Create(Self);
  SerieMedia:=TLineSeries.Create(Self);

// Definición de función
  SerieMedia.SetFunction(TAverageteeFunction.Create(Self));

// Características de Chart e inclusión de las Series

  With Chart1 do
  Begin

    AddSeries(SerieBar1);
    AddSeries(SerieBar2);
    AddSeries(SerieMedia);
    Title.Text.Clear;
    Title.text.add('Resultados del 1.999');
    Title.text.add('de la Empresa JLF S.A.');
```

```
    LeftAxis.Title.Caption:='Miles de pesetas';
    LeftAxis.Title.Font.Color:=clRed;

    BottomAxis.Title.Caption:='Meses';
    BottomAxis.Title.Font.Color:=clRed;
    BottomAxis.Automatic:=False;
    BottomAxis.Maximum:=11;
    BottomAxis.Minimum:=0;
    BottomAxis.LabelStyle:=talText;

    Legend.Visible:=False;

  end;

  With SerieBar1 do
  begin
    Marks.Visible:=False;
    for x:=1 to 12 do
      Add(aBeneficio98[x],inttoStr(x));
  end;

  With SerieBar2 do
  begin
    Marks.Visible:=False;
    for x:=1 to 12 do
      Add(aBeneficio99[x],inttoStr(x));
  end;

 // Características de Marcas para obtener valores medios
 // Añadimos somo fuente de datos las dos series
  With SerieMedia do
  begin
    Marks.Visible:=True;
    Marks.BackColor:=clWhite;
    Marks.ArrowLength:=20;
    Marks.Style:=smsValue;

    DataSources.Add(SerieBar1);
    DataSources.Add(SerieBar2);

