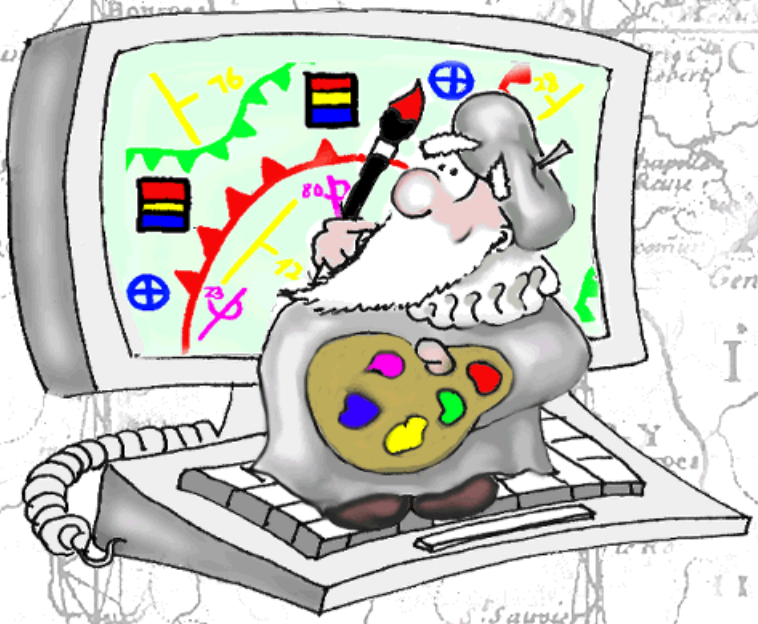
CARTOSCRIPTS

Tutoriales

Usando CartoScripts™



con

TNTmips®

TNTedit™

TNTview®

Antes del Tutorial

Algunos proyectos de mapeo pueden requerir el uso de símbolos especializados para líneas y puntos en objetos vector y CAD. El lenguaje de guiones cartográficos en TNTmips®, TNTview®, y TNTedit™ provee un conjunto completo y flexible de funciones de dibujo que le permiten diseñar símbolos de mapas personalizados para muchas aplicaciones. CartoScripts extiende las capacidades de creación de símbolos que se hallan en los editores estándares de estilos de puntos y líneas de los productos TNT. Los ejercicios en este folleto introducen las funciones de CartoScripts de uso más común, y proveen muchos ejemplos de CartoScripts para símbolos de puntos y líneas.

Requisitos Previos Este folleto asume que usted a completado los ejercicios en los siguientes folletos *Tutoriales*: *Desplegando Datos Geoespaciales, Navegando, Creando y Usando Estilos,* y *Construyendo y Usando Consultas.* Los ejercicios en estos folletos introducen las habilidades esenciales y técnicas básicas que no son cubiertas nuevamente aquí. Por favor consulte esos folletos y el manual de referencia de TNTmips para cualquier revisión que necesite.

Datos de Ejemplo Los ejercicios presentados en este folleto utilizan datos de ejemplo distribuidos con los productos TNT. Si no tiene acceso al CD de productos TNT, usted puede bajar los datos desde el sitio web de MicroImages. En particular este folleto usa los Archivos de Proyecto CARTOSMP y TOWNS en la colección de datos CARTOSCR. **Instale los archivos de ejemplo en su disco duro, de manera que los cambios puedan ser almacenados conforme trabaja con ellos.**

Mas Documentación Este folleto solo intenta ser una introducción al uso de CartoScripts para asignar estilos a los elementos vector o CAD. Para mayor información, consulte el volumen Display del Manual de referencia de TNTmips.

TNTmips y TNTlite™ TNTmips viene en dos versiones: la versión profesional y la versión libre TNTlite. Este folleto se refiere a las dos versiones como "TNTmips." Si usted no compra la versión profesional (la cual requiere de una llave de licencia de software), TNTmips opera en modo TNTlite, el cual limita el tamaño de sus materiales de proyecto y activa el compartir de datos únicamente con otras copias de TNTlite.

Todos los ejercicios pueden completarse en TNTlite utilizando los geodatos de ejemplo proporcionados.

Randall B. Smith, PhD, 25 August 2000

Sin una copia a color de este folleto podría ser difícil identificar algunos puntos importantes en algunas ilustraciones. Usted puede imprimir o leer este folleto a color en el sitio Web de MicroImages. Este sitio Web es también su fuente de nuevos Tutoriales sobre otros temas. Usted puede descargar una guía de instalación, datos de ejemplo y la última versión de TNTlite.

http://www.microimages.com

Bienvenido a Usando CartoScripts

Los editores de estilos en TNTmips y TNTview le permiten seleccionar, modificar, combinar, o crear una amplia variedad de símbolos estándar de puntos y líneas, tal como se describe en el *Tutorial: Creando y Usando Estilos*. Para aquellas instancias en las que los editores estándar de estilos no pueden proporcionar el símbolo apropiado, usted puede usar CartoScripts™ para diseñar símbolos personalizados de mapas para puntos y líneas.

Los CartoScripts son guiones de estilo que utilizan funciones especiales en el lenguaje de consulta de TNTmips. Las funciones CartoScripts le permiten dibujar y navegar a lo largo de los elementos lineales y dibujar nuevas líneas y formas para constituir los símbolos. Los símbolos pueden repetirse a lo largo de los elementos lineales o dibujarse uno a uno para los elementos punto. Usted puede añadir etiquetas a los símbolos usando textos de las tablas de las bases de datos asociadas, así como optimizar la ubicación de etiquetas dentro de un mismo nivel de dibujo. También puede estructurar un guión para usar los atributos del elemento para variar el estilo del símbolo.

Los ejercicios en este folleto introducen y explican las funciones CartoScript más comúnmente utilizadas y muestra como puede estructurar guiones para producir varios efectos para símbolos de puntos y líneas. Los CartoScripts están sujetos a las mismas reglas de sintaxis que las consultas estándar de bases de datos y guiones SML. Para una revisión de la sintaxis básica de consultas, revise el *Tutorial: Construyendo y Usando Consultas.*

Los mapas geológicos son uno de los ejemplos de mapas que requieren de símbolos especializados de puntos y líneas que pueden ser dibujados usando CartoScripts. Símbolos de puntos se usan para indicar la orientación de estructuras de afloramientos a escala, mientras que símbolos especiales de líneas se usan para representar detalles de mapeo a escala. Varios de los ejercicios en este folleto usan ejemplos geológicos para ilustrar los elementos de la estructura de los CartoScripts.

PASOS

- ☑ asegúrese de que los archivos de ejemplo mencionados en la página 2 han sido copiados a su disco duro
- ☑ inicie TNTmips
- ☑ escoja Display / Spatial Data del menú principal

Los ejercicios en las páginas 4-19 ilustran el uso de funciones CartoScripts para crear símbolos de puntos. Las páginas 4-8 introducen las funciones básicas usadas para dibujar líneas y formas geométricas simples. Las páginas 9-15 le conducen a través de añadir etiquetas de textos desde campos de bases de datos, dibujando símbolos más complejos y variando la orientación de los símbolos por atributos. La ubicación optimizada de etiquetas para símbolos de puntos es cubierta en las páginas 16-19.

Los ejercicios de las páginas 20-33 muestran como usar CartoScripts para crear símbolos de líneas. La navegación básica en líneas y las funciones de dibujo son introducidas en las páginas 20-22. Estructuras de guiones para crear símbolos repetidos se explican en las páginas 23-30. Las páginas 32-33 introducen la ubicación de etiquetas de texto para líneas, y las páginas 34-35 proveen de una lista completa de las funciones CartoScripts disponibles.

Usted puede bajar ejemplos adicionales de CartoScripts para símbolos geológicos de puntos y líneas del sitio Web de MicroImages:

www.microimages.com / freestuf / cartoscripts

Dibujar Símbolos de Puntos

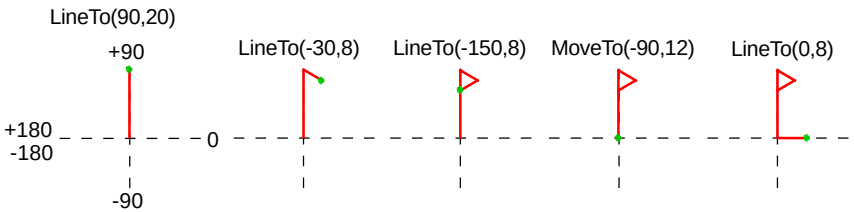
PASOS

- ☑ clic el icono del botón New 2D Group en la barra de herramientas Display 
- ☑ clic en el icono del botón Add Vector en la ventana Group Controls y escoja Add Vector Layer 
- ☑ navegue al Archivo de proyecto CARTOSMP y seleccione el objeto vector SAMPLES
- ☑ clic en la tarjeta Points en la ventana Vector Object Display Controls
- ☑ seleccione By Script de la opción Style del menú y clic [Specify...]
- ☑ escoja Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccione el objeto FLAGQRY del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

El CartoScript que usted abre en este ejercicio (se muestra en la caja abajo) dibuja el símbolo de bandera que se muestra a la derecha. La primera línea en el guión es una línea de comentario con el nombre del guión (recuerde que los comentarios son precedidos por el carácter "#"). La función LineStyleSetColor( ) en la línea 2 fija los valores de Red, Green, y Blue (entre 0 y 255) que determinan el color de las líneas dibujadas por las funciones de dibujo que siguen (en este caso, líneas rojas). La función en la línea 3 fija el ancho de línea para las funciones de dibujo (mas acerca de esto abajo).

Las líneas restantes del guión en realidad dibujan el símbolo. La función LineStyleLineTo( ) dibuja una línea hasta el punto especificado por una dirección (primer parámetro numérico) y distancia (segundo parámetro). La función LineStyleMoveTo( ) usa la misma secuencia de parámetros para mover la ubicación del "lápiz" sin dibujar. Las dos funciones referencian a un sistema local de coordenadas, centrado en el elemento punto actual (el guión es leído y evaluado una vez por cada elemento punto en el objeto). Las direcciones para estas funciones son especificadas por ángulos (0 a 180 y 0 a –180) relativos al eje ***x*** positivo del sistema de coordenadas del objeto. Las coordenadas del objeto también proporcionan las unidades por defecto para los parámetros de distancia en estas funciones, así como para el parámetro de ancho en la función LineStyleSetLineWidth( ). La secuencia de las acciones e dibujo se ilustran abajo, con el punto indicando la posición del lápiz al final de cada acción.

```
1 ##  FlagQry
2 LineStyleSetCol-
3 or(225,0,0)
4 LineStyleSetLineWidth(1)
5 LineStyleLineTo(90,20)
6 LineStyleLineTo(-30,8)
7 LineStyleLineTo(-150,8)
8 LineStyleMoveTo(-90,12)
```



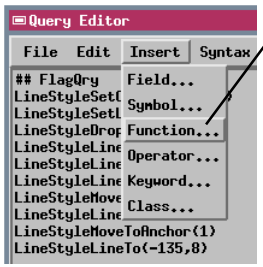
Usando Anclas

En la secuencia de movimientos de dibujo en el guión de la bandera del ejercicio previo, la posición del lápiz regresa al origen del sistema de coordenadas local, antes de dibujar la línea final en la base de la bandera. La geometría simple del símbolo de la bandera hace relativamente fácil calcular el ángulo y distancia para la función LineStyleMoveTo( ) de la línea 7 que mueve el lápiz al origen. Usted puede evitar la necesidad de tales cálculos usando **anclas:** posiciones que registra para uso posterior en un conjunto de acciones de dibujo. La función LineStyleDropAnchor( ) fija una posición de ancla y le asigna el número que usted ingresa como el parámetro numérico para la función. La función LineStyleMoveToAnchor( ) mueve el lápiz a la posición del ancla especificada. Existe también la función LineStyleLineToAnchor( ) que dibuja una línea desde la posición actual del lápiz a la posición del ancla especificada. Usted puede establecer múltiples puntos de anclaje para ayudar en el dibujo de símbolos complejos, y usarlos en cualquier orden. En el guión en esta página, ubicamos un ancla en el origen antes de empezar a dibujar, y regresamos a la posición del ancla dos veces más para dibujar líneas desde la base de la bandera.

