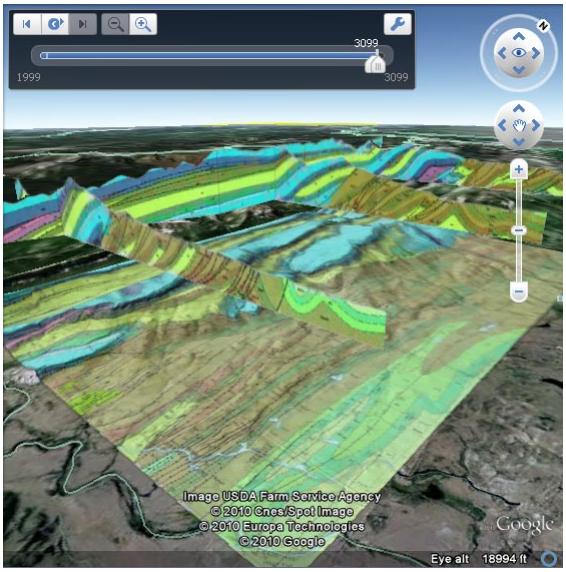
Sample Script

Export 3D Cross-Sections to Google Earth

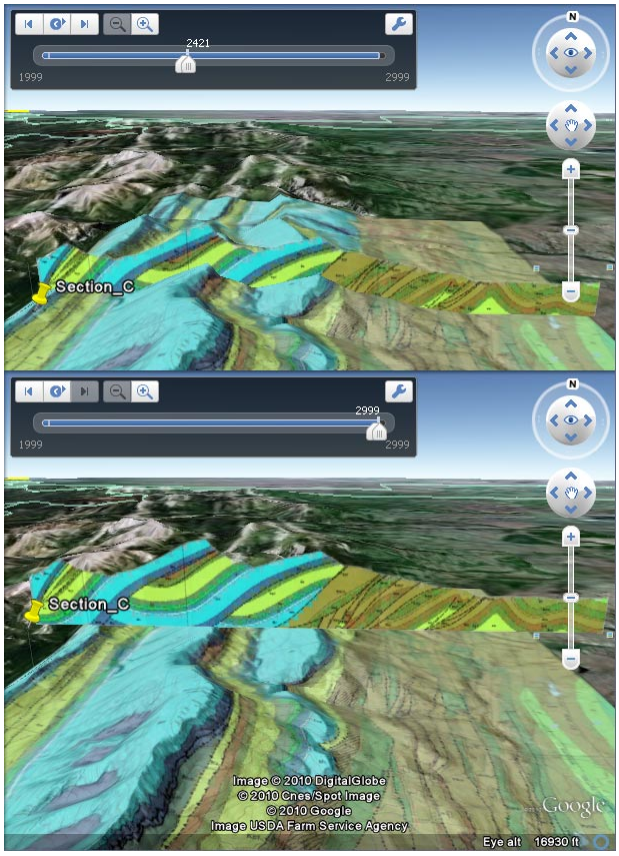
Geologic cross-sections and seismic profiles can be georeferenced with 3D control points in TNTmips to allow these objects to be displayed in 3D in their correct positions and orientations. The resulting *manifold* georeference includes 3D control points connected to form a triangular mesh to define a projective surface (see the Technical Guides entitled *Georeferencing Manifold Surfaces* and *Manifolds in 3D Views*). MicroImages has created a script using the TNT geospatial scripting language (SML) that automates the export of one or more manifold 3D cross-section objects to a KMZ file. This allows the section or sections to be viewed immediately in 3D in Google Earth in their correct geographic positions and orientations. This script, ExportMultiSectColladaKMZ.sml, is excerpted on the reverse of this page.

Google has adopted the COLLADA file format (http:www.khronos.org/collada/) as the medium to allow custom 3D models to be viewed in Google Earth. A COLLADA file representing a geologic cross-section specifies the 3D projective surface (triangulated position mesh constructed from the TNT 3D control points), references the section image (a PNG image file), and contains a mapping of these image coordinates to the vertices in the position mesh. A KML file is used to provide Google Earth with the geographic position and orientation of the COLLADA models. The sample SML script creates a COLLADA file and corresponding PNG image file for each selected cross-section, creates the KML file that positions these models, and packages all of these products in a single KMZ file (ZIP file with a .kmz file extension) for easy use in Google Earth.

Both COLLADA and KML files are XML structured text. SML includes classes that make it easy to read, modify, and write XML-formatted text, as shown in the script excerpts on the reverse. The sample script contains template COLLADA and KML files as text strings; these templates are read into memory and modified as needed before being written to the output. SML also includes classes to access control point coordinates and information on hard edges and triangulation boundaries. This information allows the manifold triangulation to be faithfully recreated in the COLLADA file.



Google Earth view of four geologic cross-sections exported together using the sample SML script, along with a geologic map tileset produced separately in TNTmips. The script packages the section image files, COLLADA models, and KML code for all of the exported cross-sections within one KMZ file.



The KMZ files produced by the SML script described here include KML animation code to raise the cross-sections from their subsurface position (where they are hidden below Google Earth's terrain) to a position above the terrain surface. The top illustration shows the Google Earth view with a single geologic cross-section partially raised and still intersecting the terrain, while the bottom illustration shows the same section raised to its highest position. A custom geologic map tileset produced separately in TNTmips is also shown in this view. The animation control automatically provided by Google Earth lets the user "play" the animation or drag the slider to manually position the sections at any intermediate elevation.

