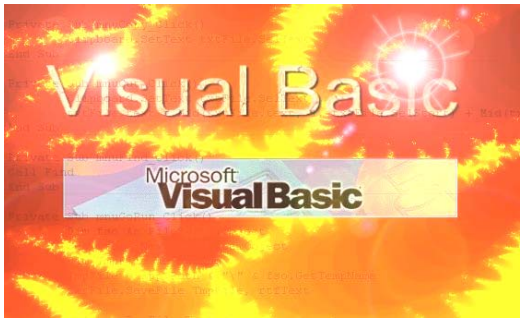
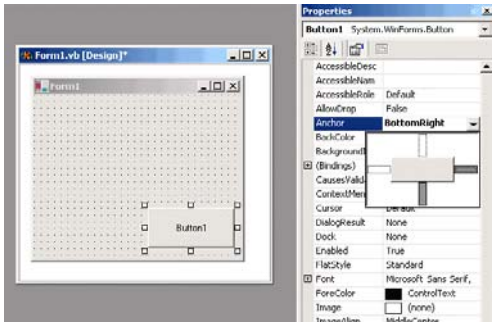
CURSO

Curso Completo de Visual Basic 6.0





Escuela Superior de Ingenieros Industriales
UNIVERSIDAD DE NAVARRA
Javier García de Jalón · José Ignacio Rodríguez
Alfonso Brazález · Patxi Funes · Eduardo Carrasco · Jesús Calleja

6.3 CONTROLES GRÁFICOS

Visual Basic 6.0 dispone de varios controles con los que insertar gráficos en un formulario.
Algunos tienen más posibilidades que otros y es necesario conocerlos bien.
A continuación se verán los controles ***line***, ***shape***, ***image*** y ***picture***.

6.3.1 Control line 

Es el control gráfico más elemental, ya que carece de propiedades como *text*, *caption* y *value*.
Además no reconoce ningún evento, por lo que su misión es casi exclusivamente decorativa.
El control ***line*** permite dibujar líneas en un ***formulario*** o en un control ***picture***. Las propiedades más importantes son las coordenadas de los puntos extremos (***X1***, ***Y1***, ***X2*** e ***Y2***), la anchura en pixels (***BorderWidth***), el estilo de la línea (***BorderStyle***) - continua, a trazos, etc.- que sólo está activo cuando la anchura es 1 pixel, el color (***BorderColor***) y el nombre (***Name***). La línea puede estar visible o no (***Visible***), y existe la propiedad ***Index***, que permite crear *arrays* de líneas.

6.3.2 Control shape 

Este control es en muchos aspectos similar al control ***line***: tampoco tiene las propiedades *text*, *caption* y *value*, ni reconoce eventos. Se diferencia en que admite formas geométricas más complejas, que vienen definidas por la propiedad ***shape***, que admite los valores siguientes: cuadrado (***Square***), rectángulo (***Rectangle***), círculo (***Circle***), elipse (***Oval***), cuadrado redondeado (***Rounded Square***) y rectángulo redondeado (***Rounded Rectangle***).

Además de las propiedades correspondientes al tamaño y posición, las propiedades más interesantes del control ***shape*** son las siguientes: *BackColor*, *BackStyle*, *BorderColor*, *BorderStyle*, *BorderWidth*, *FillColor*, *FillStyle*, *DrawMode*. Un control ***shape*** puede estar visible o no (***Visible***), y existe la propiedad ***Index***, que permite crear *arrays* de ***shapes***.

6.3.3 Ejemplo 6.1: Uso de los controles line y shape

La **Figura 6.4** muestra un formulario en el que se han dibujado tres controles ***Line*** y dos controles ***Shape***.
Las tres líneas se han dibujado con la propiedad ***BorderWidth=1***, pues si no la propiedad ***BorderStyle*** no surte efecto. La propiedad ***BorderStyle*** es *2-Dash* para la segunda línea y *3-Dot* para la tercera.