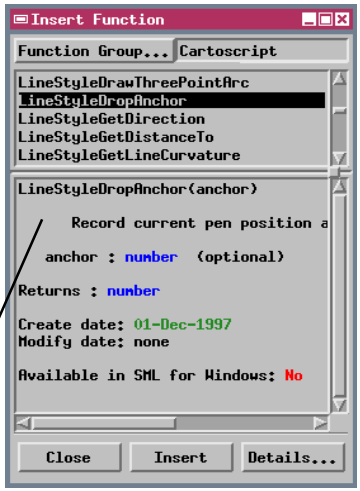
PASOS

- ☑ clic el icono del botón Vector en la fila de iconos Layer para abrir la ventana Vector Object Display Controls
- ☑ reabra la ventana Query Editor para estilos de puntos
- ☑ elimine la declaración con la función LineStyleMoveTo( ) e inserte las declaraciones que se muestran abajo en texto bold
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

```
LineStyleSetCol-
or(225,0,0)
LineStyleSetLineWidth(1)
LineStyleDropAnchor(1)
LineStyleLineTo(90,20)
LineStyleLineTo(-30,8)
LineStyleLineTo(-150,8)
LineStyleMoveToAnchor(1)
LineStyleLineTo(0,8)
LineStyleMoveToAnchor(1)
```



Usted puede manualmente ingresar nombres de funciones, de campos de bases de datos, de símbolos, o de operadores, o utilizar las opciones en el menú Insert para insertar los ítem requeridos en el guión.



Escoja CartoScript del menú Function Group para mirar la lista de funciones únicamente con las funciones de Estilo CartoScript

Usando Formas Geométricas Incorporadas

PASOS

- ☑ abrir nuevamente la ventana Query Editor para estilos de puntos
- ☑ editar el guión para duplicar el texto indicado abajo; la nueva declaración está en tipo negrilla

Varias funciones que dibujan formas geométricas simples, se hallan incluidas en el conjunto de funciones CartoScript. Las funciones LineStyleDrawRectangle( ) y LineStyleDrawCircle( ) dibujan sus respectivas formas centradas en la posición actual del lápiz, la cual es dejada sin cambios. El primer guión de este ejercicio dibuja un rectángulo relleno en la base del símbolo de la bandera. Cuatro parámetros numéricos son usados en esta instancia para controlar la función del rectángulo: *width*, *height*, *angle*, y *dofill*. Los dos primeros parámetros son necesarios, y especifican el ancho y alto del rectángulo. El tercer parámetro (opcional) especifica el ángulo de rotación, el cual en este ejemplo es igual a cero. El último parámetro (también opcional) determina si la forma es rellena con el color actual de la línea (1), o se deja sin relleno (0).

```
LineStyleSetCol-
or(225,0,0)
LineStyleSetLineWidth(1)
LineStyleDropAnchor(1)
LineStyleLineTo(90,20)
LineStyleLineTo(-30,8)
LineStyleLineTo(-150,8)
LineStyleMoveToAnchor(1)
```

- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

El segundo ejemplo en este ejercicio dibuja una elipse sin relleno en la base del símbolo de la bandera. La función LineStyleDrawEllipse( ) tiene hasta 7 parámetros: *angle*, *distance*, *radius_x*, *radius_y*, *rotangle*, *isAngleAbs*, y *dofill*; los cuatro primeros son necesarios. Los parámetros iniciales *angle* y *distance* le permiten automáticamente mover el lápiz a una nueva posición antes de dibujar. En este ejemplo, los dos son fijados a cero, dejando la elipse centrada en la base de la bandera. El lápiz regresa al centro de la elipse después de dibujarla. Las dimensiones de la elipse son inicialmente fijadas paralelas a los ejes de coordenadas X y Y por los dos parámetros de radio (en este ejemplo 8 y 3 unidades, respectivamente). El valor de 30 para el parámetro *angle* rota 30 grados a la elipse en sentido anti-horario. El parámetro *isAngleAbs* determina si la elipse es dibujada y rotada relativo al sistema local de coordenadas (0), o relativa a las coordenadas globales del objeto (1). Esta distinción no existe para datos de puntos, pero se hace importante cuando se asigna estilos de líneas, tal como lo veremos más tarde.

- ☑ abrir nuevamente la ventana Query Editor
- ☑ reemplazar la última línea en el guión con la declaración mostrada abajo

```
LineStyleDrawEl-
```

- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

Registrando y Dibujando Polígonos

La asignación de estilos para los extremos de líneas se hace por medio de la función LineStyleSetCapJoinType( ), la cual tiene los parámetros *capstype* y *jointype*. El parámetro *capstype* dibuja extremos de líneas cuadrados cuando está fijado a 1, o redondeados cuando está fijo en 0. El parámetro *jointype* usa los mismos valores para determinar el estilo de los extremos de segmentos del contorno de un polígono o polilínea. El valor por defecto para ambos parámetros es 0, de forma que extremos redondeados son dibujados si usted no incluye esta declaración en un guión.

PASOS

- ☑ abrir nuevamente la ventana Query Editor para estilos de puntos
- ☑ editar el guión para duplicar el texto indicado abajo, añadir la declaración mostrada en tipo negrilla

```
LineStyleSetColor(225,0,0)
LineStyleSetLineWidth(1)
LineStyleSetCapJoin-
Type(1,1)
LineStyleLineTo(90,20)
LineStyleLineTo(-30,8)
```

El guión en la segunda parte de este ejercicio dibuja un símbolo de bandera con color sólido, dibujando el elemento triangular de la bandera como un polígono relleno. Usted puede dibujar formas de polígonos simples o complejos usando funciones que ejecuten los siguientes pasos: 1) iniciar registrando las ubicaciones de vértices; 2) mover hacia o dibujar líneas a cada uno de los vértices en su turno; 3) conectar los vértices para dibujar el polígono. La función LineStyleRecordPolygon( ) tiene un solo parámetro *start_stop*; un valor de 1 inicia registrando las ubicaciones de los vértices especificadas por los movimientos del lápiz en las declaraciones subsiguientes. La función LineStyleDrawPolygon( ) forma el polígono usando los vértices registrados, y tiene un solo parámetro *dofill* (fijado a 1 en este ejemplo para rellenar el triángulo). La función LineStyleDrawPolygon( ) también detiene el registro de las ubicaciones de los vértices, de forma que no hay necesidad de detener explícitamente el registro con la declaración LineStyleRecordPolygon(0) a continuación de la lista de movimiento de vértices. Usted también puede usar la misma estructura para registrar ubicaciones de vértices para conectarlos como segmentos de una línea simple por medio de la función LineStyleDrawPolyline( ).

- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls
- ☑ abrir nuevamente la ventana Query Editor
- ☑ cambiar los valores para los parámetros de extremo y unión como se muestra, y añadir las declaraciones que se muestran en negrilla
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

```
LineStyleSetCol-
or(225,0,0)
LineStyleSetLineWidth(1)
LineStyleSetCapJoin-
Type(0,0)
LineStyleLineTo(90,20)
LineStyleDropAnchor(2)
LineStyleRecordPolygon(1)
LineStyleLineTo(-30,8)
LineStyleLineTo(-150,8)
```

- ☑ seleccionar Close del menú Group cuando haya completado este ejercicio

Usando Formas 3D

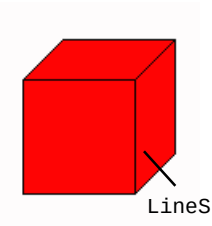
PASOS

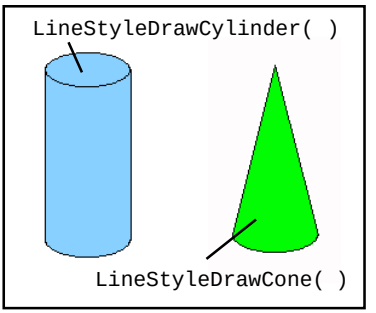
- ☑ clic el icono del botón Open en la barra de herramientas Display Spatial Data y escoja del menú Open Group
- ☑ seleccione el objeto CUBEGROUP del Archivo de Proyecto CARTOSMP
- ☑ clic el icono del botón Vector en la fila de iconos de Layer en el layer SAMPLES para abrir la ventana Vector Object Display Controls
- ☑ abrir la ventana Query Editor para estilos de puntos y examine el guión script CUBEQRY1
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

Funciones CartoScripts también están disponibles para dibujar representaciones perspectivas de formas simples tridimensionales: rectángulos sólidos, cilindros verticales, y conos verticales. Las líneas de los bordes en cada símbolo son dibujadas usando el color especificado por la función LineStyleSetColor( ). Tres parámetros de color para cada función fijan los valores de red, green, y blue para el color de relleno. El símbolo de un cubo es dibujado con la base de la cara frontal centrada en la posición actual del lápiz. Los símbolos de cilindro y cono son dibujados de manera que el centro de la base elíptica coincida con la posición actual del lápiz.

Los parámetros *width*, *depth*, y *height* para la función LineStyleDrawCube( ) especifican las longitudes de los correspondientes bordes del sólido rectangular. Si desea que el símbolo aparezca en perspectiva como un cubo verdadero, como en este ejercicio, los valores de *width* y *height* deben ser iguales y el valor de *depth* debería ser alrededor de la mitad de la longitud de las otras dimensiones.

Usted puede definir en un guión variables string o numéricas para su uso posterior como parámetros de la función. Definiendo variables (con comentarios) al inicio del guión, hace más fácil para encontrar y editar los valores de parámetros necesarios cuando esta reutilizando y modificando un guión






```
# CubeQry1
# Set dimensions of cube symbol
width = 15
depth = 0.5 * width
height = width

# Set color for cube faces
red = 255;  green= 0;  blue = 0

# Draw cube symbol
LineStyleSetColor(0,0,0)    # line color for edges
LineStyleSetLineWidth(0)
LineStyleDrawCube(width,depth,height,red,green,blue)
```

Etiquetas de Texto desde Campos de Base de Datos

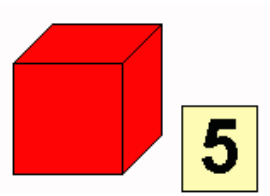
El guión en este ejercicio añade una etiqueta de texto enmarcada al símbolo de cubo. Las declaraciones adicionales necesarias para formatear y dibujar la etiqueta se muestran abajo. La etiqueta es un número de muestra almacenado como un valor numérico en un campo de la base de datos. En orden a usar este como una etiqueta, el número primero debe ser convertido a un texto string y asignado a una variable string usando la función sprintf( ). El primer parámetro en esta función es un string (entre comillas) con información del formato; "%d" indica un valor entero. El segundo parámetro en este caso es la ubicación del valor numérico, especificado en la forma TableName.FieldName.

Las etiquetas de texto son dibujadas con la esquina inferior izquierda correspondiendo a la posición del lápiz.. La orientación de una etiqueta es fijada por el parámetro *angle*. El parámetro *border* especifica el ancho del borde entre la etiqueta de texto y su recuadro circundante. El último parámetro mostrado aquí, *isAbs*(0), indica el marco de referencia del parámetro *angle*.