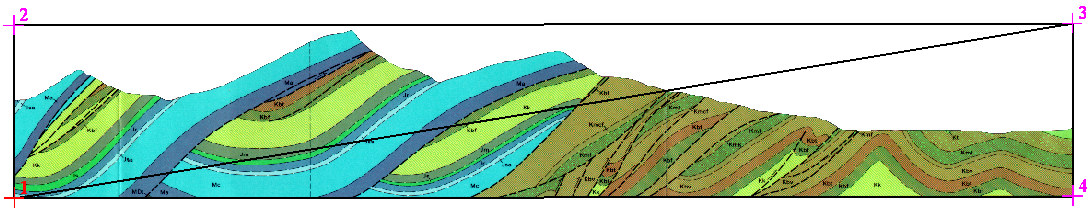
Google Earth does not allow viewing of objects below the Google Earth terrain surface (other than by toggling off the terrain, flattening it to sea level). In order to allow viewing of the COLLADA cross-section models with the Google Earth terrain, the SML script creates KML time-space animation code that allows the user to interactively raise the sections above the terrain from their natural subsurface positions. The script automatically analyzes the elevations of all cross-section control points to set the animation elevation range so that all sections are raised in concert and the maximum height raises all section bases clear of the Google Earth terrain.

This sample script is available for download along with the following sample products illustrated on this page:

Geologic map Google Earth Super Overlay: CRgeolmap.zip
Single-section KMZ file illustrated above: SectionC.kmz
Multiple-section KMZ file illustrated to the left: CastleReefSections.kmz

www.microimages.com/downloads/smlscripts.htm

Georeference Input Object view of a geologic cross-section raster with manifold georeference. The SML script described here processes the manifold control points to recreate this triangulation in a COLLADA file for 3D display in Google Earth.



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/downloads/scripts.htm.

Excerpts from ExportMultiSectColladaKMZtrack.sml

parse the COLLADA file template and get the root node

```
err = daedoc.Parse(dae$);
if (err < 0) { PopupError(err); break; }

colladaRootNode = daedoc.GetRootNode();
```

set created and modified dates, units, and scale factor

```
node = colladaRootNode.FindChild("asset");
node1 = node.FindChild("created");
node1.SetText(dt$);
node1 = node.FindChild("modified");
node1.SetText(dt$);

node1 = node.FindChild("unit");
node1.SetAttribute("name", unit$);
node1.SetAttribute("meter", sprintf("%.4f", unitConvToMeters) );
```

set the output PNG file as the library image

```
node = daedoc.GetElementByID("Xsect-image");
node1 = node.FindChild("init_from");
node1.SetText(pngFilepath.GetName() );
```

set the position mesh count and array

```
arraycount = numPts * 3;
node = daedoc.GetElementByID("mesh1-geometry-position-array");
node.SetAttribute("count", NumToStr(arraycount) );
node.SetText(xyzString);
```

set the count for the position array accessor

```
node1 = node.GetParent();
node2 = node1.FindChild("technique_common");
node3 = node2.FindChild("accessor");
node3.SetAttribute("count", NumToStr(numPts) );
```

set the texture mesh count and array

```
arraycount = numPts * 2;
node = daedoc.GetElementByID("mesh1-geometry-uv-array");
node.SetAttribute("count", NumToStr(arraycount) );
node.SetText(uvString);
```

set the count for the uv array accessor

```
node1 = node.GetParent();
node2 = node1.FindChild("technique_common");
node3 = node2.FindChild("accessor");
node3.SetAttribute("count", NumToStr(numPts) );
```

set the material and primitive for the triangles element

```
node = daedoc.GetElementByID("mesh1-geometry");
node1 = node.FindChild("mesh");
node2 = node1.FindChild("triangles");
node2.SetAttribute("count", NumToStr(numTriangles) );

node3 = node2.FindChild("p");
node3.SetText(pString);
```

write out the COLLADA file

```
class STRING daefile$ = dir$ + "/" + xsecName$ + ".dae";
printf("\nDAE filepath = %s\n", daefile$);
daedoc.Write(daefile$);

class FILEPATH daeFilepath(daefile$);
```

NOTE: Two versions of the cross-section export script are available. They differ in the method used to animate the cross-sections and limit use of the exported KMZ file to specific Google Earth versions:

ExportMultiSectColladaKMZ.sml:
KMZ useable in all Google Earth versions

ExportMultiSectColladaKMZtrack.sml:
KMZ file useable in Google Earth 5.2 and later.
Smoother animation using <gx:Track> element.

add info to the KML structure for this section

create <Document> element for this section in <Folder>

```
document = folder.NewChild("Document");
node = document.NewChild("name");
node.SetText(xsecName$);
```

create <Placemark> for the cross-section with name

```
placemark = folder.NewChild("Placemark");
placemark.NewChild("name", xsecName$);
```

create <gx:Track> element for <Placemark> and set <altitudeMode>

```
track = placemark.NewChild("gx:Track");
track.NewChild("altitudeMode", "absolute");
```

create <Model> element in <gx:Track>

```
model = track.NewChild("Model");
model.SetAttribute("id", sprintf("model%d", k) );
```

create <Orientation> element for <Model>

```
node = model.NewChild("Orientation");
node.NewChild("heading", NumToStr(angleToNorth) );
node.NewChild("tilt", "0");
node.NewChild("roll", "0");
```

create <Scale> element for <Model>

```
node = model.NewChild("Scale");
node.NewChild("x", "1.0");
node.NewChild("y", "1.0");
node.NewChild("z", "1.0");
```

create <Link> element for <Model>

```
node = model.NewChild("Link");
node.NewChild("href", "files/" + xsecName$ + ".dae");
```

create <ResourceMap> element for <Model>

```
node = model.NewChild("ResourceMap");
node1 = node.NewChild("Alias");

node1.NewChild("targetHref", "files/" + xsecName$ +".png");
node1.NewChild("sourceHref", "files/" + xsecName$ +".png");
```