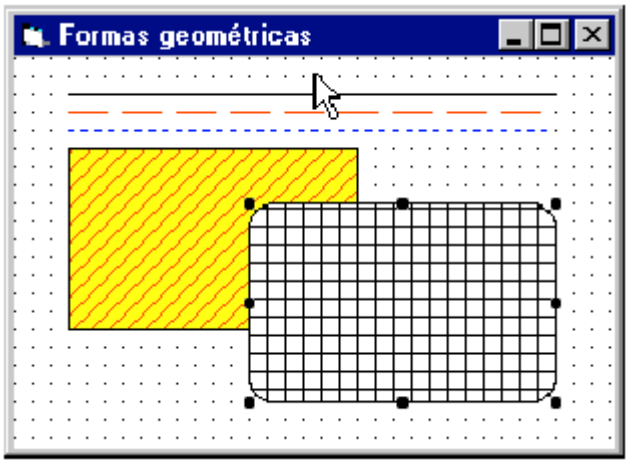


Figura 6.4. Controles ***Line*** y ***Shape***.

Después se han dibujado dos controles ***shape*** llamados ***shpRect*** y ***shpRRec***, cuyas propiedades *Shape* están respectivamente a *0-Rectangle* y a *4-Rounded Rectangle*.
La propiedad ***BackColor*** está en amarillo para ***shpRect*** y en blanco para ***shpRRect***.

En ambos casos ***BackStyle*** está en *1-Opaque*, pues si no el color de fondo no surte efecto. La propiedad ***FillColor*** (que determina el color de las líneas de rayado) está en rojo para ***shpRect*** y en negro para ***shpRRect***. Finalmente, la propiedad ***FillStyle*** que determina el tipo de rayado está en *5-Downward Diagonal* para ***shpRect*** y en *6-Cross* para ***shpRRect***. Como la propiedad ***DrawMode*** está en *13-Copy Pen* para ambos controles, ***shpRRect*** se superpone sobre ***shpRect*** porque ha sido creada sobre él con posterioridad.

6.3.4 Control image

El control ***image*** es un contenedor de gráficos ***bitmap***, ***iconos***, ***metafile***, ***enhanced metafile***, ***GIF*** y ***JPEG***. Este control admite ya una amplia colección de eventos, por lo que es ya un control con un papel mucho más activo que los anteriores.

Las propiedades más propias e importantes de este control son las propiedades ***picture*** y ***stretch***. La propiedad ***picture*** sirve para relacionar este control con el fichero que contiene el gráfico que se desea representar, a través del cuadro de diálogo ***Load Picture*** que permite elegir el fichero a insertar. El fichero deberá ser de uno de los tres tipos admitidos. Según el fichero elegido, la propiedad ***picture*** tendrá uno de los tres valores siguientes: ***icon*** (ficheros *cur*, *ico*), ***bitmap*** (*bmp*, *gif*, *jpg*) o ***metafile*** (*wmf*, *emf*).

La propiedad ***stretch*** indica cómo se comporta el control ***image*** al introducir en él el contenido del fichero gráfico. Por defecto, cuando se crea un control ***image*** arrastrando en el formulario con el ratón esta propiedad tiene el valor ***false***. Estando la propiedad ***stretch*** en ***false*** el tamaño del control se ajusta al tamaño del *bitmap* o del ***metafile*** que se introduce en dicho control.

Por el contrario, si dicha propiedad está en ***true*** el gráfico que proviene del fichero se adapta al tamaño de control.

Se puede tratar de modificar el tamaño del gráfico en modo de diseño (con el ratón o cambiando las propiedades de tamaño del control). Si el gráfico es un *bitmap* y la propiedad ***stretch*** está en ***false*** el tamaño de la imagen no cambia aunque cambie el del control (quedando en la esquina superior izquierda si el control se hace más grande, o quedando parcialmente oculta si alguna de las dimensiones del control se hace más pequeña que la del *bitmap*. Si la propiedad ***stretch*** está en ***true*** el *bitmap* se adapta al tamaño del control y su tamaño se cambia con el de éste.
Los gráficos *metafile* siempre se pueden cambiar de tamaño en modo de diseño, tanto si ***stretch*** está en ***true*** como si está en ***false***.

