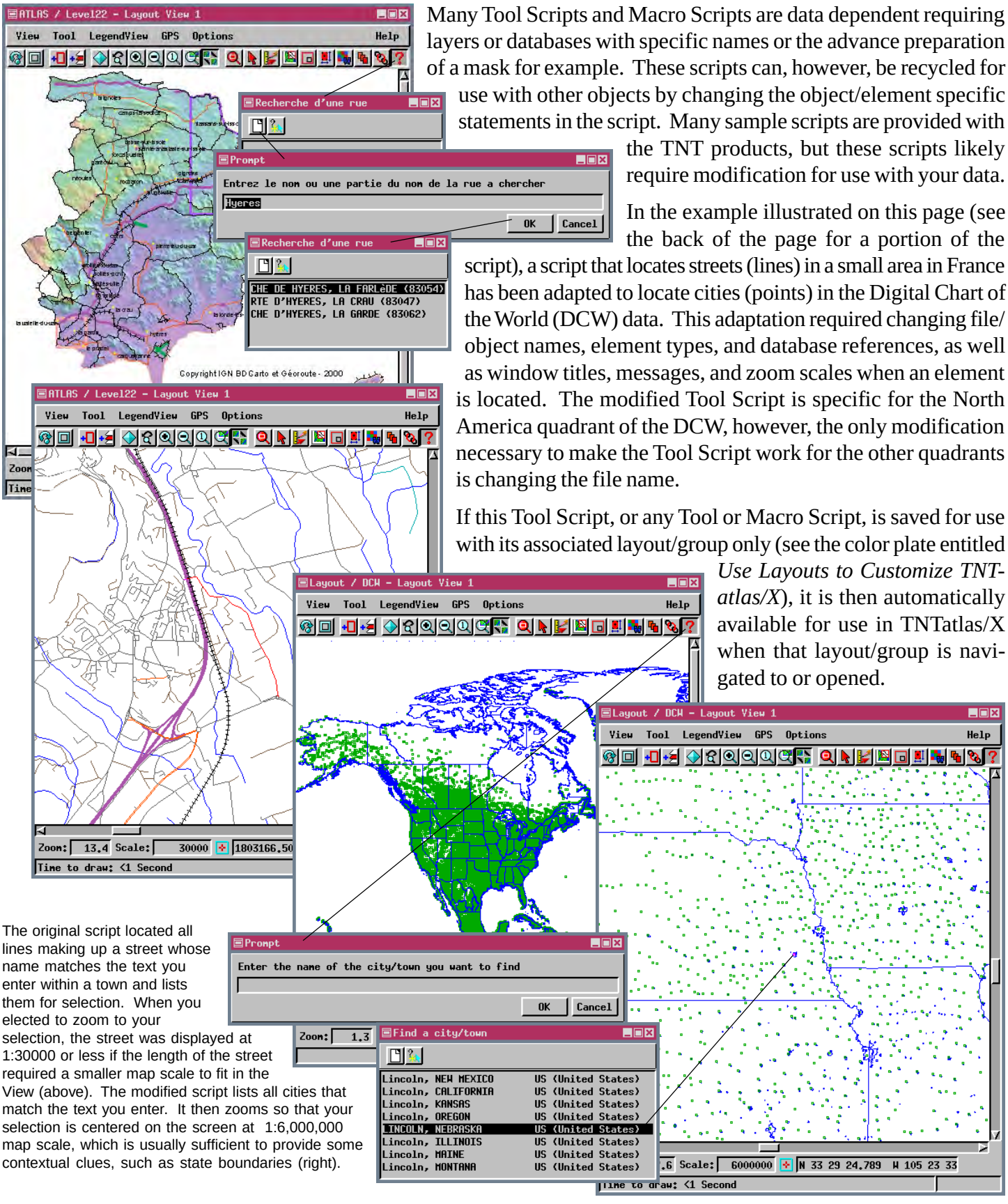
Modifying SML Tool Scripts for New Applications



Many Tool Scripts and Macro Scripts are data dependent requiring layers or databases with specific names or the advance preparation of a mask for example. These scripts can, however, be recycled for use with other objects by changing the object/element specific statements in the script. Many sample scripts are provided with the TNT products, but these scripts likely require modification for use with your data.

In the example illustrated on this page (see the back of the page for a portion of the script), a script that locates streets (lines) in a small area in France has been adapted to locate cities (points) in the Digital Chart of the World (DCW) data. This adaptation required changing file/object names, element types, and database references, as well as window titles, messages, and zoom scales when an element is located. The modified Tool Script is specific for the North America quadrant of the DCW, however, the only modification necessary to make the Tool Script work for the other quadrants is changing the file name.

If this Tool Script, or any Tool or Macro Script, is saved for use with its associated layout/group only (see the color plate entitled *Use Layouts to Customize TNTatlas/X*), it is then automatically available for use in TNTatlas/X when that layout/group is navigated to or opened.

The original script located all lines making up a street whose name matches the text you enter within a town and lists them for selection. When you elected to zoom to your selection, the street was displayed at 1:30000 or less if the length of the street required a smaller map scale to fit in the View (above). The modified script lists all cities that match the text you enter. It then zooms so that your selection is centered on the screen at 1:6,000,000 map scale, which is usually sufficient to provide some contextual clues, such as state boundaries (right).

