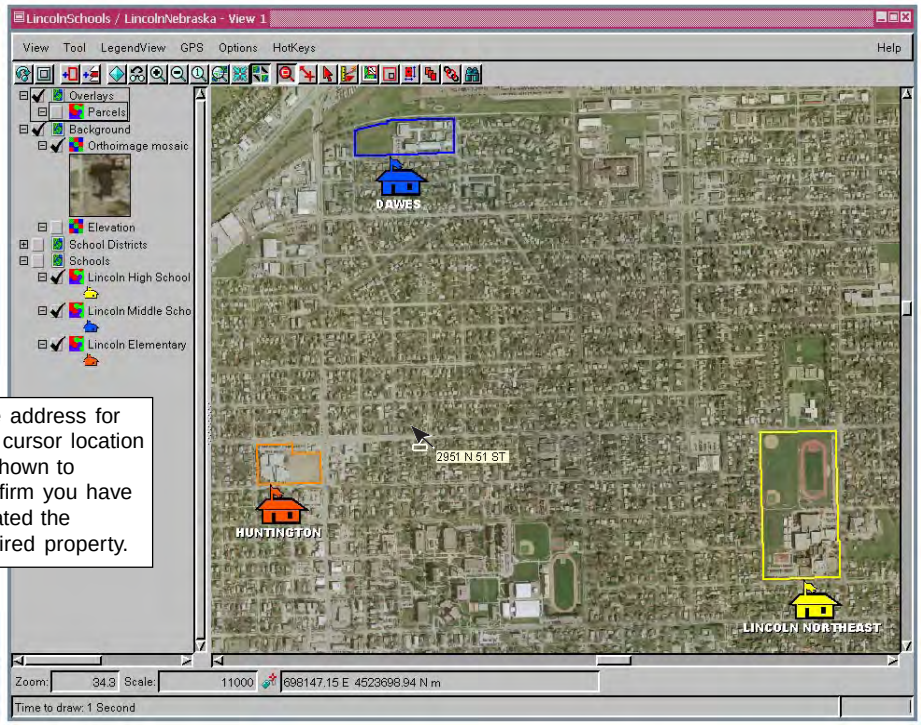
Dynamic Geospatial Analysis

Exploring District Services

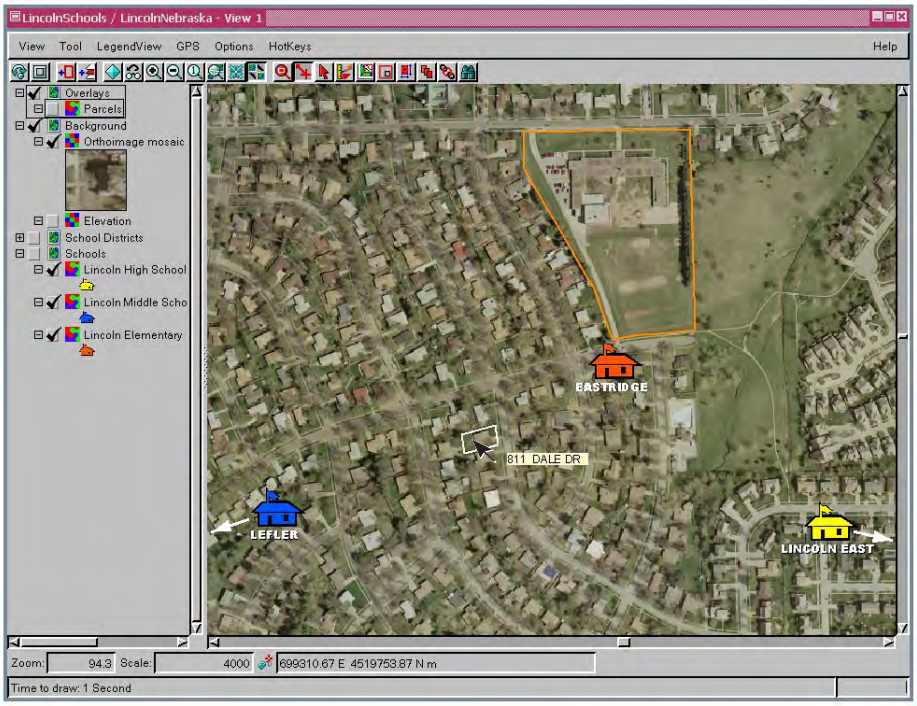
A common geospatial question is "Given my location, what facilities are closest to me?" Facilities could mean lodging, restaurants, truck stops, fire stations, polling places, or a myriad of other facility types. A related question is "What roaming vehicles (for example, cement trucks, tow trucks, express pickup) are within a particular service area (represented by a polygon)?" "Closeness" can be determined in various ways, such as overlapping polygons, shortest routes, fastest routes, least cost routes, and so on. These kinds of geospatial questions can be resolved and visualized in the TNT products using a variety of approaches (Tool Scripts, Macro Scripts, SML, queries, APPLIDATs, routing, or network analysis). Control Scripts can provide highly interactive and automatic tools for geoexploring using dynamic feature location.