    FunctionType.Period:=4;
  end;
 end;
```

¿Y si damos valores a ambos ejes de coordenadas?

Pues la realidad es que no tiene ninguna secreto y no hay tampoco una diferencia sustancial en el funcionamiento con relación a lo que hemos visto.

Hasta ahora hemos utilizado la función `Add` y tan solo se trataría se utilizar `AddXY` ¿En que casos? Pues es muy claro ¿por qué podemos dar un solo valor a un gráfico de barras?, porque se asume que su origen en el eje X es cero, pero si lo que tratamos es de dibujar, por ejemplo, un punto no se asume nada. Dentro de un eje de coordenadas está representado a la fuerza por su valor correspondiente a X y a Y, de la misma forma que actuábamos antes:

Addxy(valor_X,valor_Y,texto,color);

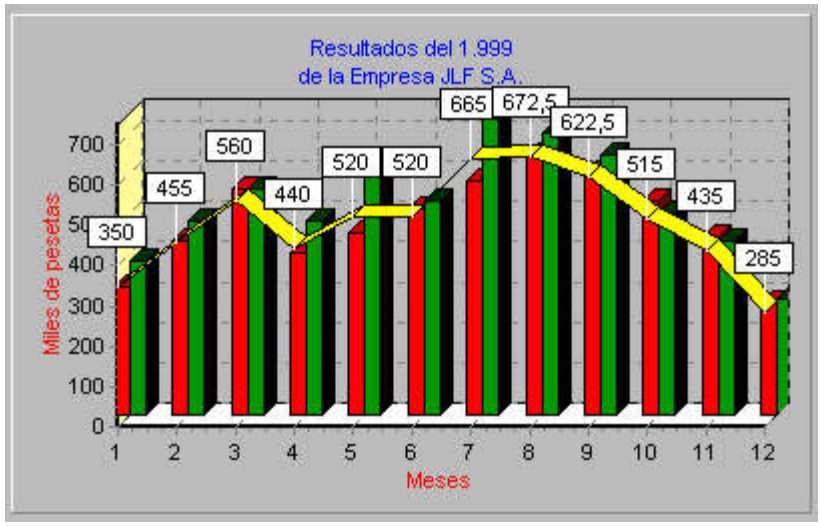

Figura 6

Y la utilidad de ello es mucho mayor de lo que parece a simple vista. En la versión estándar no existen gráficos de este tipo pero podemos idear la forma de construir el típico de “nube de puntos”.

Y en cuanto a las líneas con mayor motivo, una línea con una sola coordenada, como hacíamos antes, coincide con uno de los ejes, hay que situarla sin más remedio en el espacio de dos dimensiones. Y ya de por sí los gráficos de líneas son interesantes, pero habida cuenta que las únicas curvas que se pueden confeccionar mediante TeeChart, y aún así en la versión profesional, son Beziers, pero nada nos impide (y esto lo veremos) construir cualquier otro tipo, con los cálculos adecuados, siempre que la longitud de las líneas sea muy pequeña, en definitiva, uniones de dos puntos muy próximos que dan como resultado una curva.

Pero esto, como otros temas, lo dejaremos para el próximo.

Generación de gráficos con TeeChart (IV)

Los gráficos y el trabajo diario.

Desconozco el motivo por el cual los componentes gráficos parecen relegados al olvido en la mayoría de las aplicaciones y tan solo se utilizan en algunos tipos concretos de desarrollos, cuando la realidad es que ofrecen un toque de profesionalidad y acabado en unos casos, y en otros son imprescindibles. ¿Quién se imagina estudios comparativos de cualquier tema sin la parafernalia de colores, barras, tartas, etc.? En la mayor parte de los casos ayudan a visualizar y por lo tanto percibir de una forma genérica y aproximada lo que es engorroso a base de tiras de números. Vamos a intentar acercarnos a lo que serían los componentes TeeChart dentro de las aplicaciones.

Entrada de datos desde tablas.

Hasta ahora hemos utilizado datos ficticios para manejarnos en las posibilidades gráficas. No es esta la realidad del día a día, tendremos datos e intentaremos dar una solución gráfica a los mismos. Si estos están contenidos en cualquier componente del tipo *Ttable*, *Tquery*, etc. son susceptibles de manejar y la forma idónea para ello es a través de **TDBChart**. No son muchas las diferencias con el resto de los que hemos visto, e incluso *Tchart* también incluye la opción de tomar datos desde tablas.

Parece obvio que lo que vamos a reflejar son resultados, si estamos generando una contabilidad no tiene ningún sentido hacer un reflejo gráfico de los asientos, aunque sí, probablemente, del resultado de cuentas o, como es típico, de períodos. En cualquier caso, lo que tiene que recibir este componente son datos previamente elaborados.