PASOS

- ☑ clic el icono del botón Vector para el layer SAMPLES para volver a abrir la ventana Vector Object Display Controls
- ☑ abrir la ventana Query Editor para estilos de puntos
- ☑ escoja Open / RVC Object del menú File
- ☑ seleccione el objeto CUBEQRY2 en el Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

```
# Read sample number from database field and convert
# to text string for use as a label
label$ = sprintf("%d",Samples.Number)

# String variable for label text font
font$ = "ARIALBD.TTF"

# Define color variables for text
tred = 0;      tgreen= 0;     tblue = 0

# Define fill color variables for text box
fillred = 255;     fillgreen = 255;   fillblue = 170

# Define height, angle, and border width of text box
t_height = 10;     angle = 0; border = 2

# Set color and font for text label
LineStyleSetTextColor(tred,tgreen,tblue,fillred,fillgreen,fillblue)
LineStyleSetFont(font$)

# Move pen to right of symbol and draw label
LineStyleMoveTo(0, width * 1.2)
LineStyleDrawTextBox(label$,t_height,angle,border,0)
```



Fijando las Opciones de Tipo de Coordenada

PASOS

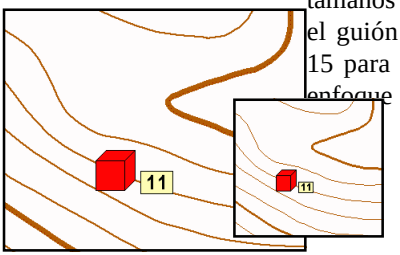
- ☑ con acercamientos y alejamientos note el efecto en los tamaños relativos de los símbolos puntos y curvas
- ☑ reabrir la ventana Vector Object Display Controls para el layer SAMPLES y la ventana Query Editor para estilos de puntos
- ☑ escoja Open / RVC Object del menú File
- ☑ seleccione el objeto CUBEQRY3 desde el Archivo de Proyecto CARTOSMP

Las unidades por defecto para los valores de tamaño y distancia que usa en CartoScripts son coordenadas de objeto internas (metros para los objetos que usó en estos ejercicios). Así conforme usted cambia los niveles de ampliación, el tamaño de los símbolos en la pantalla cambia conforme al cambio de la escala de despliegue, manteniendo un tamaño constante en coordenadas del objeto. El guión en este ejercicio añade las declaraciones mostradas abajo en negrita al guión previo CubeQry2 (conjuntamente con cambios en el símbolo y valores de tamaño de la etiqueta). Un valor de parámetro de 1 para la función LineStyleSetCoordType( ) cambia las unidades del tamaño y distancia a milímetros a la escala actual de despliegue o impresión. Cuando cambia los niveles de acercamiento o escala de impresión, los símbolos mantienen un tamaño de despliegue constante en milímetros. Estas definiciones trabajan mejor con formatos designados solo para despliegue (Un valor de 0 para este parámetro es equivalente a la condición por defecto).

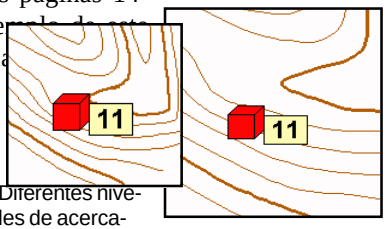
```
# CubeQry3
# Set dimensions of cube
symbol
LineStyleSetCoordType(1)
width = 4
depth = 0.5 * width
```

- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls
- ☑ con acercamientos y alejamientos note el efecto en los tamaños relativos de los símbolos y curvas de nivel
- ☑ seleccione Close del menú Group cuando haya completado este ejercicio

Si está usando CartoScripts para dibujar elementos de mapa designados para imprimirse a una escala particular (tal como 1:24.000), usted puede usar las coordenadas fijadas por defecto, para mantener los tamaños relativos de diferentes elementos a diferentes niveles de ampliación para el despliegue en pantalla, pero usar la escala del mapa para calcular el tamaño de elemento requerido para producir los tamaños deseados en el mapa final impreso. Mire el guión en las páginas 14-15 para un ejemplo de este enfoque de esca



Niveles diferentes de acercamiento con escalamiento por defecto a las coordenadas de objeto. El símbolo punto tiene un tamaño constante en coordenadas del objeto.



Diferentes niveles de acercamiento, con escalamiento a milímetros. El símbolo punto mantiene un tamaño constante en coordenadas de pantalla (o impresión).

Gráfico de Barras: Cilindro 3D

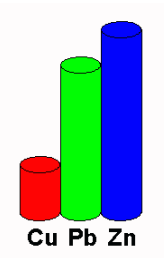
Los símbolos 3D pueden ser combinados conjuntamente para formar gráficos de barras 3D, con el alto de cada barra determinado por el valor de un campo de la base de datos. Usted puede cambiar la pers-pectiva de los símbolos del cilindro y cono, variando las longitudes relativas de los ejes mayor y menor de la base del cilindro y cono.

La función LineStyleTextNextPosition( ) es usada en este guión para aproximadamente centrar la etiqueta del elemento "Cu" bajo su cilindro. Los cuatro primeros parámetros de esta función especifican el string, su alto, ángulo y la referencia de las coordenadas locales o absolutas. Los últimos tres parámetros *next_x*, *next_y*, y *length*, son variables que la función crea para contener la coordenada_x, coordenada_y, y longitud de la etiqueta de la cadena de caracteres. Estos valores pueden ser usados en declaraciones subsiguientes para guiar el posicionamiento

PASOS

- ☑ clic el botón del icono New 2D Group en la barra de herramientas Display
- ☑ clic en el botón del icono Add Vector y escoja Add Vector Layer
- ☑ seleccione el objeto GeochemVec del Archivo de Proyecto CARTOSMP
- ☑ fije el estilo de punto a By Script y abra la ventana Query Editor
- ☑ escoja Open / RVC Object del menú File
- ☑ seleccione el objeto CylinGraph en el Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

```
# Read values for element abundances from
# database and assign to variables to use
# for cylinder heights
cuVal = Geochemistry.Cu
pbVal = Geochemistry.Pb
znVal = Geochemistry.Zn
# Set edge line color and cylinder dimensions
LineStyleSetColor(0,0,0)
longAxis = 120;    shortAxis = 50
# Set text color and font
LineStyleSetTextColor(0,0,0)
LineStyleSetFont("ARIALBD.TTF")
# Draw three cylinders side by side
LineStyleDropAnchor(1)
LineStyleDrawCylinder(longAxis,shortAxis,cuVal,255,0,0)
LineStyleMoveTo(0,longAxis)   # move right by width of cylinder
LineStyleDrawCylinder(longAxis,shortAxis,pbVal,0,255,0)
LineStyleMoveTo(0,longAxis)
LineStyleDrawCylinder(longAxis,shortAxis,znVal,0,0,255)
# Draw label centered below each cylinder
LineStyleMoveToAnchor(1)  # move to base of first cylinder
LineStyleMoveTo(-90,100)  # move down to make room for label
LineStyleTextNextPosition("Cu",70,0,0,next_x,next_y,length)
LineStyleMoveTo(180,length * 0.4) # move label point to left
LineStyleDrawText("Cu",70,0,0)    # to center first label
LineStyleMoveTo(0,longAxis)   # move right by width of cylinder
LineStyleDrawText("Pb",70,0,0)
LineStyleMoveTo(0,longAxis)
LineStyleDrawText("Zn",70,0,0)
```



- ☑ remueva el layer GEOCHEMVEC cuando haya completado este ejercicio

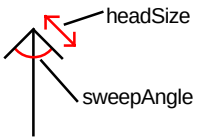
Orientando Símbolos por Atributo

PASOS

- ☑ clic en el botón del  icono Add Vector y escoja Add Vector Layer
- ☑ seleccione el objeto ArrowPts del Archivo de Proyecto CARTOSMP
- ☑ fije el estilo de puntos a By Script y abra la ventana Query Editor
- ☑ escoja Open / RVC Object desde el menú File
- ☑ seleccione el objeto ArrowQry1 en el Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

Usted puede usar un guión para dibujar símbolos que varían la orientación, dependiendo de un valor de dirección leído desde un campo de la base de datos. En ésta instancia la dirección de la flecha está en forma de acimut (ángulo de 0º a 360º medido en sentido horario desde el Norte), y debe ser convertida al sistema interno de coordenadas usado por las funciones de dibujo. Este guión también convierte todos los valores de direcciones negativas a los correspondientes valores positivos, sin embargo esta conversión no es necesaria.

La función LineStyleDrawArrow( ) dibuja puntas de flecha limitadas por líneas rectas cuya longitud es definida por la variable *headSize* en este guión. El ángulo entre esta líneas de borde es fijado por el parámetro *sweepAngle*. Si el parámetro *dofill* es fijado a 1, la cabeza es rellenada para formar un triángulo sólido. La función actualiza la posición del lápiz al vértice de la punta de la flecha. Las puntas de flecha no se representan bien con una línea de ancho mayor a 0, por lo tanto este guión dibuja la flecha con una línea de ancho 0, luego redibuja el cuerpo de la flecha con una línea más ancha.




```
# Read azimuth from database table
azim = Direct.Azimuth
# Convert azimuth to internal coordinate system
direction = -(azim - 90)
    if (direction < 0) then  direction = direction + 360
# Set color values for symbol
red = 0;    green = 0;  blue = 0
LineStyleSetColor(red,green,blue)
# Set dimensions for arrow
arrowLength = 30
headSize = 0.4 * arrowLength
sweepAngle = 40;   dofill = 1
# Draw arrow with zero line width, tip of stem at point
LineStyleDropAnchor(0)     # anchor at point
LineStyleSetLineWidth(0)
LineStyleDrawArrow(direction,arrowLength,headSize,sweepAngle,dofill)
LineStyleDropAnchor(1)     # anchor at tip of arrow
# Redraw arrow stem with wider line
stemLength = arrowLength - headSize * cosd(sweepAngle)
LineStyleMoveToAnchor(0)
LineStyleSetLineWidth(1.5)
LineStyleLineTo(direction,stemLength)
LineStyleMoveToAnchor(1)  # pen to arrow tip in prep for label
```

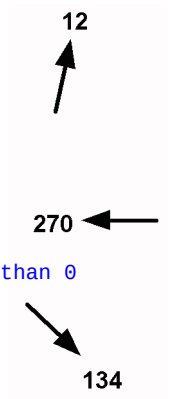
Calculando las Posiciones de Etiquetas

El guión en este ejercicio añade una etiqueta con el acimut de cada símbolo de flecha del ejercicio previo. Las declaraciones adicionales necesarias para formatear y dibujar las etiquetas se muestra abajo. La parte complicada es variar la posición de la etiqueta basada en la orientación de la flecha para evitar superposiciones entre la flecha y su etiqueta. El guión calcula dos cambios de sitio del lápiz que son aplicados antes de que la etiqueta sea dibujada, como se explica en los comentarios. Cada uno es calculado como una función del ángulo de dirección y el alto y largo de la cadena de caracteres de la etiqueta. El segundo cambio de sitio debe ser calculado separadamente para cada cuadrante, teniendo cuidado que la distancia resultante sea un valor positivo (los valores de distancia para parámetros de funciones de dibujo deben ser positivos; un valor negativo es interpretado como igual a 0).