The address for the cursor location is shown to confirm you have located the desired property.

"Automatic" indicates that these "feature location" analyses and resulting visualization all occur automatically in less than a second each time the cursor is moved—no mouse click or other selection procedure is necessary.

This example of automatic feature location uses a Control Script to draw the outlines and symbolic representations of the elementary, middle, and high schools in the attendance area assigned to the land parcel at the current cursor location. The Control Script finds and outlines the land parcel beneath the cursor and identifies it with a DataTip address. It then determines the three school attendance area polygon boundaries that include the current land parcel and draws the three school property boundaries and adds a school type symbol and name beneath each.



At map scales less than 1:50,000, the school symbol is centered over the school property and the property outline is not drawn. At map scales of 1:50,000 or greater, the school parcel outline is drawn and the symbol and school name label are positioned below the outline (illustrated above). When the location of the nearest school in any grade level grouping is not visible in the View, a symbol is positioned at the closest edge of the View in a straight line from the cursor location to the school with an arrow indicating the direction of the school [shown at left for middle and high schools (blue and yellow, respectively)].

Many sample scripts have been prepared to illustrate how you might use the features of the TNT products' scripting language for scripts and queries. These scripts can be downloaded from www.microimages.com/freestuf/scripts.htm.

Partial Script for Exploring District Services (schools.sml)