Para poner un ejemplo he creado una tabla simple que muestra los resultados de una empresa por trimestres, (en este caso utilizo Paradox 7, pero me sería indiferente). Para ello he considerado el campo trimestre de tipo fecha y otro más, al que llamo resultado, que es un numérico. Ya tengo listos los datos a reflejar, por supuesto de la misma forma podían ser el resultado de una consulta SQL, pues lo único que necesito es un *TdataSource*, y la lista de campos por la que estoy interesado. Abriendo el editor podemos ver una imagen similar a la que aparece en la “figura 1”.

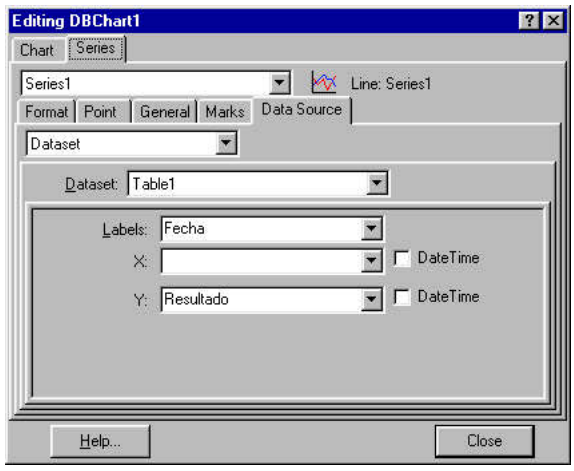

Figura 1

Como se puede observar en la misma he creado una Serie de tipo Line, y accediendo a Series, DataSource nos encontramos con un primer combo. Una de las posibilidades que ofrece es utilizar un DataSet, y a ella nos dirigimos. Este ha sido creado previamente y va a permitirnos tomar contacto con los datos que se encuentran en Table1. Sabemos los datos que tenemos, simplemente reflejar una serie de fechas y sus cantidades relacionadas, no nos ofrece mucha dificultad. En el eje de las X queremos que figuren estas fechas, se lo hacemos constar en la opción Labels, y sobre el eje de las Y el campo numérico correspondiente. Poco más debemos hacer, si queremos configurar valores máximos y mínimos, tipos de etiquetas y colores, el rótulo del Chart, es decir, las opciones normales que se han estado viendo en números anteriores y que no entramos de nuevo en ellas.

Si abrimos la tabla, ya en tiempo de diseño podemos observar los resultados y por lo tanto, ajustar la visualización y las opciones dependiendo de éstos. Y en este proceso me doy cuenta de que resulta de interés, por ejemplo, conseguir la media de los beneficios de ese período. En principio y en teoría tenemos que dar al componente los datos elaborados, pero como se mencionó en su momento también existe la posibilidad de crear Series que son funciones y aunque en la versión de TeeChart estándar son muy pocas, sí nos encontramos con la denominada *Average*. Perfecto, no nos cuesta nada crear una segunda serie, pero en este caso no del tipo general sino de función. Y ¿sobre qué tiene que hallar la media? Pues claramente sobre los valores que estamos obteniendo desde Table1, así en Series, DataSource, tras elegir la Serie2 si no la hemos denominado de distinta forma, nos encontramos con que

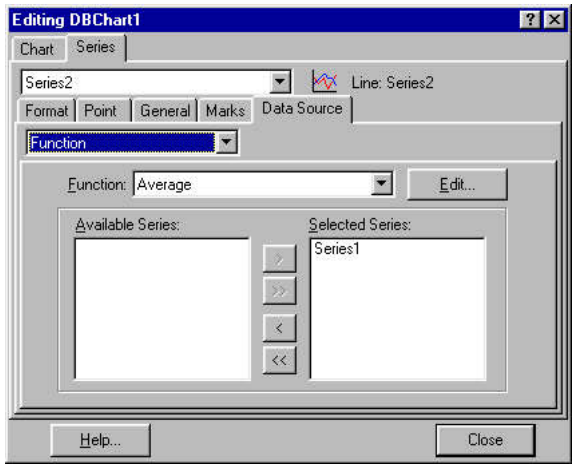

Figura 2

podemos elegir las series disponibles y seleccionados, desplazando a la derecha, la Serie1 que, aparte de ser la única que hemos creado con anterioridad y no tenemos más remedio, también es la que contiene los valores de nuestra tabla, quedando como se puede ver en la "figura 2". A nadie le ha pasado desapercibido que desde esta misma pantalla podemos especificar que lo que queremos es una función y el tipo de la misma. El resultado sería idéntico a si la seleccionamos de esa manera en el momento de su creación.

Con todo ello habremos conseguido de forma muy simple elaborar el gráficos de la "figura 3".

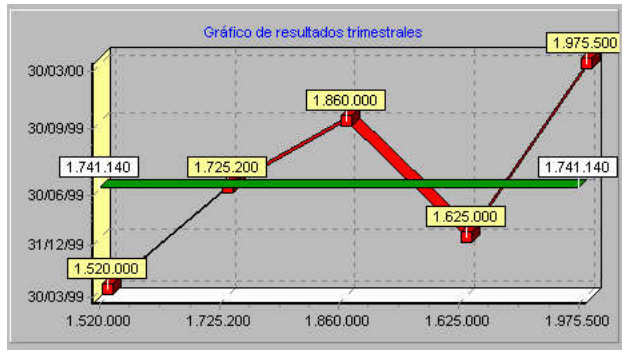

Figura 3

Personalización de un Chart en ejecución.

La forma más simple, le decimos "mire, haga usted lo que quiera" y a continuación le incluimos tan solo esto dentro de un botón:

```
Uses EditChar,DBEditCh;

procedure TForm1.Button1Click(Sender: TObject);
begin
      ChartEditor1.Chart:=Chart1;
      ChartEditor1.Execute;
end;
```

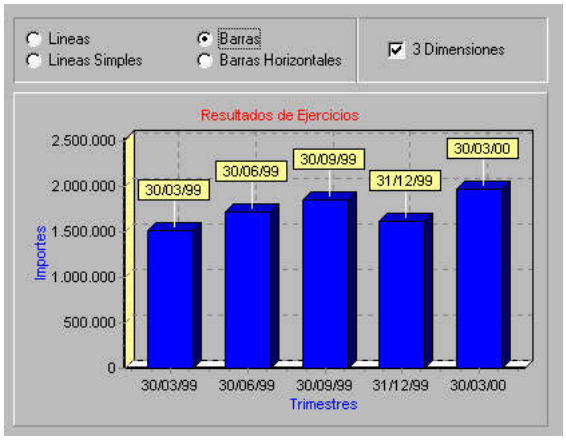

Figura 4

Lo que hemos hecho ha sido dejar al usuario el Editor para que haga en ejecución lo que se le ocurra (un comentario: este sistema lo pruebo en la versión profesional del paquete. No tengo la seguridad que en la estándar sea igualmente válido, ya que otros Executes no lo son, no obstante la incluyo por tener razones para pensar que este sí se soporta).

Esto, como todos sabemos, es un arma de doble filo. Puede tanto evitarnos el preparar los gráficos como obligarnos a una estancia permanente en la oficina del cliente.

Realmente suele ser más simple dejar el cambio de los atributos que resulten más interesantes y ejecutar las nuevas configuraciones a toque de tecla. En el ejemplo de utilización de un DBChart se complica si se crean Series en tiempo de diseño, pues se han de pasar a estas las propiedades del DataSource correspondiente, teniendo muy en cuenta la asignación de propiedades que aparece en el "figura 1". Un sistema muy simple para realizar esto es crear previamente las series y activar la que el usuario elija. En este caso lo que haré será tomar los valores en dos matrices y utilizar el Chart típico, para conseguir lo que aparece en la "figura 4".

Generación de gráficos con TeeChart (IV)

Nota: El sistema que utilizo no me parece el ideal en absoluto, pero pretendo simplificar el código todo lo que puedo.

```
uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, Db,
DBTables, ExtCtrls, StdCtrls, TeeProcs, TeEngine, Chart, Series;
```

Crearemos las Series en ejecución, por lo tanto tenemos que incluirlas a mano en *Uses*

```
var
  Form1: TForm1;
  aFecha: Array of string;
  aResultado: Array of Integer;
```

Puesto que vamos a trasladar los valores a representar a dos matrices, ya que estamos utilizando un componente del tipo *Tchart* las incluyo en Interface sin dimensionar, lo haremos en el siguiente paso cuando activemos el formulario.

```
procedure TForm1.FormActivate(Sender: TObject);
var nReg,x:integer;
begin
  Table1.Active:=True;
  nReg:=Table1.RecordCount;
  SetLength(aFecha, nreg);
  SetLength(aResultado,nreg);
  Table1.Active:=True;
  try
    for x:=0 to nReg-1 do
    begin
      aFecha[x]:=DatetoStr(Table1.FieldbyName('fecha').value);
      aResultado[x]:=Table1.FieldbyName('resultado').value;
      Table1.Next;
    end;
  Except
    Showmessage('Error')
  end;
end;
```

Recordemos que tenemos los datos en una tabla de Paradox, tan solo hemos dimensionado las matrices con tantos elementos como registros tiene y hemos asignado a cada uno su valor.

Como hemos visto en la “figura 4” creamos tantos *RadioButton* como tipos de Series vayamos a crear. Para ello a cada uno le he dado un valor en su propiedad Tag.

```
procedure TForm1.RadioButtonClick(Sender: TObject);
var
  Serie1:TLineSeries;
  Serie2:TFastLineSeries;
  Serie3:TBarSeries;
  Serie4:THorizBarSeries;
  x:integer;
begin
  if Chart1.SeriesCount>0 then
    Chart1.SeriesList.Clear;
// para evitar la superposición de gráficos borro la lista de Series
// cada vez que se dispara el evento. Y a continuación procedo
// a utilizar el constructor, asocio la Serie con el Chart y dibujo
// mediante la función Add()
  Case (Sender As TRadioButton).Tag of

