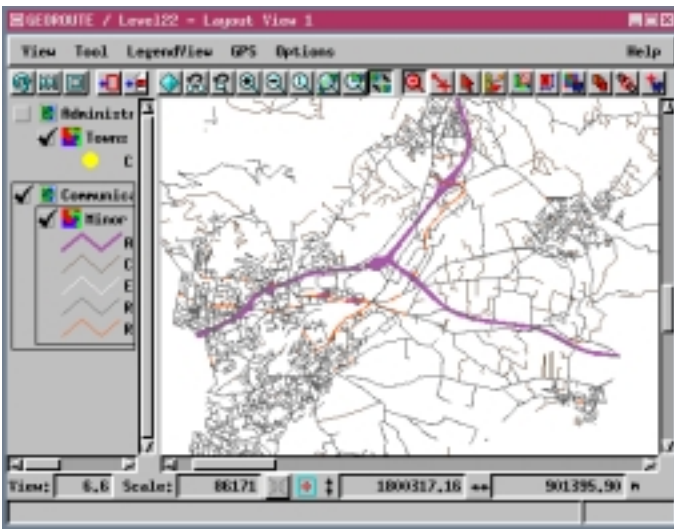


# Sample SML Tool Script

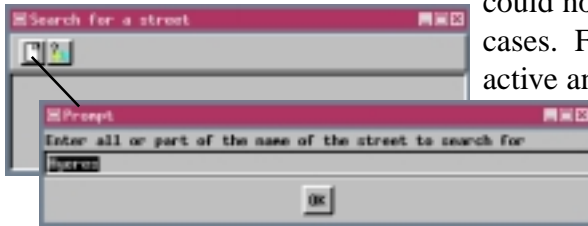
## Find Streets



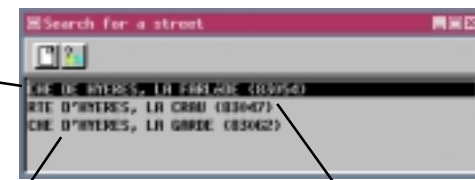
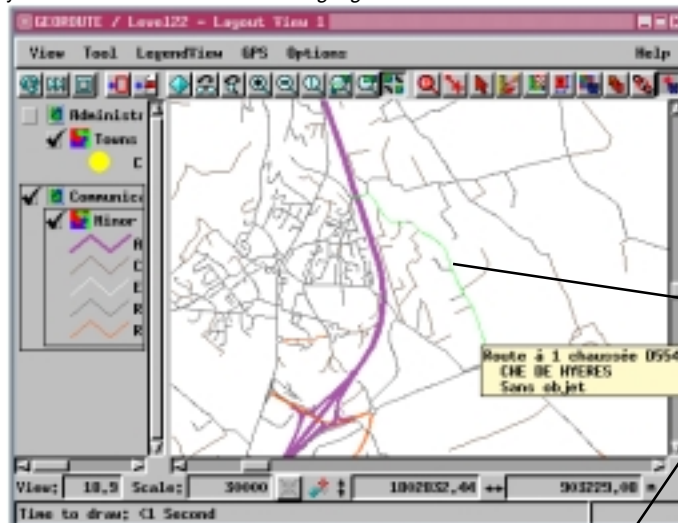
The Find Streets tool script locates and highlights streets you ask it to search for. You enter all or part of the street name to search for, and the tool script produces a list of all streets containing the text you entered. You then select the street you want to find and the script redraws the view at 1:30000 with the lines that form the street highlighted and centered in the View window. If all selected lines will not fit in the View at 1:30000, the View is redrawn at a smaller scale that fully contains the lines.

All lines that form the street are highlighted. The highlight colors for these lines are your designated selected and active element colors (Options / Colors). You may need to change these colors to take into account the drawing styles of the vector lines, which are purple, red, or black in this vector object. Thus, the default red highlight color could not be distinguished from unhighlighted lines in some cases. For the purposes of this tool, you may want to set the active and selected colors to be the same so that the selected street has a uniform appearance. The highlight color for the illustrations on this page is green.

