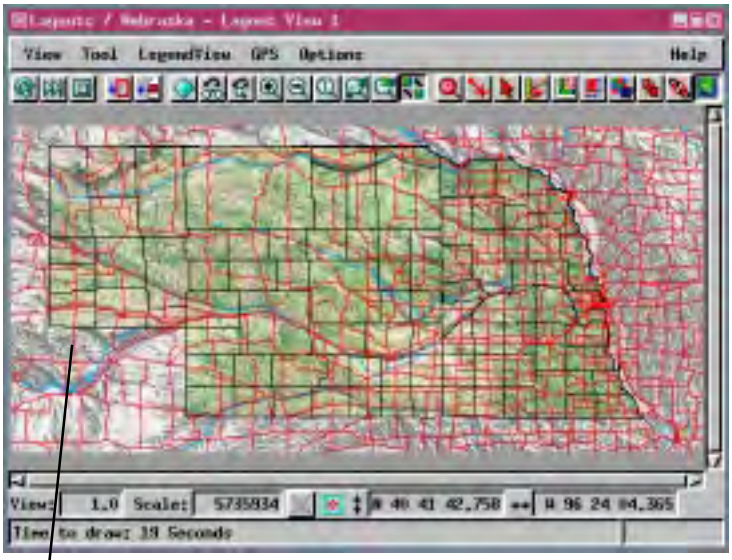
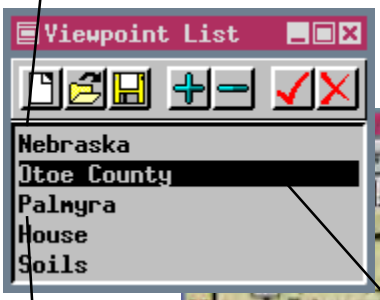


Sample SML Tool Script

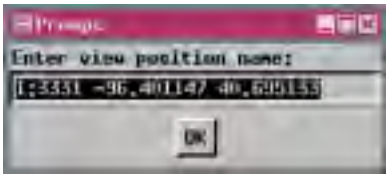
ViewMarks

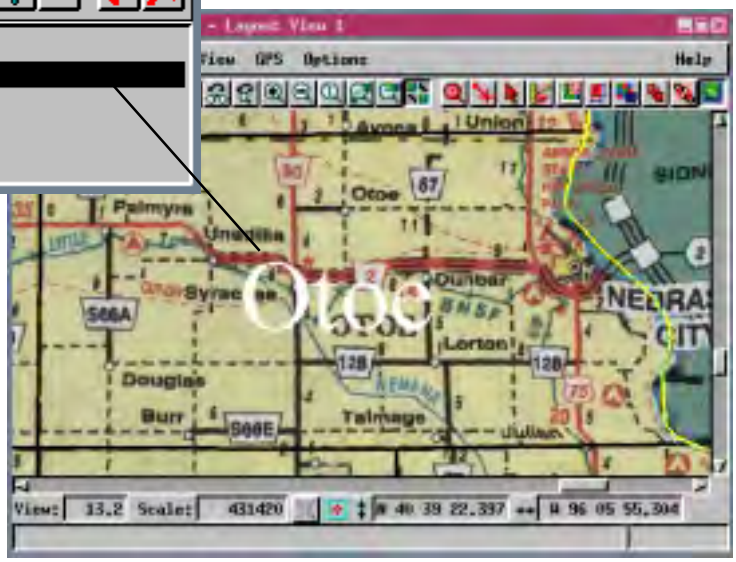
ViewMarks are position markers for a single view window. They are particularly useful for layouts covering a large geographic area, especially when limited map scale visibility is used to add and remove layers as you zoom in and out. Mark a view of interest and return to that view from any scale or position by selecting it from the list of viewpoints you build up.

The script, a portion of which is shown on the other side of this page, creates the Viewpoint List window (below, left) with the buttons needed to make, save, and open viewpoint lists; to add and remove points from the list; and to zoom to the selected point and close the window. You can also double-click on a list entry to display it.