see back of plate entitled *Using Overlapping Polygons* for more of this script

```
func OnViewDataTipShowRequest(class GRE_VIEW view, class
        POINT2D mouseScrn, class TOOLTIP datatip)
{                                          function called when cursor
  view.RestoreAll();                       pause triggers DataTip
  class GC gc = CreateGCForDrawingArea(view.DrawingArea);
  ActivateGC(gc);
  gc.SetColor("white");
  class STRING name = "Schools";
  class GRE_LAYER_VECTOR curPtLayer =
        MainLayout.GetGroupByName("Schools").FirstLayer;
  class GRE_LAYER_VECTOR curDistLayer =
        MainLayout.GetGroupByName("School Districts").FirstLayer;
  class GRE_LAYER_VECTOR parcels =
        MainLayout.GetGroupByName("Overlays").GetLayerByName("Parcels");
  class VECTOR parcelVec;
  VectorLayerGetObject(parcels, parcelVec);
  class TRANSPARM tpPrclToScrn = ViewGetTransLayerToScreen(view,
        parcels);
  class POINT2D mousepoint;
  mousepoint = TransPoint2D(mouseScrn,          draw parcel under
        ViewGetTransViewToScreen(view, 1));     mouse pointer if map
  mousepoint = TransPoint2D(mousepoint,         scale appropriate
        ViewGetTransMapToView(view, parcels.Projection, 1));
  numeric closeParcel = FindClosestPoly(parcelVec, mousepoint.x,
        mousepoint.y);
  if(closeParcel != 0 && view.CurrentMapScale < mouselocPrclScl) {
    class RECT parcelExtents = drawParcel(
      gc, tpPrclToScrn, parcelVec, closeParcel);   draw DataTip
    class string parcelAddress =                   for parcel
        parcelVec.Poly[closeParcel].CA032904.SITUS_ADDR$;
    gc.DrawTextSetHeightPixels(12);
    numeric addressLen = gc.TextGetWidth(parcelAddress);
    gc.SetColor("Lemon Chiffon");
    gc.FillRect(parcelExtents.x2 + 10, parcelExtents.y2, addressLen, 12);
    gc.DrawTextSetFont("Arial");
    gc.DrawTextSimple(parcelAddress, parcelExtents.x2 + 10,
        parcelExtents.y2 + 11);
  }
  class STRINGLIST parcelColors;
  parcelColors.AddToFront("Yellow");          set parcel
  parcelColors.AddToFront("Blue");            boundary colors
  parcelColors.AddToFront("Dark Orange");
  numeric iterNum=1;
  for(iterNum = 1; curPtLayer != 0 && curDistLayer != 0; iterNum++) {
    class VECTOR ptsVect, distVect;
    VectorLayerGetObject(curPtLayer, ptsVect);
    VectorLayerGetObject(curDistLayer, distVect);
    class string styleObj = "STYLE";
    gc.DrawUseStyleSubObject(_context.Filename, ptsVect.$info.Name,
        styleObj);
    class POINT2D schoolLoc, mousepoint;
    numeric pointElemNum;                       draw school
    mousepoint = TransPoint2D(mouseScrn,        point symbols
        ViewGetTransViewToScreen(view, 1));
    mousepoint = TransPoint2D(mousepoint,
        ViewGetTransMapToView(view, curDistLayer.Projection, 1));
  if((pointElemNum = FindPointInPolyElemNum(ptsVect, distVect,
      mousepoint)) != -1) {
    schoolLoc.x = ptsVect.Point[pointElemNum].Internal.x;
    schoolLoc.y = ptsVect.Point[pointElemNum].Internal.y;
    schoolLoc = TransPoint2D(schoolLoc,
      ViewGetTransMapToView(view, curDistLayer.Projection));
    schoolLoc = TransPoint2D(schoolLoc,
      ViewGetTransViewToScreen(view));
    class RECT extents;
    numeric buf=5;
    extents.x1 = buf; extents.y1 = buf; extents.x2 = view.width-buf;
      extents.y2 = view.height-buf;
    class POINT2D newLocation = clipPoint(schoolLoc, mouseScrn,
      extents);
    class TEXTSTYLE textStyle;                  draw school name
    textStyle.Shadow = 1;                       below symbol
    gc.TextStyle = textStyle;
    gc.DrawTextSetColors("white", "black");
    gc.DrawTextSetFont("Arial Black");
    gc.DrawTextSetHeightPixels(12);
    class string schoolName =
      ptsVect.Point[pointElemNum].SchoolName.Name$;
    schoolName = schoolName.toUppercase();
    numeric textWidth = gc.TextGetWidth(schoolName);
    gc.DrawSetPointStyle("School");
    if(newLocation.x != schoolLoc.x || newLocation.y != schoolLoc.y) {
      gc.SetColor("white");
      numeric angle = calcAngle(schoolLoc, mouseScrn);
      gc.DrawArrow(newLocation.x, newLocation.y, angle * 180 / PI,
      20, 20);                                  draw arrow if
      gc.SetLineWidth(3);                       school not in View
      drawAngledLine(gc, newLocation.x, newLocation.y, angle, 40);
      gc.SetLineWidth(1);
      drawAngledPoint(gc, newLocation.x, newLocation.y, angle, 70);
      class POINT2D textpos = getAngledPoint(newLocation.x,
      newLocation.y, angle, 70);
      gc.DrawTextSimple(schoolName, textpos.x - textWidth/2, textpos.y
      + labelOffset);
    } else {
      if(view.CurrentMapScale < schoolPrclScl) {
        class POINT2D schoolLocParcel = TransPoint2D(schoolLoc,
      ViewGetTransViewToScreen(view, 1));
        schoolLocParcel = TransPoint2D(schoolLocParcel,
      ViewGetTransMapToView(view, parcels.Projection, 1));
        numeric schoolPolyID = FindClosestPoly(parcelVec,
      schoolLocParcel.x, schoolLocParcel.y);
        gc.SetColor(parcelColors.GetString(iterNum-1));
        class RECT parcelExtents = drawParcel(gc, tpPrclToScrn,
      parcelVec, schoolPolyID);
        class POINT2D screenCenter;
        screenCenter.x = view.Width/2;
        screenCenter.y = view.Height/2;
        numeric yPointPos = parcelExtents.y2 + schoolSymOffset;
        gc.DrawPoint(newLocation.x, yPointPos);
        gc.DrawTextSimple(schoolName, newLocation.x - textWidth/2,
      yPointPos + labelOffset);
      } else {
```

draw school parcel if map scale appropriate

last 11 lines of script omitted