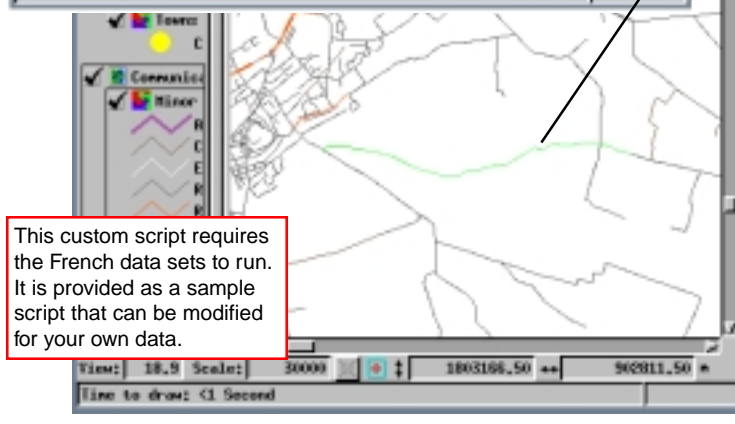
The New Search button opens a Prompt window so you can type in all or part of the name of the street you want to locate. Click OK, and all streets that match the search criteria are listed. Double-click on the name of the street you want to find or click on the *Highlight the street* icon.



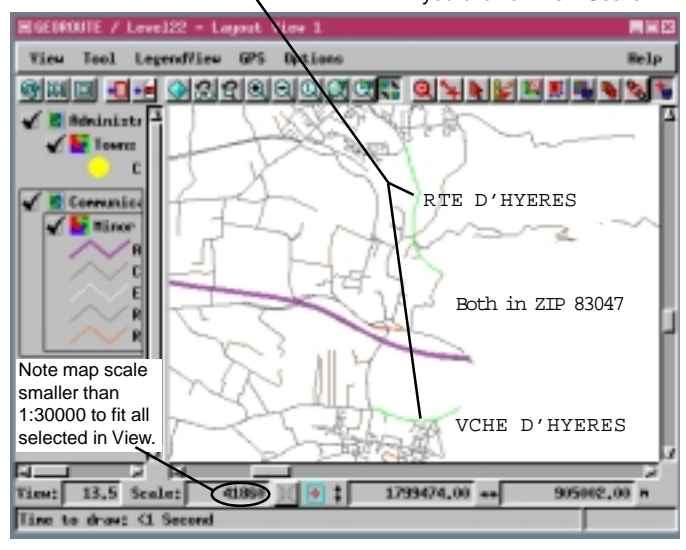
The name of the town and the zip code are also provided in the list of streets found. The script assumes there are not two separate streets in the same zip code with the same name. If, however, it turns out that the part of the street name you entered belongs to two different streets in the same zip code (one Main Street, the other Main Drive, for example), only the first encountered is listed but both are highlighted when that selection is made.



The script automatically pans to the selected street when you double click on the listing or click on the *Highlight the street* icon. The list is cleared when you click on New Search.



This custom script requires the French data sets to run. It is provided as a sample script that can be modified for your own data.



Note map scale smaller than 1:30000 to fit all selected in View.

Macro and Tool Scripts can be created using SML in any TNTmips process that uses a View window (Options / Customize from the View window menu bar). These scripts are then available from an icon, which you select or design, on the toolbar. Sample scripts have been prepared to illustrate how you might use these features, which are available only in TNTmips 6.4 or later, to assist with specific tasks you perform on a regular basis. If possible, the full script is printed below for your quick perusal. When a script is too long to fit on one page, key sections are reproduced below. The sample Tool Script illustrated can be downloaded from the SML script exchange at [www.microimages.com/sml/ftpsmlink/TNT\\_Products\\_V6.4\\_CD](http://www.microimages.com/sml/ftpsmlink/TNT_Products_V6.4_CD).