PASOS

- ☑ reabrir la ventana Query Editor para estilos de puntos
- ☑ escoger Open / RVC Object del menú File
- ☑ seleccionar el objeto ArrowQry2 en el Archivo de Proyecto cartosmp
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls
- ☑ remover el layer ArrowPts cuando haya completado este ejercicio

```
# Convert azimuth value to text string for use as a label
label$ = sprintf("%d",azim)
# Find length of label text for label positioning
height = 10
LineStyleTextNextPosition(label$,height,0,0,next_x,next_y,length)
# Set font name and color
LineStyleSetFont("ARIALBD.TTF")
LineStyleSetTextColor(red,green,blue)
# Compute label shift perpendicular to arrow to center label
shift1 = 0.5 * (height*cosd(direction) - length*sind(direction))
# Compute label shift parallel to arrow to avoid overwriting arrow
offset = arrowLength * 0.1
if (direction >= 0 and direction < 90) then
    shift2 = offset
if (direction >= 90 and direction < 180) then
    shift2 = offset - length * cosd(direction)
if (direction >= 180 and direction < 270) then
    shift2 = offset - length * cosd(direction)
        - height * sind(direction)
if (direction >= 270 and direction <= 360) then
    shift2 = offset - height * sind(direction)
# shift pen position and draw label
if (shift1 < 0 ) {   # MoveTo distances can't be less than 0
    shift1 = abs(shift1)    # absolute value
    LineStyleMoveTo(direction + 90,shift1) }
    else {
        LineStyleMoveTo(direction - 90,shift1) }
LineStyleMoveTo(direction,shift2)
LineStyleDrawText(label$,height,0,0)
```



Dibujando Símbolos de Buzamiento y Pendiente

PASOS

- ☑ clic en el botón del  icono Add Vector y escoja Add Vector Layer
- ☑ seleccione el objeto BEDDING del Archivo de Proyecto CARTOSMP
- ☑ fije el estilo de punto a By Script y abra la ventana Query Editor
- ☑ escoja Open / RVC Object desde el menú File
- ☑ seleccione el objeto BEDDINGQRY en el Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls
- ☑ seleccione Close del menú Group cuando haya completado este ejercicio

Los mapas geológicos usan símbolos especiales de puntos para indicar la orientación de estructuras planas o lineales en afloramientos rocosos. El guión en este ejercicio dibuja símbolos estándar mostrando el buzamiento y pendiente de las capas (estratificación) en rocas sedimentarias, como se muestra abajo. El guión usa muchas de las funciones y conceptos introducidos previamente. La pendiente y el buzamiento son leídos desde una tabla de la base de datos asociada (El ángulo de buzamiento debe ser especificado como un acimut usando la denominada "regla de la mano derecha": la línea de buzamiento apunta hacia el acimut para el cual la dirección de la pendiente está hacia la derecha). El símbolo es orientado de forma que la línea larga (buzamiento) sea paralela a la dirección del buzamiento, y el símbolo es etiquetado con el valor del ángulo de la pendiente. Símbolos especiales son dibujados para estratos horizontales y verticales (valores especiales del ángulo de la pendiente), y para estratos inclinados (indicados por un campo lógico en la base de datos). La segunda parte del guión, la cual etiqueta los símbolos con el valor de la pendiente, es similar al guión de la página previa, y no se muestra aquí.

Inclinada

Volcada


Horizontal


Vertical

```
# BeddingQry
# Read strike azimuth and dip value from table.field
azStrike = Bedding.Strike;    dip1 = Bedding.Dip
# Check logical field for overturned bedding.  Variable is set
# to 1 if Yes, 0 if No
overturned = Bedding.Overturned
# Variables define the color of the symbols and label
red = 0;       green = 0;     blue = 0
# This variable defines the denominator of the intended map scale.
scale = 5000
# These variables define the dimensions and line widths of the
# symbol.  strikeLengthMap is the desired length of the symbol
# strike line in mm, assuming vector coordinates are in meters.
# lineWidthMap is the desired line width in mm.
strikeLengthMap = 6;      lineWidthMap = 0.3
strikeLength = strikeLengthMap * scale / 1000
halfLength = 0.5 * strikeLength
tickLength = halfLength / 3
doubTick = tickLength * 2
lineWidth = lineWidthMap * scale / 1000
```

Guión de Buzamiento y Pendiente (continuación)

```
###### Process
# Convert strike azimuth to internal coordinate system.
direction = -(azStrike - 90)
if (direction < 0)
    direction = direction + 360;
oppStrike = direction -180
dipDir = direction - 90
oppDip = dipDir - 180

# Set line color, width, and end type
LineStyleSetColor(red, green, blue)   # set symbol color
LineStyleSetLineWidth(lineWidth)
LineStyleSetCapJoinType(1,1)  # square ends of lines

############ Draw symbol
# Special symbol for horizontal bedding (cross in circle)
if (dip1 == 0){
    LineStyleDropAnchor(0)
    LineStyleDrawCircle(halfLength)
    LineStyleMoveTo(90, halfLength)
    LineStyleLineTo(-90, strikeLength)
    LineStyleMoveToAnchor(0)
    LineStyleMoveTo(0, halfLength)
    LineStyleLineTo(180, strikeLength)
    }
else {
    # For nonzero dip, draw strike line with center at point
    LineStyleDropAnchor(0)
    LineStyleMoveTo(direction, halfLength)
    LineStyleLineTo(oppStrike, strikeLength)
    LineStyleMoveToAnchor(0)

    # Draw appropriate symbol for dip direction
    if (dip1 == 90) {             # crossbar for vertical bed
        LineStyleMoveTo(dipDir, tickLength)
        LineStyleLineTo(oppDip, doubTick)
        }
    else {
        if (overturned == 1) {    # dip symbol for overturned beds
            LineStyleDrawArc(0, 0, tickLength, tickLength,
                             direction, -180, 0)
            LineStyleMoveTo(direction, tickLength)
            LineStyleLineTo(oppDip, doubTick)
            }
        else {                    # dip direction tick mark
            LineStyleLineTo(dipDir, tickLength)
            }
        }
    }
```

Optimización de Etiquetas

STEPS

- ☑ click the Open icon button on the Display Spatial Data toolbar and choose Open Group from the menu 
- ☑ select the OPTGROUP object from the TOWNS Project File
- ☑ click the Vector icon button for the TOWNS layer to open the Vector Object Display Controls window 
- ☑ open the Query Editor window for point styles and examine the script OPTQRY1
- ☑ click [OK] in the Query Editor window and again in the Vector Object Display Controls window

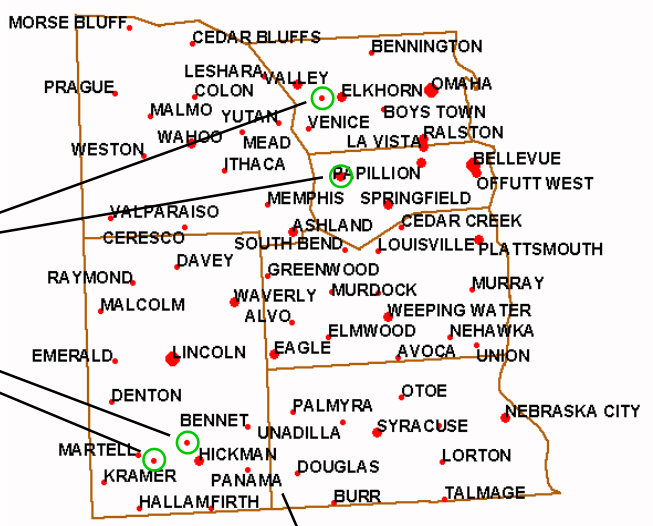
Optimización de Etiqueta es un procedimiento para encontrar automáticamente el conjunto óptimo de posiciones para las etiquetas de símbolos de puntos generadas desde un campo de una base de datos. El objetivo es evitar superposiciones de una etiqueta con otra de un símbolo cercano. Las etiquetas individuales pueden ser movidas o borradas para evitar estas colisiones. El optimizador puede automáticamente ubicar una etiqueta en una serie de posiciones diferentes alrededor del símbolo. La posición preferida ubica la esqui-na inferior izquierda de la etique-ta sobre el punto. Los puntos pueden ser categorizados usando información de atributos, y estas categorías pueden ser usadas por el optimizador para dar preferencias a puntos de mayor rango cuando se está moviendo o eliminando etiquetas.

El grupo de despliegue usado en este ejercicio muestra un bloque de Condados en el este de Nebraska y las ciudades y pueblos incluidos. En el guión de la página de enfrente, el número de habitantes de cada pueblo en 1990 es usado para asignar uno de los tres *rangos* de valor. Las clasificaciones son usadas para dibujar símbolos de puntos de diferentes tamaños, y también por el optimizador de etiquetas para seleccionar las etiquetas para su eliminación.

Un CartoScript es ejecutado una vez para cada elemento en un objeto, sin embargo la optimización de etiquetas requiere información acerca de todas las etiquetas para resolver los conflictos de posicionamiento. Para resolver este dilema, los guiones de optimización usan la función LineStyleAddToOptimizer( ) para recolectar información acerca de las dimensiones de cada etiqueta. Después que todos los puntos han sido procesados, el optimizador mueve o borra etiquetas conforme sea necesario, luego llama a una función denominada FuncDrawLabel( ) para dibujar las etiquetas. Las instrucciones para esta función deben ser incluidas en la definición de funciones en el guión, tal como se muestra al final de la página de enfrente. Esta definición debería especificar la fuente, color y alto de la etiqueta, e incluir la función LineStyleDrawText( ) o LineStyleDrawTextBox( ).

Los cuatro primeros parámetros de la función LineStyleAddToOptimizer( ) son usados para determinar las dimensiones del rectángulo limitante de la etiqueta. Los parámetros *xstart* y *ystart* fijan la esquina inferior izquierda de la etiqueta, y pueden ser leídas desde las coordenadas del objeto internas. Los parámetros *xlast* y *ylast* que fijan la esquina superior derecha, pueden ser calculadas a partir de los parámetros de alto y longitud de la etiqueta, los que son devueltos por la función LineStyleTextNextPosition( ). Un valor de 0 para el parámetro *dooptimize* limita los cambios en la posición de la etiqueta a un paso simple a través de los puntos de las etiquetas. El parámetro final *dodelete* es usado para activar la eliminación de etiquetas: activo (1) o apagado (0). Con eliminación activada, una etiqueta en conflicto de igual o menor rango puede ser eliminada durante la optimización.

Optimización con Supresión Jerárquica



Despliegue de un grupo con supresión jerárquica. Los círculos indican ciudades para las cuales la etiqueta fue eliminada durante la optimización.

```
# Rank towns by population
pop = TownData.POP90
if (pop < 1000) then rank = 1
else {
    if (pop >= 100000 ) then rank = 3
    else rank = 2
    }
# Draw circle with size based on rank
radius = rank * 500
LineStyleSetColor(255,0,0)
LineStyleDrawCircle(radius,1)
# Read label text and determine dimensions
height = 3500
LineStyleSetFont("ARIALBD.TTF")
LineStyleTextNextPosition(TownData.NAME,height,0,0,next_x,
              next_y,length)
# Define parameters for optimization
xstart = Internal.x;        ystart  = Internal.y
xlast = xstart + length;    ylast = ystart + height
dooptimize = 0;             dodelete = 1
LineStyleAddToOptimizer(xstart,ystart,xlast,ylast,rank,
              dooptimize,dodelete)
# Define required function to draw labels after optimization
func FuncDrawLabel() {
    LineStyleSetFont("ARIALBD.TTF")
    LineStyleSetTextColor(0,0,0)
    height = 3500
    LineStyleDrawText(TownData.NAME,height,0,0)
    }
```

Existen algunos aspectos aleatorios para cambiar la posición de etiquetas durante la optimización, de forma que al volver a dibujar el grupo podría causar que algunas etiquetas cambien de posición y que otras previamente eliminadas reaparezcan. La ilustración superior muestra un posible conjunto de posiciones de etiquetas.

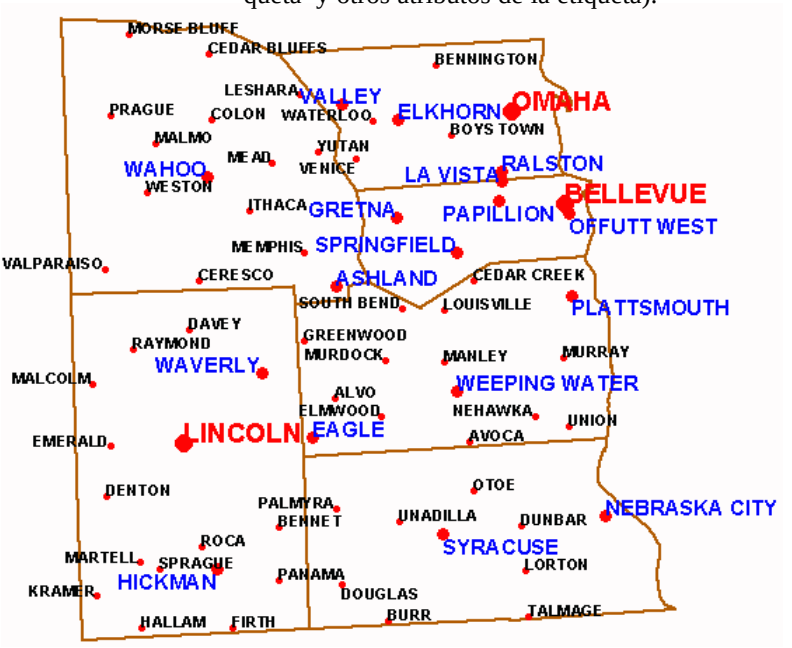
Optimización Completa de Etiquetas

PASOS

- ☑ Abrir la ventana Query Editor para estilos de punto
- ☑ escoger Open / RVC Object del menú File
- ☑ escoger el objeto OPTQRY2 del Archivo de Proyecto TOWNS
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls
- þ cierre el grupo de despliegue actual cuando haya completado este ejercicio.