Existen otras formas de cargar un gráfico en un control ***image***, además de utilizar la propiedad ***picture*** en modo de diseño, como se ha visto anteriormente. Una segunda forma, utilizable también en modo de diseño, es hacer ***Copy*** y ***Paste*** a partir de un gráfico contenido en otra aplicación como ***Paint Shop Pro*** o ***Excel***.

En modo de ejecución se puede copiar el contenido de un control ***image*** en otro control del mismo tipo por medio de una sentencia de asignación en la forma:

```
imgCuadro.picture = imgCaja.picture
```

y se puede también cargar una imagen de un fichero utilizando el procedimiento ***LoadPicture***, por ejemplo en la forma siguiente (habrá que estar seguro de que existe el fichero):

```
imgCuadro.picture = LoadPicture("G:\graficos\pc.wmf")
```

Aunque el control ***image*** admite algunos eventos (*Click*, *DblClick*, *DragDrop*, *DragOver*, *MouseUp*, *MouseDown*, *MouseMove*), sus posibilidades son también limitadas. Por la forma en que se dibuja, el control ***image*** no puede estar sobre otro control, como por ejemplo un botón (ver los *layers*, más adelante en este capítulo). Tampoco puede contener otros controles en su interior: sólo puede contener gráficos. Finalmente, este control no puede obtener el ***focus*** y por tanto no puede responder a acciones desde el teclado. El control ***picture***, que se verá a continuación, resuelve estas limitaciones aunque presenta la desventaja de ser más lento en dibujar que el control image.

6.3.5 Control Picture Box 

Este es el control gráfico más potente y general de ***Visual Basic 6.0***. Se trata de una especie de formulario reducido, pues puede contener imágenes y otros tipos de controles tales como botones, shapes, labels, cajas de texto, etc.

Con respecto a los *bitmaps*, el control ***picture*** se comporta de modo diferente que el control ***image***. El control ***picture*** no tiene propiedad ***stretch***, con lo cual al cargar un icono o un bitmap siempre aparecen con su tamaño natural (tal y como se puede observar en la Figura 6.5).

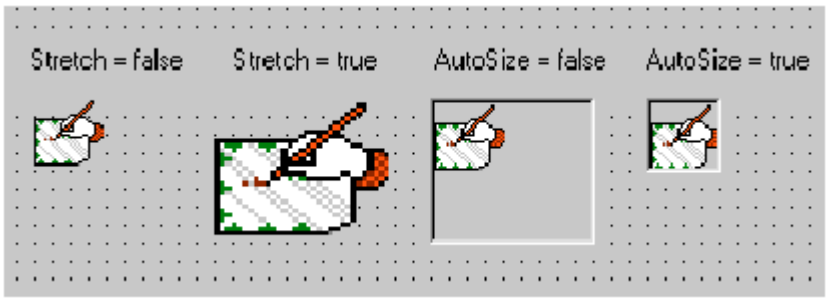


Figura 6.5. Comparación entre ***image*** y ***picture*** con *bitmaps*.

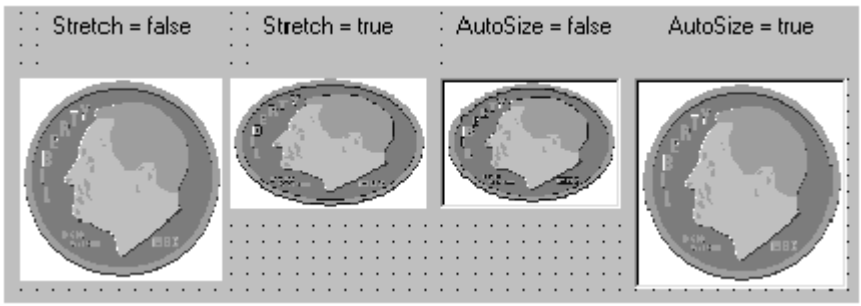


Figura 6.6. Comparación entre ***image*** y ***picture*** con *metafiles*.

Sin embargo el control ***picture*** tiene la propiedad ***AutoSize***, que por defecto está en ***false***.

Cuando se carga un *bitmap* con ***AutoSize*** en ***false*** el gráfico aparece en la esquina superior izquierda del control; sin embargo, si ***AutoSize*** está en ***true*** el control ***picture*** adapta su tamaño al del *bitmap* que es cargado. La Figura 6.5 muestra los resultados de introducir un icono en un control ***image*** (***Stretch***: *false* y *true*) y en un control ***picture*** (***AutoSize***: *false* y *true*).