## Partial Script for Find Streets (street.sml)

```
# Zoom to selected position
proc DoZoom () {
  #Finding the element numbers of this street (not sure that they are sorted)
  local selpos;
  if (nline == 0) return;

  selpos = poslist.GetFirstSelectedPos();
  array streetline[1];
  nstreetline = 0;

  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 (View.GetMapScale(View) < 30000) {
    View.SetMapScale(View, 30000);
  }
  View.DisableRedraw = 0;
  View.Redraw(View);
}#DoZoom
```

zooms to  
selected  
elements

```
# 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 all or part of the name of the street to search for", "");
  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_DS contains street$ or
      V.line[i].TRONCON_ROUTE.NOM_RUE_G$ contains street$) {
      nline = nline+1;
      linelist[nline] = i;
    }#if
  }#if
  ViewSetMessage(View, NumToStr(nline) + " lines found");

  if (nline == 0) {
    #no element corresponding found
    PopupMessage("No streets found containing this 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.line[linelist[i]].TRONCON_ROUTE.INSEE_COMD;

    j=1;
    while (!found and j<=ncode) { #looks if code already found
      if (codelist[j,1] == curcode) {
        found = true;
      }#if
      j = j+1;
    }#while

    if (!found) { #new street
      ncode = ncode+1;
      codelist[ncode,1] = curcode;
      codelist[ncode,2] = linelist[i];
    }#if
  }#if
}#DoNew
```

clears list,  
prompts for  
new search  
street, and  
finds lines

reports total  
number of  
lines found by  
search

```
#Retrieve the table containing the names of the towns
array townnames[ncode]; #V.Town.point[townname[i]].ZONE_HABITAT.INSEE ==
  V.line[codelist[i,2]].TRONCON_ROUTE.INSEE_COMD;
npts = NumVectorPoints(V.Town);
ResizeArrayClear(townnames, ncode);

for i=1 to ncode {
  curcode = V.line[codelist[i,2]].TRONCON_ROUTE.INSEE_COMD;
  j = 1;
  found = false;
  townnames[i] = 0; #init
  while (!found and j<=npts) {
    if (V.Town.point[j].ZONE_HABITAT.INSEE == curcode) {
      townnames[i] = j;
      found = true;
    }
    j = j+1;
  }
  if (townnames[i] == 0) { #error
    PopupMessage("Town name corresponding to " +
      NumToStr(curcode) + " not found");
  }
}#i

for i = 1 to ncode {
  name$ = V.line[codelist[i,2]].TRONCON_ROUTE.NOM_RUE_DS;
  zip$ = "(" + NumToStr(codelist[i,1]) + ")";
  town$ = " " + street$ =
    toupper$(V.Town.point[townnames[i]].ZONE_HABITAT.TOPONYMES);
  poslist.AddItem(name$ + town$ + zip$);
}
poslist.SelectPos(1);
}#DoNew
```

#DoNew

# Close the window, switching to default tool

```
proc DoClose () {
  if (setDefaultWhenClose) {
    setDefaultWhenClose = false;
    View.SetDefaultTool();
  }
}
```

closes search  
window and acti-  
vates default tool

# Called the first time the tool is activated.

```
func OnInitialize () {
  dlgform = CreateFormDialog("Search for a street", View.Form);
  WidgetAddCallback(dlgform.Shell.PopdownCallback, DoClose);
  dlgform.Width = 200;
```

creates  
search  
dialog

```
class PushButtonItem btnItemNew;
class PushButtonItem btnItemZoom;
btnItemNew = CreatePushButtonItem("New search", DoNew);
btnItemNew.IconName = "new";
btnItemZoom = CreatePushButtonItem("Highlight the street", DoZoom);
btnItemZoom.IconName = "apply_query";
btnItemClose = CreatePushButtonItem("Close", DoClose);
btnItemClose.IconName = "delete";
```

# Icon button rows are automatically attached to their parent form on the left  
# and right. The "right" widget is unattached by setting the attachment to itself.

```
class XmRowColumn btnrowaction;
btnrowaction = CreateIconButtonRow(dlgform, btnItemNew, btnItemZoom);
btnrowaction.TopWidget = dlgform;
btnrowaction.TopOffset = 4;
btnrowaction.LeftWidget = btnrowaction;
btnrowaction.LeftOffset = 8;
btnrowaction.RightOffset = 4;
```

#

```
class XmSeparator btnsep;
btnsep = CreateHorizontalSeparator(dlgform);
btnsep.TopWidget = btnrowaction;
btnsep.TopOffset = 4;
btnsep.LeftWidget = dlgform;
btnsep.RightWidget = dlgform;
```

```
poslist = CreateScrolledList(dlgform);
```

(see street.sml for full script)