La optimización completa de etiquetas se activa fijando a 1 el valor del parámetro *dooptimize* en la función LineStyleAddToOptimizer( ). El optimizador entonces realiza múltiples pasadas por las etiquetas de los puntos, para determinar las posiciones óptimas de las etiquetas. El guión en la página de enfrente utiliza optimización completa sin eliminación, para la ubicar las etiquetas. También asigna diferentes tamaños de etiquetas y colores para los pueblos de diferentes rangos, tal como se ilustra abajo.

Note que la función FuncDrawLabel( ) no puede acceder directamente a ninguno de los valores de las variables asignados en el cuerpo principal del guión (incluyendo la variable *rank* usada por el optimizador). Para variar el estilo de dibujo de la etiqueta por rangos, como en este guión, la declaración de la función FuncDrawLabel( ) debe repetir el procedimiento de jerarquización que se halla en el cuerpo principal del guión (al igual que la asignación de fuente, origen del texto de la etiqueta y otros atributos de la etiqueta).



Guión para Optimización Completa

```
# Label sizes for three sizes of towns
small = 2500;  med = 3500;big = 4500

# Rank towns by population
pop = TownData.POP90
if (pop < 1000) then {
    rank = 1;  height = small
    }
else {
    if (pop >= 30000 ) then {
        rank = 3;  height = big
         }
    else {
        rank = 2;  height = med
        }
    }
# Draw circle with size based on rank
radius = rank * 500
LineStyleSetColor(255,0,0)
LineStyleDrawCircle(radius,1)
```

Atajo para CartoScripts
Cualquier símbolo de punto que usted diseñe usando el Editor de Estilos de Puntos, puede también almacenarse como un CartoScript; complete éste con las variables declaradas y comentarios. Usted puede usar este atajo para crear la estructura básica de un guión, luego añadir cualquier característica personalizada o referencia a campos de base de datos conforme necesite.

```
# Read label text and determine dimensions
LineStyleSetFont("ARIALBD.TTF")
LineStyleTextNextPosition(TownData.NAME,height,0,0,next_x,
                  next_y,length)
# Define parameters for optimization
xstart = Internal.x;      ystart  = Internal.y
xlast = xstart + length;  ylast = ystart + height
dooptimize = 1;        dodelete = 0
LineStyleAddToOptimizer(xstart,ystart,xlast,ylast,rank,
                  dooptimize,dodelete)

# Define required function to draw labels after optimization
func FuncDrawLabel() {
    small = 2500;   med = 3500;   big = 4500
    pop = TownData.POP90
    if (pop < 1000) {
        height = small;  LineStyleSetTextColor(0,0,0)
        }
    else {
        if (pop >= 30000 ) {
            height = big;   LineStyleSetTextColor(255,0,0)
            }
        else {
            height = med;   LineStyleSetTextColor(0,0,255)
            }
        }
    LineStyleSetFont("ARIALBD.TTF")
    LineStyleDrawText(TownData.NAME,height,0,0)
    }
```

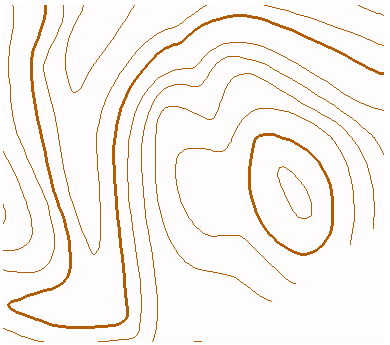
Ancho de Curvas de Nivel por Guión

PASOS

- ☑ clic el botón del icono New 2D Group en la barra de herramientas Display 
- ☑ clic en el botón del icono Add Vector en la ventana Group Controls y escoja Add Vector Layer 
- ☑ seleccione el objeto CONTOURS del Archivo de proyecto CARTOSMP
- ☑ clic en la tarjeta Lines en la ventana Vector Object Display Controls
- ☑ seleccione By Script de la opción del menú Style y clic [Specify...]
- ☑ escoja Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccione el objeto CONQRY del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls
- ☑ use la herra mienta Zoom Box para acercarse al área de curvas cerrada cerca al extremo inferior del objeto 

Los CartoScripts pueden usarse también para dibujar símbolos simples o complejos para elementos lineales en los objetos Vector o CAD. La función más básica para dibujar símbolos de líneas es LineStyleDrawLine( ), la cual no tiene parámetros. Esta simplemente dibuja una línea sólida para cada línea usando el color fijado por la función LineStyleSetColor( ) y el ancho establecido por la función LineStyleSetLineWidth( ). Dejando la posición del lápiz al inicio de la línea después de dibujar.

El guión en este ejercicio dibuja líneas con diferentes anchos para las curvas de nivel índices y secundarias. El mapa original tiene un intervalo de curvas de 40 pies, y cada quinta curva (divisible uniformemente para 200 pies) es una índice mostrada con una línea de ancho mayor. Sin embargo, en TNTmips los valores internos de Z son almacenados en metros. El guión lee desde la Tabla Interna el valor mínimo de Z para la línea y lo convierte a pies. La elevación en pies es luego dividida para 200 usando el operador *módulo*, el cual retorna el saldo de la división, almacenada aquí en la variable *rem*. El valor de *rem* es 0 únicamente para las curvas índices, de forma que este valor es usado para asignar el ancho apropiado antes de dibujar la línea.



```
# ConQry
# Read contour elevation & convert to feet
elevm = Internal.MinZ
elevft = round(elevm * 3.28084)
      # Use modulo operator to
      # identify contour elevations
      # that are not evenly divisible
      # by 200 (nonzero remainder)
      rem = elevft % 200
      # Define widths for minor and
      # major contours
      if (rem  <> 0) width = 2
         else width = 6
      # Set line color and width and
      # draw line
      LineStyleSetColor(170,85,0)
      LineStyleSetLineWidth(width)
      LineStyleDrawLine()
```

Navegando por las Líneas

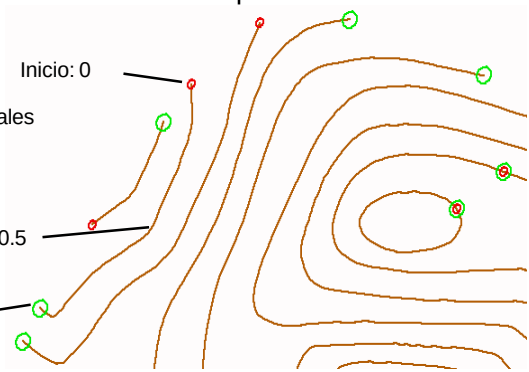
Otras funciones de dibujo pueden ser utilizadas conjuntamente con la función LineStyleDrawLine( ) para crear símbolos de líneas más complejos. El guión en este ejercicio primero dibuja los elementos lineales como líneas sólidas, luego dibuja círculos de diferente tamaño y color en cada fin de línea. Este guión podría ser usado para elementos lineales vector que esté editando en el Spatial Data Editor.

PASOS

- ☑ volver a abrir la ventana Query Editor para estilos de líneas
- ☑ escoger Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccionar el objeto StartEndQry del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

Ciertas funciones son provistas para permitir navegar a lo largo de los elementos lineales con el fin de dibujar símbolos componentes. La función LineStyleSetPosition( ) es utilizada en este guión para mover la posición de lápiz al final de cada línea después de marcar el inicio. El único parámetro numérico de esta función especifica una posición en la línea como una distancia relativa entre 0 (inicio de la línea) y 1.0 (fin de la línea). Usted puede utilizar la función LineStyleGetPosition( ) para encontrar la posición actual del lápiz y asignar el valor devuelto a una variable. Fije a 0 el valor del único parámetro de la función, si desea que el valor devuelto sea la posición relativa. Si lo fija a 1, el valor devuelto es la distancia absoluta desde el inicio en coordenadas del objeto.

```
# StartEndQry
# Set radius parameters for small
and
# large circles
radius1 = 8;   radius2 = 16;   dofill
= 0
# Set line color and width and draw
line
LineStyleSetColor(170,85,0)
LineStyleSetLineWidth(4)
LineStyleDrawLine()
# Draw small red circle at start of
line
LineStyleSetColor(225,0,0)
LineStyleDrawCircle(radius1,dofill)
# Move to end of line and draw large
```



Posición relativa en elementos lineales usando las funciones LineStyleSetPosition() y LineStyleGetPosition( ).

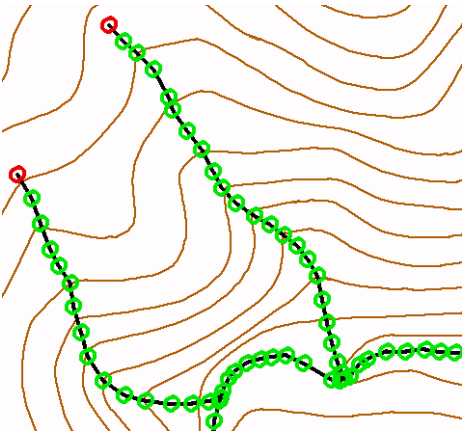
Marcando los Vértices de Líneas

PASOS

- ☑ clic en el botón del  icono Add Vector en la ventana Group Controls y escoja Add Vector Layer
- ☑ seleccione el objeto STREAMS del Archivo de Proyecto CARTOSMP
- ☑ clic en la tarjeta Lines en la ventana Vector Object Display Controls
- ☑ seleccione By Script de la opción del menú Style y clic [Specify...]
- ☑ escoja Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccione el objeto VERTEXQRY del Archivo de proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

Los CartoScripts pueden también marcar los vértices de líneas como una ayuda para la edición. El guión en este ejercicio dibuja un círculo rojo al inicio de cada elemento lineal, y uno verde en cada vértice subsiguiente.

La función LineStyleNextVertex( ) utilizada en este guión mueve la posición del lápiz al siguiente vértice a lo largo de la línea. La función también devuelve un valor de 1 si el fin de la línea ha sido hallado, caso contrario 0. (La función LineStylePrevVertex( ) mueve al vértice previo y devuelve un valor de 1 al inicio de la línea). Estas funciones pueden por lo tanto ser usadas en una estructura de bucle "while" para repetir un conjunto de acciones de dibujo en cada vértice. En este guión, el bucle "while" se repite mientras la función LineStyleNext-Vertex( ) devuelve un valor de 0. El bucle termina cuando la función devuelve el valor de 1 al final de la línea.



```
# VertexQry
# Set parameters for circles
# marking vertices
radius = 5;    dofill = 0

# Draw solid black line
LineStyleSetLineWidth(3)
LineStyleSetColor(0,0,0)
LineStyleDrawLine()

# Draw red circle at beginning of line
LineStyleSetColor(225,0,0)
LineStyleDrawCircle(radius,dofill)

# While not at end of line, move to
# next vertex and draw green circle
LineStyleSetColor(0,225,0)
while (LineStyleNextVertex() <> 1) {
    LineStyleDrawCircle(radius,dofill)
    }
```

- ☑ remueva el layer STREAMS cuando haya completado este ejercicio 

Dibujando Símbolos Espaciados Regularmente

Los símbolos de líneas que usted crea para usar en formatos de mapas incluyen componentes espaciados regularmente a lo largo de cada línea. Este ejercicio ilustra la estructura básica de un guión de ese tipo dibujando círculos rellenos espaciados regularmente a lo largo de las líneas.