Los gráficos *metafile* se comportan de un modo diferente, según puede verse en la **Figura 6.6.**

En el control ***image*** se cargan con su verdadero tamaño si la propiedad ***stretch*** es ***false***, mientras que se adaptan al tamaño del control si dicha propiedad es ***true***. Con el control ***picture*** se adaptan al tamaño del control si ***AutoSize*** es ***false***, mientras que se cargan con su propio tamaño si es ***AutoSize*** es ***true***.

En el control ***picture*** (al igual que en los formularios) son importantes las cuatro propiedades relacionadas con el color: *BackColor*, *ForeColor*, *FillColor* y *FillStyle*. La propiedad ***BackColor*** controla el color de fondo del control. La propiedad ***ForeColor*** controla el color del texto que se escribe en el control (con el método ***print***, por ejemplo, como luego se verá). Las propiedades ***FillColor*** y ***FillStyle*** no afectan directamente al control sino a los elementos gráficos que se dibujen sobre él con métodos tales como ***line*** y ***circle***, que se verán a continuación. ***FillStyle*** determina el tipo de relleno o *pattern* (líneas horizontales, verticales, inclinadas, cruzadas), mientras que ***FillColor*** determina el color de estas líneas del relleno.

6.4 MÉTODOS GRÁFICOS

Tanto los ***formularios*** como los controles ***picture*** pueden albergar otros tipos de controles. Además es posible escribir texto y dibujar directamente sobre ellos por medio de ciertos ***métodos*** de ***Visual Basic***. Por defecto estos métodos actúan sobre el ***formulario activo***. Si se desea que actúen sobre un control ***picture*** hay que precederlos por el nombre del control y el operador punto.

6.4.1 Método print

En tiempo de ejecución se puede escribir texto en un ***formulario*** o en un control ***picture*** por medio del método ***print***. La forma general de este método es la siguiente:

```
objeto.print {spc(n)|tab(n)} expresion poschar
```

donde ***spc(n)*** es opcional y sirve para insertar ***n*** caracteres en la salida; ***tab(n)*** es también opcional y sirve para posicionar la salida en una posición absoluta determinada por ***n*** con un tabulador. Si ***tab*** se utiliza sin argumentos lleva al comienzo de la siguiente ***región de salidas***; *expresion* representa cualquier expresión cuyo resultado sea un número o una cadena de caracteres. *poschar* indica dónde se imprimirá el siguiente carácter. Si es un punto y coma (;) la impresión se hace a continuación del último carácter impreso; si es un ***tab(n)*** o un ***tab*** tiene el efecto antes descrito; si se omite, la impresión comienza en una nueva línea.

El color, la fuente y el tamaño del texto se toman de las correspondientes propiedades del formulario o control ***picture***.

6.4.2 Dibujo de puntos: método Pset

El método ***pset*** sirve para dibujar puntos en un formulario o en un control ***picture***.
Su forma general es la siguiente:

```
object.PSet Step (x, y), color
```

donde:

object es opcional y representa el objeto (***form*** o ***picture***) en el que se va a dibujar el punto. Si se omite, el punto se dibuja en el formulario activo (el que tiene el *focus*).

Step es opcional. Si se introduce las coordenadas que le siguen son relativas respecto a las propiedades ***CurrentX*** y ***CurrentY***. Al dibujar un punto, estas propiedades se actualizan a las coordenadas de dicho punto.

(x, y) son las coordenadas absolutas o relativas del punto a dibujar (expresiones, variables o constantes *single*). Tanto las coordenadas como los paréntesis son obligatorios. Las unidades dependen de la propiedad ***ScaleMode*** del objeto en que se dibuja.

Color es opcional y es un nombre de color (***vbRed***, ***vbBlue***, etc.) o un ***long*** conteniendo el código de color hexadecimal (puede ser el valor de retorno de la función ***RGB***). Si se omite se utiliza la propiedad ***ForeColor*** del objeto en el que se dibuja.