    1: begin
         Serie1:=TLineSeries.Create(Self);
         With Serie1 do
```

```
          Begin
            ParentChart:=Chart1;
            for x:=0 to Length(aFecha)-1 do
              Add(aResultado[x],aFecha[x],clBlue);
          End;
        end;
    2: begin
          Serie2:=TFastLineSeries.Create(Self);
          With Serie2 do
          Begin
            ParentChart:=Chart1;
            for x:=0 to Length(aFecha)-1 do
              Add(aResultado[x],aFecha[x],clBlue);
          End;
        end;
    3: begin
          Serie3:=TBarSeries.Create(Self);
          With Serie3 do
          Begin
            ParentChart:=Chart1;
            for x:=0 to Length(aFecha)-1 do
              Add(aResultado[x],aFecha[x],clBlue);
          End;
        end;
    4: begin
          Serie4:=THorizBarSeries.Create(Self);
          With Serie4 do
          Begin
            ParentChart:=Chart1;
            for x:=0 to Length(aFecha)-1 do
              Add(aResultado[x],aFecha[x],clBlue);
          End;
        end;
  End;
end;
```

El CheckBox en el que se elige si el funcionamiento es a 3D o no, se soluciona fácilmente

```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if CheckBox1.Checked then
    Chart1.View3D:=True
  else
    Chart1.View3D:=False;
end;
```

Impresión.

En cualquiera de ellos, tanto si la fuente de datos es una tabla como si no, se puede utilizar directamente la sentencia

```
Chart1.Print;
```

Que producirá una salida por la impresora por defecto. Además de las opciones relacionadas

PrintPortrait
PrintLandscape
PrintMargins
PrintPartial
PrintRect
PrintResolution

El Preview que aparece al pulsar en edición con el botón principal sobre el gráfico, se puede realizar con

las propiedades del Chart e incluyendo el *PrinterSetupDialog* pues al fin y al cabo, es lo que hace en ese momento.

En cualquier caso, el sistema que en realidad resulta atractivo dada la integración entre TeeChart y Quick Report es el componente QRChart. Tiene el mismo comportamiento que los componentes anteriores y se encuentra en la paleta Qreport.

Utilicemos un componente de éste tipo al que adjuntamos un QRChart. Para la toma de datos especifiquemos que utilice un DataSet, de manera que como se ve en la "figura 5" al ejecutar

```
QuicRep1.Preview;
```

visualizamos tanto los datos contenidos en las bandas correspondientes como el gráfico relacionado.

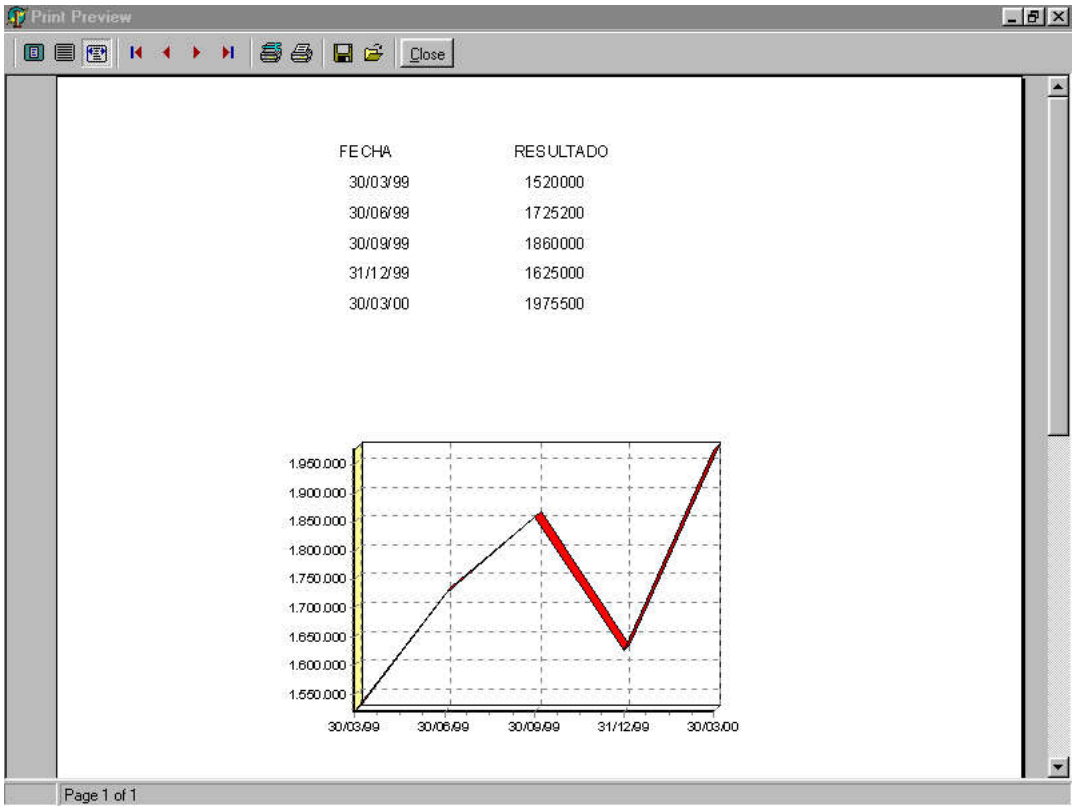

Figura 5

Gráficos y curvas procedentes de operaciones matemáticas.

Una de las posibilidades que mas fruto personal he obtenido de estos componentes ha sido precisamente introducirme en lo que no hacen por sí solos, y debo de decir que con unos resultados que para sí los quisieran los sistemas gráficos de algunas de las hojas de cálculos más populares del mercado.

No se trata ahora de generar complejos sistemas de cálculos matemáticos (que por otra parte necesitaría quien los hiciese por mi), tan solo de ver la representación de distintas ecuaciones en forma gráfica.

La primera tentación sería la representación del tipo *Point*, y de paso la menos adecuada en la práctica, ya que no se dibujan como tal, sino como círculos, cuadrados, etc. que no conforman una línea mas que de forma bastante burda. Lo ideal en este caso es la utilización nuevamente de Series *Tlines*. Si pretendemos representar la ecuación de una recta, imaginemos de la forma X=(2*Y)+25