La función LineStyleRoll( ) mueve una distancia especificada a lo largo de la línea sin dibujar. La distancia a moverse es fijada por el único parámetro de la función. La función también devuelve un valor de 0 por cualquier posición en la línea excepto por el final, donde devuelve un valor de 1. La función puede luego ser usada en una estructura de bucle "while" para repetir un conjunto de acciones de dibujo a intervalos regulares a lo largo de cada línea.

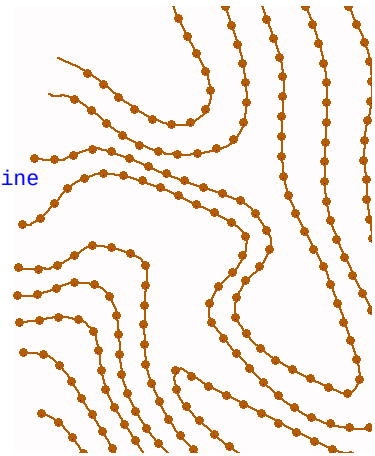
PASOS

- ☑ abrir nuevamente la ventana Query Editor para los estilos de línea del objeto CONTOURS
- ☑ escoger Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccionar el objeto CIRCLINEQRY del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

Este guión está estructurado para continuar dibujando círculos en tanto la distancia desde la posición actual al fin de la línea sea mayor que el espaciamiento deseado. La función LineStyleGetDistanceTo( ) es utilizada para verificar esta distancia. La distancia a varias características de la línea, puede ser determinada fijando el valor apropiado del parámetro para esta función, tal como se muestra en el recuadro de la derecha.

Valores de Parámetros para la función LineStyleGetDistanceTo( ):
1 = siguiente vértice
2 = vértice previo
3 = fin de la línea
4 = inicio la línea

```
# CircLineQry
# Set parameters for circles
radius = 6;  dofill = 1
# Set spacing between circles
spacing = 30

# Set line color and width and draw line
LineStyleSetColor(170,85,0)
LineStyleSetLineWidth(3)
LineStyleDrawLine()

# Draw circle at start of line
LineStyleDrawCircle(radius,dofill)

# Draw rest of circles
while (LineStyleRoll(spacing) <> 1) {
    dist = LineStyleGetDistanceTo(3)
    if ( dist > spacing) {
        LineStyleDrawCircle(radius,dofill)
        }
    }
```



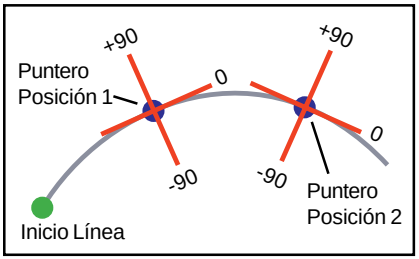
Posiciones y Sistemas de Coordenadas

PASOS

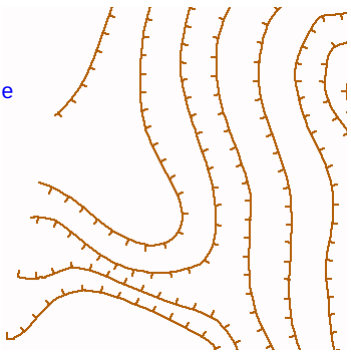
- ☑ abrir nuevamente la ventana Query Editor para estilos de líneas del objeto CONTOURS
- ☑ escoger Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccionar el objeto TICKLINEQRY del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

El motor de dibujo de CartoScripts mantiene un registro de dos posiciones durante la ejecución del guión. La primera es la posición actual a lo largo de la línea, la cual puede pensar como el "puntero" que es movido a lo largo de la línea por la función LineStyleRoll( ) o por las otras funciones de navegación de líneas. La segunda posición que es registrada es la posición del lápiz, la cual puede o no estar sobre la línea.

La posición actual del puntero sirve como el origen de un sistema local de coordenadas que está orientado relativo a la línea como se muestra en la ilustración del recuadro. Las funciones de dibujo que usan parámetros de ángulo y distancia para mover la posición del lápiz o dibujar elementos tales como LineStyleLineTo( ) y LineStyleDrawArrow( ), referencian este sistema local de coordenadas. Este sistema le permite dibujar componentes de símbolos repetidos que son consistentemente orientados relativos a la dirección local de la línea, tal como las marcas perpendiculares en el guión de este ejercicio. Las marcas son dibujadas en el lado izquierdo de cada línea (relativo a los puntos inicial y final). Si símbolos asimétricos como éstos se necesitan dibujar, en un lado particular de cada línea, usted podría necesitar revertir la dirección de las líneas individuales usando el Spatial Data Editor para obtener el símbolo deseado.




```
# TickLineQry
# Denominator of
# desired map scale
scale = 5000
# Desired spacing and
# length of tick lines
# in mm at map scale
spaceMap = 6; lengthMap = 1.5
# Scaled spacing and length
# of tick lines
spacing = spaceMap * scale / 1000
length = lengthMap * scale / 1000
# Set line color and width and draw line
LineStyleSetColor(170,85,0)
LineStyleSetLineWidth(3)
LineStyleDrawLine()
# Draw tick lines
LineStyleLineTo(90,length)
while (LineStyleRoll(spacing) <> 1) {
    LineStyleMoveTo(0,0)
    LineStyleLineTo(90,length)
    }
```

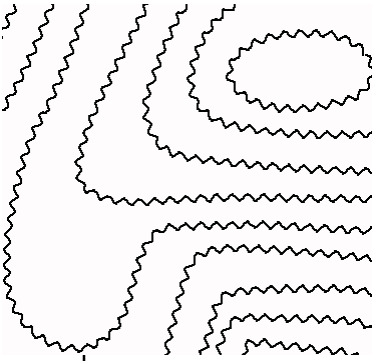


Más Acerca de Posiciones del Lápiz

Cuando usted utiliza la función LineStyleRoll( ) necesita recordar que ésta mueve el puntero, pero no la posición del lápiz. En muchos guiones este punto no es tan obvio, porque la mayoría de funciones de dibujo referenciadas localmente mueven el lápiz al origen del sistema local de coordenadas antes de dibujar. En el guión círculo-línea (pag. 21), por ejemplo, la función LineStyleDrawCircle( ) automáticamente mueve el lápiz a la posición actual del puntero después de cada acción LineStyleRoll. Una función que ***no*** actualiza la posición del lápiz antes de dibujar es LineStyleLineTo( ). Esta función dibuja una línea desde la posición actual del lápiz hasta un punto especificado en el sistema local de coordenadas. Cuando desea utilizar esta función en un bucle LineStyleRoll para dibujar una línea iniciando en el origen del sistema local de coordenadas, tal como en el guión de las marcas de la página precedente, debe usar la declaración "LineStyleMoveTo(0,0)" para mover el lápiz a la posición actual del puntero antes de dibujar.

PASOS

- ☑ nuevamente abrir la ventana Query Editor para estilos de líneas del objeto CONTOURS
- ☑ escoger Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccionar el objeto SINEWAVEQRY del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls



Las peculiaridades de las funciones LineStyleRoll( ) y LineStyleLineTo( ) fueron diseñadas con un propósito. Ellas hacen posible dibujar líneas entrecortadas o continuas separadas (offset) del elemento vector lineal. El guión en este ejercicio muestra un ejemplo elegante, un símbolo de línea que parece como una curva sinusoidal. La "curva" actualmente esta formada por pequeños segmentos de línea recta dibujados por la función LineStyleLineTo( ). Cada iteración del bucle "while" incrementa en un radian el ángulo para la función *sine*, produciendo una amplitud sinusoidal variable para la función LineStyleLineTo( ).

```
# SineWaveQry
# Set line color and width
LineStyleSetColor(0,0,0)
LineStyleSetLineWidth(3)
# Set sine wave parameters
angle = 0; space = 4
# Draw line
while (LineStyleRoll(space) <> 1) {
    angle = angle + 1    # in radians
    a = sin(angle) * 5       # ampli-
tude
    if (a > 0) then
LineStyleLineTo(90,a)
```

Dibujando Líneas Entrecortadas

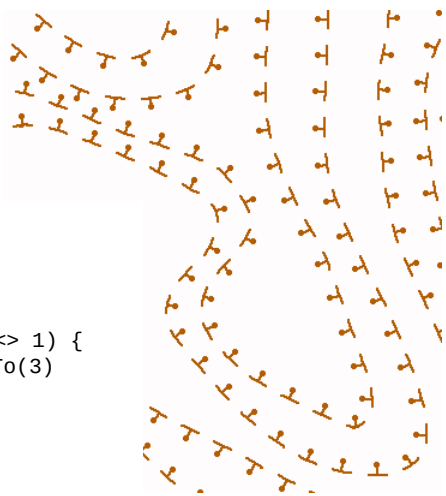
PASOS

- ☑ abrir nuevamente la ventana Query Editor para estilos de línea para el objeto CONTOURS
- ☑ escoger Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccionar el objeto BARBQRY del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

Para dibujar líneas entrecortadas simples o complejas, usted puede usar un bucle "while", para alternar las acciones LineStyleRoll( ) y LineStyleRollPen( ). La función LineStyleRoll-Pen( ) dibuja una línea a lo largo de un elemento lineal por una distancia especificada, iniciando en la posición actual del puntero. La distancia de dibujo es fijada por el valor del único parámetro de la función. Para una línea entrecortada simple con trazos y espacios de igual tamaño, use la misma distancia para las dos funciones LineStyleRoll( ) y LineStyleRollPen( ).

El guión en este ejercicio es un poco más elaborado, añadiendo al medio de cada trazo, una línea de marca con un círculo relleno al final de la misma. En cada iteración del bucle "while", la función LineStyleRollPen(), dibuja la primera mitad del trazo, luego son dibujados la línea de marca y círculo, y finalmente se dibuja la otra mitad del trazo.

```
# BarbQry
# Set line color and width
LineStyleSetColor(170,85,0)
LineStyleSetLineWidth(3)

# Set dash parameters
dashSize = 20
half = 0.5 * dashSize

# Set circle parameters
radius = 3;    dofill = 1

# Draw line
while (LineStyleRoll(dashSize) <> 1) {
    dist = LineStyleGetDistanceTo(3)
    if (dist > dashSize) {
        LineStyleRollPen(half)
        LineStyleMoveTo(0,0)
        LineStyleDropAnchor(0)
        LineStyleLineTo(90,half)
        LineStyleDrawCircle(radius,dofill)
        LineStyleMoveToAnchor(0)
        LineStyleRollPen(half)
        }
    else LineStyleRollPen(dist)
    }
```



☑ remover el layer CONTOURS cuando usted haya completado este ejercicio

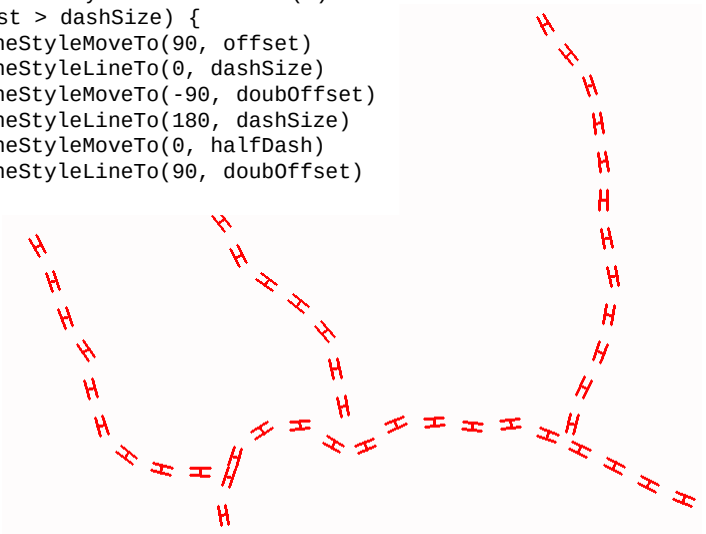
Líneas Entrecortadas Dobles

El guión en este ejercicio ilustra otra variación de línea entrecortada. Este dibuja cada línea con trazos dobles, conectados por líneas que cruzan. Cada par de trazos está centrado en el elemento lineal, lo que significa que los trazos en sí mismo están separados de la línea por una distancia especificada por la variable *offset*. Debido a que ninguna parte del elemento lineal mismo es dibujado, la función LineStyleLineTo( ) se usa para dibujar los trazos y las líneas que cruzan.