El tamaño del punto viene determinado por la propiedad ***DrawWidth*** del objeto en que se dibuja. Si el tamaño es mayor que uno, el punto se dibuja centrado en las coordenadas suministradas a ***pset***. Si se desea eliminar un punto previamente dibujado es necesario volver a pintar ese punto con el color de fondo del objeto (*BackColor***)**.

6.4.3 Dibujo de líneas y rectángulos: método line

El método ***line*** dibuja líneas y -en ciertas condiciones- cajas rectangulares de lados horizontales y verticales. Su forma general es la siguiente:

```
object.Line Step (x1, y1) - Step (x2, y2), color, BF
```

donde ***object***, ***step*** y ***color*** tienen el mismo significado que en ***pset***, y

(x1, y1) son opcionales y son las coordenadas del punto inicial de la línea. Si se omiten la línea comienza en las coordenadas definidas por ***CurrentX*** y ***CurrentY***.

(x2, y2) son obligados y contienen las coordenadas del punto final de la línea.

B es un carácter opcional. Si se incluye se dibuja un rectángulo (*Box*) con los puntos dados como extremos de una de sus diagonales.

F es también un carácter opcional, que sólo se puede incluir si se ha incluido ***B***. Si se incluye, la caja rectangular se rellena (*Fill*) con el mismo color del contorno. Si se omite la caja se rellena con las propiedades ***FillColor*** y ***FillStyle*** del objeto en el que se dibuja.

Después de ejecutarse este método las propiedades ***CurrentX*** y ***CurrentY*** tienen el valor del punto final de la línea. Es necesario introducir el carácter (-), aunque se omita el primero de los puntos que definen la línea.

Las propiedades ***DrawWidth*** y ***DrawStyle*** determinan cómo se dibujan las líneas rectas o curvas en ***Visual Basic 6.0***.

Más en concreto, la propiedad ***DrawWidth*** determina el grosor en pixels, mientras que ***DrawStyle*** determina el tipo de línea.

La **Tabla 6.3** considera los posibles valores de la propiedad ***DrawStyle***.

Valor	Estilo de línea
0	continua (valor por defecto)
1	trazos (continua si w>1)
2	puntos (continua si w>1)
3	raya-pto (continua si w>1)
4	raya-pto-pto (continua si w>1)
5	transparente (continua si w>1)
6	continua interna

Los tipos de raya discontinua no permiten que el grosor sea mayor que 1 pixel. Si el grosor es superior, la línea se dibuja de modo continuo.

Ejemplo:

```
Line (0 ,0 )-(100 , 0)    ' Línea del punto (0,0) al (100,0)
Line -(100 , 100)          ' Línea del punto (100,0) al (100,100)
Line -Step (20 , 80)       ' Línea del punto (100,100) al (120,180)
Line (100,100)-(200 , 200), vbRed, BF ' Rectángulo rojo del punto
                                      ' (100,100) al (200,200)
```

6.4.4 Dibujo de circunferencias, arcos y elipses: método circle

Este método permite dibujar circunferencias, elipses y arcos. Su forma general es la siguiente:

```
object.Circle Step (x, y), radius, color, start, end, aspect
```

donde ***object***, ***step*** y ***color*** tienen el mismo significado que en ***pset*** y ***line***, y

(x, y) son obligatorias, y contienen las coordenadas del centro de la circunferencia.

Radius es obligatoria y define el radio de la circunferencia.

Start, end son opcionales, y permiten definir arcos por medio del ángulo inicial (***start***) y final (***end***). Los ángulos se miden siempre en radianes y en sentido contrario a las agujas del reloj. Sus valores deben estar entre -2π y 2π. En principio se dibuja solamente el arco, pero si uno o ambos valores son negativos se tratan como positivos, pero se dibuja una línea que une el centro de la circunferencia con el origen o el extremo del arco.

Aspect es también opcional y se utiliza para dibujar elipses. Es la relación entre el Diámetro vertical y el horizontal. El valor por defecto es 1.0, lo que corresponde a una circunferencia. Cuando ***aspect*** es distinto de 1.0, el parámetro ***radius*** define el mayor de los dos diámetros.