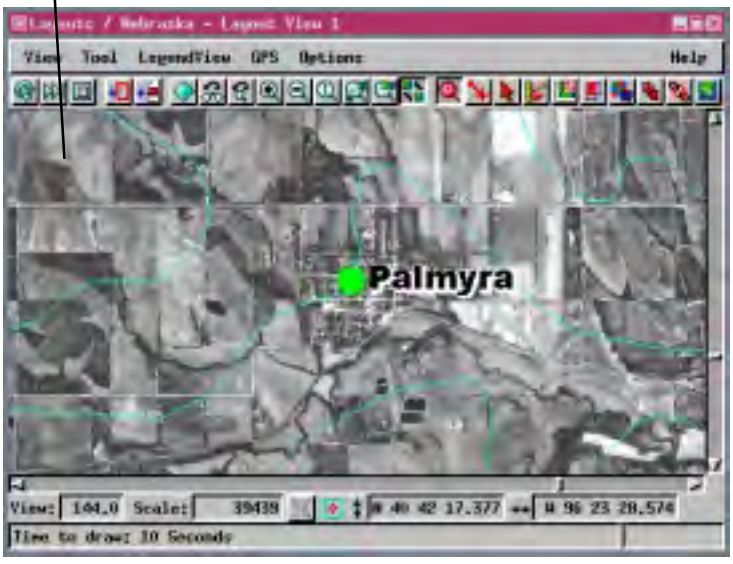


When you add a ViewMark, the default name provided is the map scale and center coordinates for the view, which can be changed to a more descriptive name.





The Viewpoint List remains as long as the current View window is open. If you want to use the same ViewMarks in another display session, you need to save the list. When you choose to save your viewpoint list, a .pos file is created. This file simply contains the name you entered for the viewpoint, the map scale, and center point for each viewpoint on the list. Thus, the file can be used again with the same group, layout, or single-layout atlas, or it can be used with a completely different set of layers that covers the same geographic area.



Thus, ViewMarks let you work with data sets that cover large areas and still rapidly locate and return to areas of interest in high resolution imagery or detailed vector objects. ViewMarks have use beyond TNTmips' Display process; they can be set up for any process that uses a View window, for example, the Spatial Data Editor. Thus, you can mark a number of positions that are critical to check after some global editing operation, such as line snapping or filtering, and return to each with ease.

The example on this page is derived from the Nebraska Statewide atlas, which is a single layout that uses map scale controlled visibility to increase the level of detail shown as you zoom in.

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. All sample Tool and Macro Scripts illustrated can be found in their entirety on your TNT products CD-ROM in the directory where your TNT products are installed. These scripts, among others, can be downloaded from the SML script exchange at www.microimages.com/sml/ftpsmllink/TNT_Products_V6.4_CD.