PASOS

- ☑ clic en el botón del icono Add Vector en la ventana Group Controls y escoja el layer Add Vector
- ☑ seleccione el objeto STREAMS del Archivo de Proyecto CARTOSMP
- ☑ clic en la tarjeta Lines en la ventana Vector Object Display Controls
- ☑ seleccione By Script de la opción Style del menú y clic [Specify...]
- ☑ escoja Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccione el objeto DOUBDASHQRY del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

```
# DoubDashQry
dashSize = 15
halfDash = dashSize * 0.5
double = 2 * dashSize
offset = dashSize * 0.2
doubOffset = offset * 2

# Set line color and width
LineStyleSetColor(255,0,0)
LineStyleSetLineWidth(2)

# Draw double dash line and crossing lines
LineStyleMoveTo(90, offset)
LineStyleLineTo(0, dashSize)
LineStyleMoveTo(-90, doubOffset)
LineStyleLineTo(180, dashSize)
LineStyleMoveTo(0, halfDash)
LineStyleLineTo(90, doubOffset)

while (LineStyleRoll(double) <> 1) {
    dist = LineStyleGetDistanceTo(3)
    if (dist > dashSize) {
        LineStyleMoveTo(90, offset)
        LineStyleLineTo(0, dashSize)
        LineStyleMoveTo(-90, doubOffset)
        LineStyleLineTo(180, dashSize)
        LineStyleMoveTo(0, halfDash)
        LineStyleLineTo(90, doubOffset)
        }
    }
```



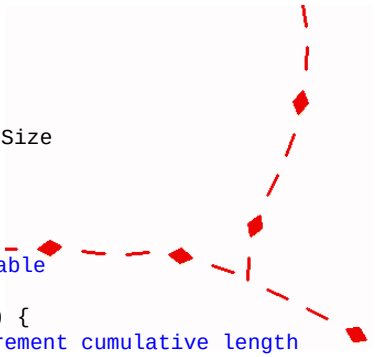
Manejando Multiple Repetición de Intervalos

PASOS

- ☑ abrir nuevamente la ventana Query Editor para line styles del objeto STREAMS
- ☑ escoger Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccione el objeto DashDiQry del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls

Los símbolos de línea pueden incluir componentes repetidos a diferentes intervalos a lo largo de la línea, tal como se ilustra por los trazos y el paralelogramo dibujados por el guión en este ejercicio. La variable *cumL* en el guión mantiene el registro de la longitud acumulativa de los trazos y espacios dibujados desde el último símbolo paralelogramo. Cuando el valor de *cumL* alcanza la distancia especificada para los paralelogramos, un símbolo de éstos es dibujado en lugar de un trazo, y el valor de *cumL* es restablecido a 0. El símbolo paralelogramo es creado usando la función Line-StyleSideshot( ). Esta función permite especificar un número de puntos por ángulo y distancia en el sistema local de coordenadas y conectarlos para formar una polilínea, fijando a 1 el valor del parámetro *dodraw*. Este guión también registra los puntos como un polígono, permitiendo que la figura del paralelogramo pueda ser rellenada.

```
# DashDiQry
# Set line color and width
LineStyleSetColor(225,0,0)
LineStyleSetLineWidth(2)
# Set dash parameters
dashSize = 12;     half = 0.5 * dashSize
# Set diamond parameters
spacing = 5 * dashSize
dodraw = 1;    dofill = 1
width = 0.3 * dashSize
cumL = 0   # Cumulative length variable
# Draw line
while (LineStyleRoll(dashSize) <> 1) {
    cumL = cumL + dashSize    # increment cumulative length
    if (cumL >= spacing) {    # draw diamond symbol
        LineStyleRoll(half)
        LineStyleMoveTo(0,0)
        LineStyleRecordPolygon(1)
        LineStyleSideshot(dodraw,0,half,90,width,180,
            half,-90,width)
        LineStyleDrawPolygon(dofill)
        LineStyleRoll(half)
        cumL = 0       # reset cumulative length to 0
        }
    else {
        LineStyleRollPen(dashSize)# draw dash
        cumL = cumL + dashSize    # increment cumulative length
        }
    }
```



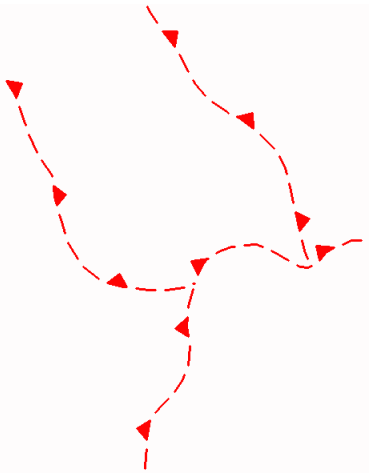
Símbolo de Fallas con Línea Entrecortada

El guión en esta consulta provee otro ejemplo de símbolos utilizando diferentes intervalos repetidos. Este dibuja la versión del símbolo para fallas en mapas geológicos. Los triángulos son dibujados como polígonos usando puntos de anclaje en las esquinas. Las bases de los símbolos de los triángulos se dibujan usando la función LineStyleRollPen( ), de forma que ellas se ajusten a las curvas de los elementos lineales. Note que una distancia negativa puede ser usada con la función LineStyleRoll( ) para moverse hacia atrás a lo largo del elemento lineal.

PASOS

- ☑ abra nuevamente la ventana Query Editor para line styles del objeto STREAMS
- ☑ escoja Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccione el objeto DashThrustQry del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls
- ☑ remueva el layer STREAMS cuando haya completado este ejercicio

```
# DashThrustQry
dashSize = 16      # length of dashes
halfDash = dashSize * 0.5
# triangle spacing and dimensions
spacing = dashSize * 6
quartSpace = spacing * 0.25
triWidth = dashSize
halfTri = triWidth * 0.5
height = dashSize * 0.6
# Set line color and width
LineStyleSetColor(255,0,0)
LineStyleSetLineWidth(2)
# Initialize variable to control placement of triangles
cumL = spacing
# Draw dashed fault line and triangles
while (LineStyleRoll(halfDash) <> 1) {
    dist = LineStyleGetDistanceTo(3)  # distance to end of line
    cumL = cumL + dashSize
    if (dist > quartSpace and cumL >= spacing) {
        LineStyleDropAnchor(1)
        LineStyleRoll(halfTri)
        LineStyleMoveTo(0,0)
        LineStyleMoveTo(90, height)
        LineStyleDropAnchor(2)
        LineStyleRoll(-halfTri)
        LineStyleRecordPolygon()
        LineStyleRollPen(triWidth)
        LineStyleLineToAnchor(2)
        LineStyleLineToAnchor(1)
        LineStyleDrawPolygon(1)
        cumL = 0
        }
    else {
        LineStyleRollPen(dashSize)
        cumL = cumL + dashSize
        }
    }
```



Símbolo de Línea Sinclinal

PASOS

- ☑ clic en el botón del icono Add Vector en la ventana Group Controls y escoja Add Vector Layer
- ☑ seleccionar el objeto SYNCLINE del Archivo de Proyecto CARTOSMP
- ☑ clic en la tarjeta Lines en la ventana Vector Object Display Controls
- ☑ seleccione By Script de la opción del menú Style y clic [Specify...]
- ☑ escoger Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccione el objeto SYNCLINEQRY del Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamen te en la ventana Vector Object Display Controls

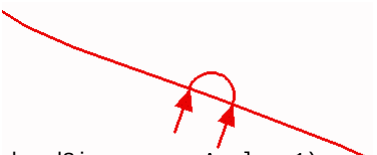
Un CartoScript puede estructurarse para dibujar diferentes símbolos de líneas, dependiendo de los atributos enlazados al elemento lineal. El guión en este ejercicio dibuja símbolos de líneas geológicas para la traza axial de un pliegue sinclinal (descendente) en rocas sedimentarias. Si una rama del pliegue se ha rotado más allá de la orientación vertical, el pliegue esta derribado, y se utiliza un símbolo especial. La condición de derribado para líneas vector en este ejercicio se indica por un campo lógico en la base de datos.

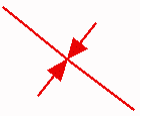
Este guión dibuja cada elemento lineal como una línea sólida, luego ubica el símbolo del pliegue en el medio de la línea. El semicírculo que forma parte del símbolo del sinclinal derribado, se dibuja usando la función LineStyleDraw-Arc( ), la que dibuja un arco alrededor de un punto central especificado. Los dos primeros parámetros de la función, especifican un ángulo y distancia al punto central del arco proyectado, de forma que tiene la opción de mover el lápiz a una nueva ubicación, antes de dibujar el arco. También puede especificar un ángulo inicial y el ángulo de barrido para el arco, y tener la opción de girando el arco entero después de dibujar, asignando un valor distinto de cero para el parámetro *rotangle*

```
# SynclineQry
# Read a logical database field (Yes/No) to check if syncline is
# overturned.  Numeric variable is set to 1 if yes, 0 if no
overturned = Syncline.Overturned
# dimensions of arrow symbols
arrowSize = 30;    halfSize = arrowSize * 0.5
headSize = 0.5 * arrowSize;   sweepAngle = 45
stemSize = arrowSize - headSize * cosd(sweepAngle)
width = 2          # width of lines
# Parameters for half circle in overturned syncline symbol
angle = 0;  shift = halfSize   # angle and distance to arc center
radius_x = halfSize            # radii of arc
radius_y = halfSize
startAngle = -180;     swpAngle = -180
rotAngle = 0;          isAngleAbs = 0
# Set line color, width, and draw fold line
LineStyleSetColor(228,0,0)
LineStyleSetLineWidth(width)
LineStyleDrawLine()
```

Símbolo de Línea Sinclinal (continuación)

```
# Draw arrow symbols in middle of each line
LineStyleSetPosition(0.5)     # move to middle of line
 if ( overturned == 1 ) {     # draw overturned syncline symbol
    # Draw arrows with 0 line width
    LineStyleRoll(-halfSize)
    LineStyleDropAnchor(1)
    LineStyleSetLineWidth(0)
    LineStyleMoveTo(-90, arrowSize)
    LineStyleDropAnchor(2)
    LineStyleDrawArrow(90, arrowSize, headSize, sweepAngle, 1)
    LineStyleMoveTo(0, arrowSize)
    LineStyleMoveTo(-90, arrowSize)
    LineStyleDropAnchor(3)
    LineStyleDrawArrow(90, arrowSize, headSize, sweepAngle, 1)

    # Draw arrow stems and arc with linewidth = width
    LineStyleSetLineWidth(width)
    LineStyleMoveToAnchor(2)
    LineStyleLineTo(90, stemSize)
    LineStyleMoveToAnchor(1)
    LineStyleDrawArc(angle, shift, radius_x, radius_y,
                              startAngle, swpAngle, isAngleAbs)
    LineStyleMoveToAnchor(3)
    LineStyleLineTo(90, stemSize)
    }
else {      # draw upright syncline symbol
    # Draw arrows with 0 line width
    LineStyleDropAnchor(1)
    LineStyleSetLineWidth(0)
    LineStyleMoveTo(90, arrowSize)
    LineStyleDropAnchor(2)
    LineStyleDrawArrow(-90, arrowSize, headSize, sweepAngle, 1)
    LineStyleMoveToAnchor(1)
    LineStyleMoveTo(-90, arrowSize)
    LineStyleDropAnchor(3)
    LineStyleDrawArrow(90, arrowSize, headSize, sweepAngle, 1)

    # Redraw arrow stems
    LineStyleMoveToAnchor(2)
    LineStyleSetLineWidth(width)
    LineStyleLineTo(-90, stemSize)
    LineStyleMoveToAnchor(3)
    LineStyleLineTo(90, stemSize)
    }
```





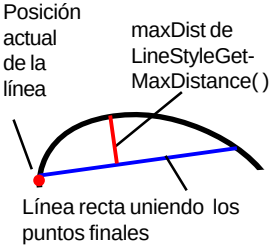
☑ Remover el layer SYNCLINE cuando haya completado este ejercicio.