Sólo las figuras cerradas (no los arcos sin líneas que unan los extremos con el centro) pueden ser rellenadas con el color determinado por las propiedades ***FillColor*** y ***FillStyle*** del objeto en que se dibuja). El grosor y estilo de las líneas se determina con las propiedades ***DrawWidth*** y ***DrawStyle***. Después de ejecutarse este método las propiedades ***CurrentX*** y ***CurrentY*** tienen el valor del centro de la circunferencia. Si se omite algún argumento (excepto los que van al final), deben respetarse las comas de separación entre argumentos.

6.4.5 Otros métodos gráficos

Existen algunos otros métodos gráficos de interés. Por ejemplo, el método ***Cls*** cuya forma general es

```
object.Cls
```

borra del formulario o control ***picture*** todos los resultados de los métodos gráficos y del método ***print***, al mismo tiempo que pone las propiedades ***CurrentX*** y ***CurrentY*** a cero. No afecta a los gráficos introducidos en modo de diseño (por ejemplo con la propiedad ***picture***). Tampoco se borran con este método el texto y gráficos que se hayan creado con la propiedad ***AutoRedraw*** en ***true***, si dicha propiedad se pone a ***false*** antes de llamar al método ***Cls***. De esta forma se pueden realizar borrados selectivos.

El método ***Point*** devuelve, como entero *long*, el color (RGB) del punto especificado en un formulario o control ***PictureBox***. Su forma general es:

```
object.Point(x, y)
```

Si se desea, el entero *long* devuelto por ***point*** puede convertirse a la notación hexadecimal que se usa para los colores utilizando la función ***hex***.

6.5 SISTEMAS DE COORDENADAS

Un punto de particular importancia con ***Visual Basic 6.0*** es el que hace referencia a la posición y tamaño de los formularios y de los demás controles, así como a las unidades en que se expresan y determinan.

Valor	Unidades
0	definidas por el usuario
1	twips (1440 por pulgada)
2	puntos (72 por pulgada)
3	pixels
4	caracteres (120x240 twips)
5	pulgadas
6	milímetros
7	centímetros

Tabla 6.4. Valores de ***ScaleMode***.

Visual Basic 6.0 permite elegir entre distintas unidades, e incluso utilizar distintas unidades para distintos elementos de la aplicación. Las unidades se especifican con la propiedad ***ScaleMode***, que es propia de los formularios y controles ***picture***.

La **Tabla 6.4** especifica los posibles valores de esta propiedad. La unidad por defecto es el ***twip***, que es la vigésima parte del ***punto*** o ***pixel***.

En un formulario las propiedades relacionadas con la escala y las dimensiones, agrupadas de cuatro en cuatro, son las siguientes: (***top***, ***left***, ***height*** y ***width***) y (***scaleTop***, ***scaleLeft***, ***scaleHeight*** y ***scaleWidth***). Su significado de explica a continuación con ayuda de la **Figura 6.7.**

En esta figura se muestra la *pantalla* y un *formulario*. La posición y dimensiones del formulario vienen dadas por las propiedades (***top***, ***left***, ***height*** y ***width***).
Para un formulario, estas propiedades se definen ***siempre*** en ***twips***. Obsérvese que se miden a partir de la esquina superior izquierda.

Sin embargo, el formulario puede tener su propio sistema de coordenadas interno, definido por las propiedades (***scaleTop***, ***scaleLeft***, ***scaleHeight*** y ***scaleWidth***), para lo cual su propiedad ***ScaleMode*** debe estar puesta a cero. Las propiedades ***scaleLeft*** y ***scaleTop*** determinan las coordenadas de la esquina superior izquierda en el propio sistema de coordenadas, mientras que ***scaleWidth*** y ***scaleHeight*** determinan su anchura y altura en dichas coordenadas.

En realidad estas propiedades determinan indirectamente la posición del origen de coordenadas y la escala y orientación de los ejes. Si ***scaleHeight*** es positiva el eje de ordenadas va hacia abajo, mientras que si es negativa estará orientado hacia arriba.
El eje horizontal va hacia la derecha si ***scaleWidth*** es positiva, y hacia la izquierda si es negativa. Las cuatro propiedades (***scaleTop***, ***scaleLeft***, ***scaleHeight*** y ***scaleWidth***) se pueden establecer conjuntamente con el método ***scale***, como se verá en el siguiente apartado. Sólo los formularios y los controles ***pictureBox*** pueden tener las propiedades ***scaleTop***, ***scaleLeft***, ***scaleHeight*** y ***scaleWidth***.