```
var
  x,y,a: Integer;
begin
  for y:=1 to 20 do
  begin
```

```
    for a:=1 to 300 do
    begin
      x:=2*y+25;
      Series1.addxy(x,y,'',clblue);
    end;
  end;
```

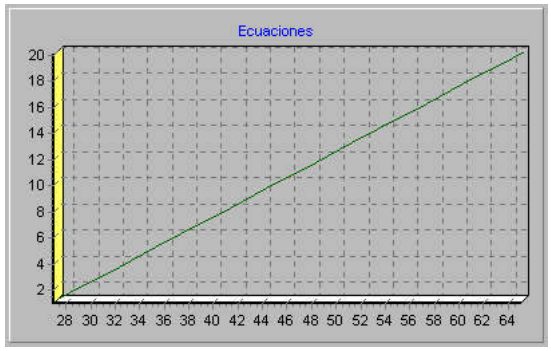

Figura 6

Los valores que he dado a la recta en los bucles, evidentemente son inventados. El resultado sería el que se ve en la “figura 6”.

Un caso especial. Series Bézier.

Vaya por delante el agradecimiento a mi amigo Ignacio Alvarez y a la cantidad de pastillas para el dolor de cabeza que tuve que consumir en su día tras sus explicaciones :).

No es momento para hablar sobre API de Windows, pero sí un comentario. El GDI soporta lo que denomina “primitivas” o sistemas básicos de gráficos ¿por qué menciono esto? Por que aparte de los arcos y de las rectas (ambos con sus variantes) tan sólo permite un sistema de curvas que es el **Bézier**, estas se consideran como unas de las primarias para realizar gráficos de curvas, de manera que otro tipo de curvas, en general no soportado por el entorno gráfico, como pueden ser elípticas, se consideran una aproximación a ellas. Su base son puntos, los dos exteriores “aferran” a los extremos de la curva y tradicionalmente otros dos actúan de “imán”, de manera que se crea una fuerza hacia ellos sin llegar a tocarlos, y ésta depende de sus valores.

Este es el que utiliza TeeChart, de manera que uno de los tipos de Series y el único estrictamente que deja las rectas al margen tiene ese nombre.

Pues bien, este concepto que se considera básico para la consecución de cualquier tipo de curvas y prácticamente un axioma no lo es tanto, o no lo entiendo de la misma forma. Las Bézier, con toda su importancia, son un tipo de otro más genérico denominado **Splines** con el cual se pueden manejar formatos irregulares y lo que resulta más importante, de alguna manera hipotética salta la barrera de las dos dimensiones a la que las Bézier están sujetas. Si dibujásemos desde hojas de cálculo u otros programas una hipérbola irregular en la cual su cúspide sobrepasase los ejes de coordenadas, de tal forma que no se correspondiese con ninguno de los puntos representados, pero tampoco tendiese a ningún otro superior, esos programas hallarían una aproximación inexacta, para ello es necesario la utilización de muchos más elementos de los cuales en principio no disponemos, ni siquiera nos atendríamos de forma real a una ecuación de tercer grado que podría resolverse sin problemas.

En definitiva, nos encontramos con una serie de puntos que se han de ajustar a una curva, el resultado no es único, al contrario, las soluciones son infinitas. Pero buscamos aquella que se ajuste con “suavidad”, si uniésemos los puntos de dos en dos por medio de las típicas Bezier (de otra manera serían rectas como es normal) cada fragmento solo sería coherente con esos dos puntos, no formaría un todo lógico. Para conseguirlo podemos utilizarlas, pero en forma de parábolas, es decir, buscamos que en los puntos intermedios las curvas se continúen, esto es, que las derivadas del punto por su parte inferior y superior coincidan.

Como no se trata aquí de clases matemáticas y además sería el menos indicado, habría que ir evolucionando sobre el planteamiento anterior hasta llegar a comprobar que realmente las parábolas no nos sirven, darían una curva irregular, y por lo tanto el planteamiento de las Bézier tampoco, llegaríamos a la conclusión tan extraña como que dos puntos se han de unir por ecuaciones cúbicas para cada pareja consecutiva e ellos, lo que parece un contrasentido, que se resuelve considerando las derivadas primera y segunda y teniendo en cuenta que éstas últimas tienen valor cero en los extremos. Habríamos conseguido pues un **Spline cúbico natural**.

Alguno que haya llegado hasta aquí se preguntará qué ha hecho para merecer esto y a qué viene. Pues tan solo a justificar dos cosas, que no hay ninguna representación que no se pueda hacer, a pesar de que los componentes no consideren mas que un número limitado de Series, y a introducirnos en la importancia de

las ecuaciones.

Ecuaciones.

En el ejemplo anterior trazamos la ecuación de una recta, ahora vamos a modificar ligeramente el código para dar lugar a una curva. Es claro que la primera sería una ecuación de grado uno, y en este caso lo sería de grado dos.