Este guión dibuja las flechas de buzamiento para el símbolo de sinclinal derribado en el lado derecho del elemento lineal. Las direcciones de las líneas se han fijado en el objeto vector para acomodar el diseño de este guión.

Etiquetando Curvas de Nivel

PASOS

- ☑ clic en el icono del botón Add Vector en la ventana Group Controls y escoja Add Vector Layer
- ☑ seleccione el objeto CONTOURS desde el Archivo de Proyecto CARTOSMP
- ☑ clic en la tarjeta Lines en la ventana Vector Object Display Controls
- ☑ seleccione By Script de la opción del menú Style y clic [Specify...]
- ☑ escoja Open / RVC Object del menú File en la ventana Query Editor
- ☑ seleccione el objeto CONLABLQRY desde el Archivo de Proyecto CARTOSMP
- ☑ clic [OK] en la ventana Query Editor y nuevamente en la ventana Vector Object Display Controls



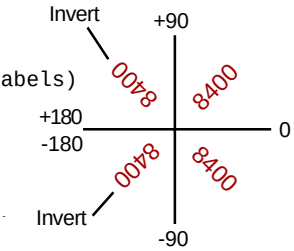
Usted puede diseñar un CartoScript para modificar la ubicación y orientación de los componentes de los símbolos de líneas, para acomodar la dirección local y forma de los elementos de la línea. En este guión que dibuja y etiqueta las curvas de nivel, los valores de elevación para las curvas índices son trasladados si la porción local de la línea es demasiado curvada, y si es necesario son invertidas para hacerles leíbles. (Unicamente la porción del proceso principal del guión se muestra en la página siguiente).

Estas condiciones son verificadas usando las funciones LineStyleGetDirection( ) y LineStyleGetMaxDistance( ) . El primer parámetro de ambas funciones es un valor que especifica la longitud de la porción de línea que usted desea examinar. Los parámetros adicionales de las funciones son variables a las que se asignan valores por medio de la función. La función Line-StyleGetDirection( ) encuentra los ángulos de dirección máximo y mínimo para el segmento de línea especificado en el marco de referencia de las coordenadas del objeto (eje X positivo = 0 grados). La función LineStyleGet- MaxDistance( ) encuentra la distancia perpendicular máxima entre la porción especificada del elemento lineal y una línea recta que une sus puntos extremos (mirar la ilustración a la izquierda), al igual que el ángulo de dirección de este segmento de línea recta (usada en este guión para orientar la etiqueta de las curvas).

Ambas funciones retornan un valor de 1 si estamos al final del elemento lineal o 0 caso contrario. El guión verifica le valor de la distancia máxima, así como el cambio en la dirección de la línea sobre la longitud de la etiqueta para determinar si la porción local de la línea es demasiado curvada para ubicar la etiqueta en ese lugar. El ángulo de dirección de la etiqueta de la línea se usa para identificar las etiquetas que necesitan ser invertidas.

Etiquetando Curvas de Nivel (continuación)

```
if (rem <> 0) {            # draw minor contours
    LineStyleSetLineWidth(width)
    LineStyleDrawLine()
    }
else {                     # draw and label major contours
    LineStyleSetLineWidth(widthBold)
    str$ = sprintf("%d",elev) # read elevation to string variable
    # find length of contour label
    LineStyleTextNextPosition(str$,labelSize,0,1,nextx,
                                                nexty,length)
    begShift = 5 * length     # offset from beginning of line
    stopLength = begShift     # min label distance from end of line
    spLength = 1.5 * length   # label length plus spaces

    LineStyleRollPen(begShift)    # draw beginning of contour line

    while ((LineStyleGetMaxDistance(spLength,drawAngle,
                                                maxDist)) <> 1) {
        LineStyleGetDirection(0.1 * labelSize,minAngle,maxAngle)
        # find change in line direction over length of label
        devAngle = drawAngle - minAngle
        # find distance to end of line and compare to stopLength
        remLength = LineStyleGetDistanceTo(3)
        if (remLength < stopLength) break
        # check deviation distance and angle of line segment
        if ((maxDist < 1.5 * labelSize) and (abs(devAngle) < 15)) {
            LineStyleRoll(0.25*length) # space before drawing label

            # Check if label needs to be inverted to be readable
            isInverse = 0
            if (abs(drawAngle) > 90) isInverse = 1
            if (isInverse) {
                LineStyleMoveTo(devAngle, length)
                LineStyleMoveTo(devAngle + 90, halfSize)
                LineStyleDrawText(str$,labelSize,drawAngle+180,1)
                }
            else {
                LineStyleMoveTo(-90,halfSize)
                LineStyleDrawText(str$,labelSize,drawAngle,1)
                }
            LineStyleRoll(1.2 * length)
            LineStyleMoveTo(0,0)
            LineStyleSetLineWidth(widthBold)
            LineStyleRollPen(minDistBetweenLabels)
            }
        else LineStyleRollPen(length)
        }
    # Draw remainder of line
    LineStyleRollPen(remLength)
    }
```



Lista de Funciones CartoScript

Funciones de Cotrol de Estilos

LineStyleSetCapJoinType	Poner finales de segmentos de línea o polilínea cuadrados o redondeados
LineStyleSetColor	Fijar valores RGB (0-255) para color de líneas
LineSyleSetCoordType	Fijar tipos de coordenadas para ingreso de valores (objeto o milímetros)
LineStyleSetFont	Fijar el nombre de la fuente para los textos
LineStyleSetLineWidth	Fijar el ancho de líneas a ser dibujadas
LineStyleSetScale	Fijar el factor de escala aplicado a las distancias ingresadas
LineStyleSetTextColor	Fijar valores RGB (0-255) para color de textos

Optimización de Etiquetas

LineStyleAddToOptimizer	Accesar el optimizador de ubicación de etiquetas

Funciones que referencian posición dentro de elementos lineales

Navegación

LineStyleNextVertex	mover lápiz al siguiente vértice en la línea
LineStylePrevVertex	mover lápiz al vértice previo en la línea
LineStyleRoll	mover el puntero una distancia específica en la línea
LineStyleSetPosition	mover lápiz a una posición relativa especificada en la línea (0.0 = inicio, 1.0=final)

Dibujo

LineStyleRollPen	mover lápiz a la posición del puntero y dibujar la línea por una distancia específica

Información

LineStyleGetDirection	encontrar ángulo de dirección mínimo y máximo de la línea
LineStyleGetDistanceTo	encontrar distancia desde el puntero al vértice siguiente (1), previo (2), fin (3), o inicio (4) de la línea.
LineStyleGetLineCurvature	encontrar curvatura del elemento lineal.
LineStyleGetMaxDistance	encontrar distancia perpendicular máxima desde la línea a la recta que une los puntos inicial y final.
LineStyleGetPosition	encontrar posición actual relativa del puntero en la línea (0.0=inicio, 1.0=final)
LineStyleIsClosed	verificar si la línea actual forma un polígono cerrado (1) o no (0)

Nota: el segmento lineal operado por ciertas funciones en el último grupo arriba, inicia en la posición actual del puntero a lo largo de la línea y se extiende a una distancia especificada de él.

Lista de Funciones CartoScript (continuación)

Funciones que referencian el sistema local de coordenadas vigente

Navegación

`LineStyleDropAnchor` registrar posición actual del lápiz como numero de anclaje para uso posterior

`LineStyleMoveTo` mover lápiz a ubicación especificada por dirección y distancia desde la posición actual

`LineStyleMoveToAnchor` mover lápiz a posición de anclaje especificada

Dibujo

`LineStyleDrawArc` dibujar arco

`LineStyleDrawArrow` dibujar flecha en la posición especificada por dirección y distancia desde la posición actual

`LineStyleDrawCircle` dibujar círculo relleno o nó

`LineStyleDrawCone` dibujar el cono

`LineStyleDrawCube` dibujar el cubo

`LineStyleDrawCylinder` dibujar cilindro

`LineStyleDrawEllipse` dibujar elipse rellena o nó

`LineStyleDrawPolygon` dibujar polígono conectando puntos previamente registrados

`LineStyleDrawPolyline` dibujar polilínea conectando puntos previamente registrados

`LineStyleDrawRectangle` dibujar rectángulo relleno o nó

`LineStyleDrawText` dibujar etiqueta con texto especificado

`LineStyleDrawTextBox` dibujar etiqueta de texto encuadrada

`LineStyleDrawThreePointArc` dibujar arco conectando tres puntos

`LineStyleLineTo` dibujar línea a ubicación especificada por dirección y distancia desde la posición actual de lápiz

`LineStyleLineToAnchor` dibujar línea a ubicación especificada de anclaje desde la posición actual de lápiz

`LineStyleRecordPolygon` empezar o detener registro de ubicaciones de vértices

`LineStyleSideshot` especificar secuencia de posiciones por dirección y distancia desde la posición actual, y opcionalmente conectar para formar polilínea

`LineStyleTextNextPosition` calcular longitud y posición final de texto

Funciones que dibujan o transforman completo ingreso de líneas

`LineStyleDrawLine` dibujar linea copleta vector o CAD

`LineStyleRestoreLine` restaurar coordenadas originales de línea

`LineStyleSpline` reemplazar línea con línea suavizada

`LineStyleThinLine` reemplazar línea con línea adelgazada

Software Avanzado para Análisis Geosepacial

CARTOSCRIPTS

MicroImages, Inc. produce una línea completa de software profesional para visualización, análisis y publicación de datos geoespaciales. Contáctenos o viste nuestra página en Internet para información detallada del producto.

TNTmips TNTmips es un sistema profesional con una completa integración GIS, análisis de imágenes, CAD, TIN, cartografía de escritorio y gestión de Bases de Datos geoespaciales.

TNTedit TNTedit provee de herramientas interactivas para crear, georeferenciar y editar materiales de proyectos tipo vector, imagen, CAD, TIN y Bases de Datos Relacionales en una gran variedad de formatos.

TNTview TNTview tiene las mismas características poderosas de despliegue de TNTmips y es perfecto para aquellos que no necesitan las características de procesamiento técnico y preparación de TNTmips.

TNTatlas TNTatlas permite publicar y distribuir materiales de proyectos en CD-ROM a bajo costo. Los CDs de TNTatlas pueden ser usados en cualquier plataforma popular de computadora.

TNTserver TNTserver permite publicar sus Atlas en TNT en Internet o en su Intranet. Navegue a través de atlas de geodatos con su navegador web y el applet Java TNTclient.

TNTlite TNTlite es una versión libre de TNTmips para estudiantes y profesionales con proyectos pequeños. Usted puede descargar TNTlite del sitio Internet de MicroImages, o puede ordenar TNTlite en CD-ROM con el conjunto actualizado de folletos *Tutoriales*.

Indice

MicroImages, Inc.

11th Floor - Sharp Tower
206 South 13th Street
Lincoln, Nebraska 68508-2010 USA

Voice: (402) 477-9554
FAX: (402) 477-9559

email: info@microimages.com
internet: www.microimages.com