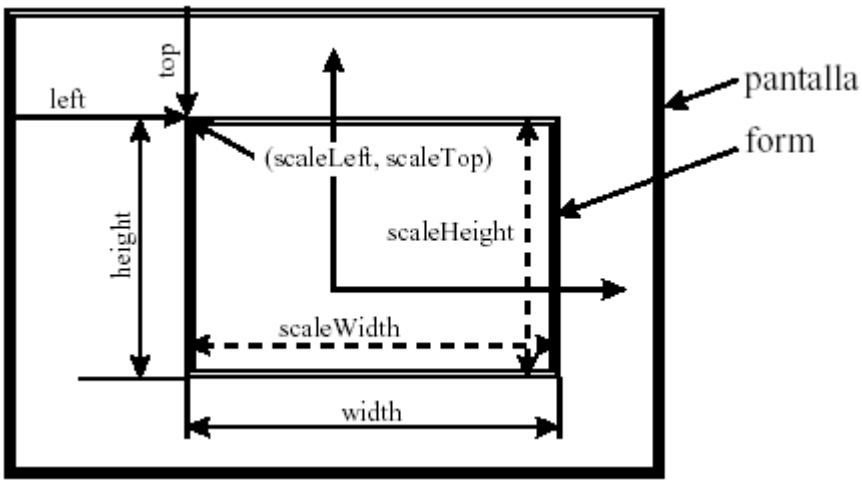


Figura 6.7. Posición y tamaño de una caja ***PictureBox***.

Si las propiedades (***top***, ***left***, ***height*** y ***width***) no se aplican a un formulario sino a un control, ya no es obligatorio medirlas en ***twips***, sino que se miden en las unidades determinadas por la propiedad ***scaleMode*** del ***formulario*** o ***pictureBox*** que las contiene. Cuando estas propiedades se utilizan sin anteponerles el nombre de un objeto se aplican al formulario activo. Para que se apliquen a un objeto cualquiera basta anteponerles el nombre del objeto separado por el operador punto (.).

6.5.1 Método Scale

El método ***Scale*** permite definir las cuatro propiedades (***scaleTop***, ***scaleLeft***, ***scaleHeight*** y ***scaleWidth***) de un formulario o ***pictureBox*** simultáneamente. Su forma general es:

```
object.Scale (x1, y1) - (x2, y2)
```

donde ***object*** es el nombre del control ***picture*** (si se omite, el método se aplica al formulario activo).
Las coordenadas ***(x1, y1)*** son las coordenadas del vértice superior izquierdo del formulario o ***pictureBox***, mientras que ***(x2, y2)*** corresponden al vértice inferior derecho.

Por ejemplo, el siguiente método:

```
pctCaja.Scale (-100, 100) - (100, -100)
```

establece unos ejes en el centro de la ***pictureBox***, con los sentidos ordinarios, que varían entre -100 y 100, tal como puede verse en la Figura 6.8. Este método equivale establecer las cuatro propiedades siguientes:

```
pctCaja.scaleTop = 100
```

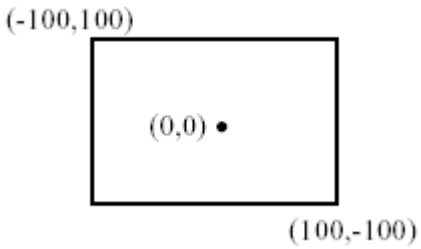


Figura 6.8. Método ***Scale***.

```
pctCaja.scaleLeft = -100
pctCaja.scaleHeight = -100
pctCaja.scaleWidth = 100
```

Continuará.....

Nota de Radacción: El lector puede descargar este capítulo y capítulos anteriores del curso desde la sección ***"Artículos Técnicos"*** en el sitio web de **EduDevices (www.edudevices.com.ar)**