Excerpt from script for finding streets

(original script available at microimages.com/freestuf/smlscripts.htm)

```
# ToolScript looking for the street$ entered by the user
                    :
        for i=1 to nline {
                curcode =
V.line[linelist[i]].TRONCON_ROUTE.INSEE_COMD;
                if (curcode == codelist[selpos,1]) {
                        nstreetline = nstreetline+1;
                        ResizeArrayPreserve(streetline,
nstreetline);
                        streetline[nstreetline] = linelist[i];
                }
        }
        ViewSetMessage(View, NumToStr(nstreetline) + " lines found
for this street");

        #Zoom in to the lines
        class VECTORLAYERLINES vll;
        vll = layer.Line;
        vll.HighlightMultiple(nstreetline, streetline);
        View.DisableRedraw = 1;
        layer.ZoomToHighlighted();
        if (ViewGetMapScale(View) < 30000) {
                ViewSetMapScale(View, 30000);
        }
        View.DisableRedraw = 0;
        ViewRedraw(View);
}#DoZoom


# New Request
proc DoNew () {
        poslist.DeleteAllItems();
        nline = 0;
        ncode = 0;

        #asking to enter the name of a street (or a word contained in it)
        street$ = PopupString("Entrez le nom ou une partie du nom de
la rue a chercher", "");
        if (street$ == "") return;
        street$ = toupper$(street$);

        #looking for a line containing street$ in its NOM_RUE_D or
NOM_RUE_G attributes of the TRONCON_ROUTE table
        for i=1 to NumVectorLines(V) {
                #class DATABASE DB =
V.line[i].TRONCON_ROUTE;
                if (V.line[i].TRONCON_ROUTE.NOM_RUE_D$
contains street$ or V.line[i].TRONCON_ROUTE.NOM_RUE_G$ contains
street$) {
                        nline = nline+1;
                        linelist[nline] = i;
                }#if
        }#i
        ViewSetMessage(View, NumToStr(nline) + " lines found");

        if (nline == 0) {       #no element corresponding found
                PopupMessage("Aucune rue de l'échantillon de la
base de donnee ne contient ce mot!");
                return;
        }

        #Some streets are found : find the different ones (by zip code)
        #Assertion : not 2 streets with the same name in a town
        #Limits : don't take into account the streets separating 2 towns
(the right zip code INSEE_COMD and the left one INSEE_COMG are
different)
        for i=1 to nline {
                found = false;
                curcode =
V.line[linelist[i]].TRONCON_ROUTE.INSEE_COMD;
          j=1;
```

references to tables must be changed

element type references changed from lines to points

zoom scale changed to be reasonable for new data

text for prompts is changed

Adapted script for finding cities

(modifications shown in red)

```
# ToolScript looking for the street$ entered by the user
                    :
        for i=1 to nline {
                curcode =
V.point[linelist[i]].POareas.COUNTRY_OR;
                if (curcode == codelist[selpos,1]) {
                        nstreetline = nstreetline+1;
                        ResizeArrayPreserve(streetline,
nstreetline);
                        streetline[nstreetline] = linelist[i];
                }
        }
        ViewSetMessage(View, NumToStr(nstreetline) + " lines found
for this street");

        #Zoom in to the lines
        class VECTORLAYERPOINTS vll;
        vll = layer.Point;
        vll.HighlightSingle( ptlist[ selpos ] );
        View.DisableRedraw = 1;
        layer.ZoomToHighlighted();
        if (ViewGetMapScale(View) < 6000000) {
                ViewSetMapScale(View, 6000000);
        }
        View.DisableRedraw = 0;
        ViewRedraw(View);
}#DoZoom


# New Request
proc DoNew () {
        poslist.DeleteAllItems();
        nline = 0;
        ncode = 0;

        #asking to enter the name of a street (or a word contained in
it)
        street$ = PopupString("Enter the name of the city/town you
want to find", "");
        if (street$ == "") return;
        street$ = toupper$(street$);

        #looking for a line containing street$ in its NOM_RUE_D or
NOM_RUE_G attributes of the TRONCON_ROUTE table
        for i=1 to NumVectorPoints(V) {
                #class DATABASE DB = V.point[i].PPPOINT;
                if (V.point[i].PPPOINT.PPPTNAME$ contains
street$) {
                        nline = nline+1;
                        linelist[nline] = i;
                }#if
        }#i
        ViewSetMessage(View, NumToStr(nline) + " towns found");

        if (nline == 0) {       #no element corresponding found
                PopupMessage("No cities found that contain that
word!");
                return;
        }

        #Some streets are found : find the different ones (by zip code)
        #Assertion : not 2 streets with the same name in a town
        #Limits : don't take into account the streets separating 2
towns (the right zip code INSEE_COMD and the left one INSEE_COMG
are different)
        for i=1 to nline {
                found = false;
                curcode =
V.point[linelist[i]].POareas.COUNTRY_OR;
          j=1;
```

you would likely also change the comments to apply to the modified script, but it is not necessary