Partial Script for ViewMarks (vptool.sml)

```
class XmForm dlgform;
class XmList poslist;
class MAPPROJ projLatLon;
class TRANSPARM transMapToView;
class FILE posfile;
number ischanged;
number setDefaultWhenClose;
number numpos;
array posX[1];
array posY[1];
array posScale[1];

func DoSave () {
    if (numpos == 0) return;
    posfilename$ = GetOutputFileName("","Select position file to save as:","pos");
    DeleteFile(posfilename$);
    posfile = fopen(posfilename$,"w");
    if (posfile == 0) return (false);
    local i;
    for i = 1 to numpos {
        fprintf(posfile,"%s,%f,%f,%f\n",poslist.GetItemAtPos(i),posX[i],posY[i],posScale[i]);
        }
    fclose(posfile);
    ischanged = false;
    return (true);
    }
func AskSave () {
    if (!ischanged || numpos == 0) return (true);
    local answer;
    answer = PopupYesNoCancel("Save current point list?",1);
    if (answer < 0) return (false);
    if (answer == 0) return (true);
    return (DoSave());
    }
proc DoZoom () {
    local selpos;
    if (numpos == 0) return;
    selpos = poslist.GetFirstSelectedPos();
    if (selpos > 0) {
        transMapToView = ViewGetTransMapToView(View,projLatLon);
        if (transMapToView == 0) {
            PopupMessage("Cannot obtain map/view transformation.");
            return;
            }
        class POINT2D zpoint;
        zpoint.x = posX[selpos];
        zpoint.y = posY[selpos];
        zpoint = TransPoint2D(zpoint,transMapToView,false);
        class RECT vextents;
        vextents = View.Extents;
        if (zpoint.x < vextents.x1 || zpoint.x > vextents.x2 || zpoint.y < vextents.y1 || zpoint.y > vextents.y2) {
            PopupMessage("Point is outside extents of objects being viewed.");
            return;
            }
        View.DisableRedraw = true;
        View.CurrentMapScale = posScale[selpos];
        View.Center = zpoint;
        View.DisableRedraw = false;
        ViewRedraw(View);
        }
    }
proc DoAdd () {
    transMapToView = ViewGetTransMapToView(View,projLatLon);
    if (transMapToView == 0) {
        PopupMessage("Cannot obtain map/view transformation.");
        return;
        }
    class POINT2D cpoint;
    cpoint = TransPoint2D(View.Center,transMapToView,true);
    numpos = numpos + 1;
    ResizeArrayPreserve(posX,numpos);
    ResizeArrayPreserve(posY,numpos);
    ResizeArrayPreserve(posScale,numpos);
    posX[numpos] = cpoint.x;
    posY[numpos] = cpoint.y;
    posScale[numpos] = View.CurrentMapScale;
    namestr$ = sprintf("1:%.0f %f %f",posScale[numpos],posX[numpos],posY[numpos]);
    namestr$ = PopupString("Enter view position name:",namestr$);
    while (poslist.ItemExists(namestr$)) {
        namestr$ = PopupString("Name already used.\nEnter view position name:",namestr$);
        }
    poslist.AddItem(namestr$);
    ischanged = true;
    }
proc DoRemove () {
    local selpos;
    local i;
    if (numpos == 0) return;
    selpos = poslist.GetFirstSelectedPos();
    if (selpos > 0) {
        poslist.DeletePos(selpos);
        for i = selpos to numpos - 1 {
            posX[i] = posX[i+1];
            posY[i] = posY[i+1];
            posScale[i] = posScale[i+1];
            }
        numpos = numpos - 1;
        ischanged = true;
        }
    }
proc DoNew () {
    if (!AskSave()) return;
    numpos = 0;
    poslist.DeleteAllItems();
    ischanged = false;
    }
proc DoOpen () {
    if (!AskSave()) return;
    posfile = GetInputTextFile("","Select positions file to open:","pos");
    if (posfile == 0) return;
    numpos = 0;
    poslist.DeleteAllItems();
    ischanged = false;
    while (!feof(posfile)) {
        filestr$ = fgetline$(posfile);
        if (NumberTokens(filestr$,",") < 4) continue;
        numpos = numpos + 1;
        ResizeArrayPreserve(posX,numpos);
        ResizeArrayPreserve(posY,numpos);
        ResizeArrayPreserve(posScale,numpos);
        poslist.AddItem(GetToken(filestr$,",",1));
        posX[numpos] = StrToNum(GetToken(filestr$,",",2));
        posY[numpos] = StrToNum(GetToken(filestr$,",",3));
        posScale[numpos] = StrToNum(GetToken(filestr$,",",4));
        }
    fclose(posfile);
    }
proc DoClose () {
    if (setDefaultWhenClose) {
        setDefaultWhenClose = false;
        View.SetDefaultTool();
        }
    }
func OnInitialize () {
    class MAPPROJ tempLatLon;
    tempLatLon.System = "LatLon";
    tempLatLon.Datum = "WGS84";
    projLatLon = tempLatLon;
    dlgform = CreateFormDialog("Viewpoint List",View.Form);
    WidgetAddCallback(dlgform.Shell.PopdownCallback,DoClose);
    class PushButtonItem btnItemNew;
    class PushButtonItem btnItemOpen;
    class PushButtonItem btnItemSave;
    class PushButtonItem btnItemAdd;
    class PushButtonItem btnItemRemove;
    class PushButtonItem btnItemZoom;
    class PushButtonItem btnItemClose;
    btnItemNew = CreatePushButtonItem("New",DoNew);
    btnItemNew.IconName = "new";
    btnItemOpen = CreatePushButtonItem("Open...",DoOpen);
    btnItemOpen.IconName = "open_";
    btnItemSave = CreatePushButtonItem("Save...",DoSave);
    btnItemSave.IconName = "save";
    btnItemAdd = CreatePushButtonItem("Add",DoAdd);
    btnItemAdd.IconName = "add_sel";
    btnItemRemove = CreatePushButtonItem("Remove",DoRemove);
    btnItemRemove.IconName = "remove_sel";
    btnItemZoom = CreatePushButtonItem("Zoom",DoZoom);
    btnItemZoom.IconName = "apply";
    btnItemClose = CreatePushButtonItem("Close",DoClose);
    btnItemClose.IconName = "delete";
```

saves the list to a file

zooms to selected position

adds current viewpoint to list

removes selected item from list

clears the list

opens file containing list

closes the window and switches to default tool

is called the first time the tool is activated

(see vptool.sml for full script)