```
var
  y,a: Integer;
  x:extended;
begin
  for y:=1 to 20 do
  begin
    for a:=1 to 300 do
    begin
      x:=power(y,2)+16;
      Series1.addxy(x,y,'',clblue);
    end;
  end;
```

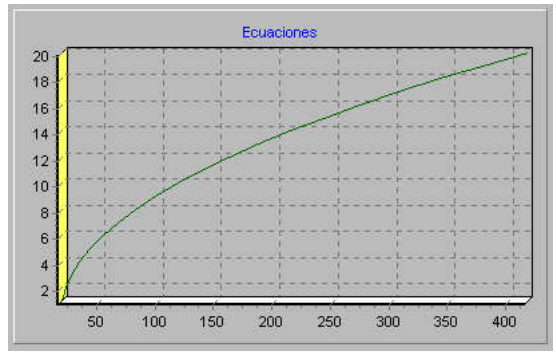

Figura 7

Con esto representaríamos, partiendo de una ecuación de segundo grado, la curva que se ve en la "figura 7"

Lo que no se ha visto.

La pretensión de esta serie de cuatro artículos ha sido tan solo la de proporcionar un acercamiento a unos componentes que, como decía al inicio, parecen en el olvido. Lo que no he intentado en ningún momento ha sido profundizar en los mismos, sus posibilidades son mucho mayores de lo que aquí haya podido plantear, de forma que opciones como pueden ser gráficos con desplazamientos, zooms, etc. ni siquiera los he mencionado.

Tampoco he considerado los componentes como tal, en el sentido estricto del término, pues no he entrado en ningún momento a estudiar los eventos de los mismos, no he utilizado, pues, las respuestas de éstos a nuestras acciones, y todo ello por dos motivos: necesitaría una cantidad de artículos muy alta para intentar, mínimamente, agotar las posibilidades, y mi imaginación es muy limitada a la hora de mostrar éstas con ejemplos medianamente realistas.