DIGITAL MAPPING TECHNIQUES 2015

The following was presented at DMT'15
(May 17-20, 2015 - Utah Geological Survey,
Salt Lake City, UT)

The contents of this document are provisional

See Presentations and Proceedings
from the DMT Meetings (1997-2015)

http://ngmdb.usgs.gov/info/dmt/

Building 3D PDFs to Visualize Geological Data

Daniel W. Eungard and David A. Jeschke

Digital Mapping Techniques Conference 2015

Primary contact: Daniel Eungard
Washington State Department of Natural Resources (DNR)
Phone: 360-902-1463
Email: Daniel.Eungard@dnr.wa.gov

Introduction

Geologic maps traditionally consist of 2D products on paper, or more recently in digital format as a pdf. These maps typically contain geologic cross sections that visually convey topographic and subsurface conditions to the reader. Proper interpretation of these map products can be difficult and potentially cause confusion and misinterpretation. Our goal is to help the readers avoid such confusion by displaying the geologic information in 3D space. To accomplish this, we have developed a method to take the published information and display it spatially using ESRI ArcMap, ArcScene, and PDF3D ReportGen to create a model which is viewable in Adobe Reader/Viewer.

Methodology

Originally, a geologist creates polygons for the geologic map and cross sections in ArcMap at the correct spatial location and the projection origin respectively. The plate for the publication (Fig 1.) is derived from the export of these polygons to Adobe Illustrator. Finishing touches are then applied in Illustrator. The finished geologic map is georeferenced back into ArcMap; however, the cross sections pose a larger challenge. In ArcMap, the cross section displays in map view with X-axis representing the length of cross section and Y-axis represents the vertical dimension (Z). The line of cross section on the map represents both X and Y coordinates. These sets of coordinate values need to be translated so that Y becomes Z from the cross section and individual X and Y coordinates are generated from the line of cross section. Once translated, the cross section will display properly in ArcScene.

ArcScene provides a method of displaying polygons in 3D by reading XY values from feature geometry and Z values from attribute data. In the example provided (Fig. 2), a buffered borehole is “extruded” with the top and bottom elevation of each geologic layer stored as attribute data. Each boring depicted derives from multiple polygons (one for each layer) that are stacked forming a visually continuous cylinder. We can use the same method here to show the geologic units in the cross section as a series of extruded polygons.

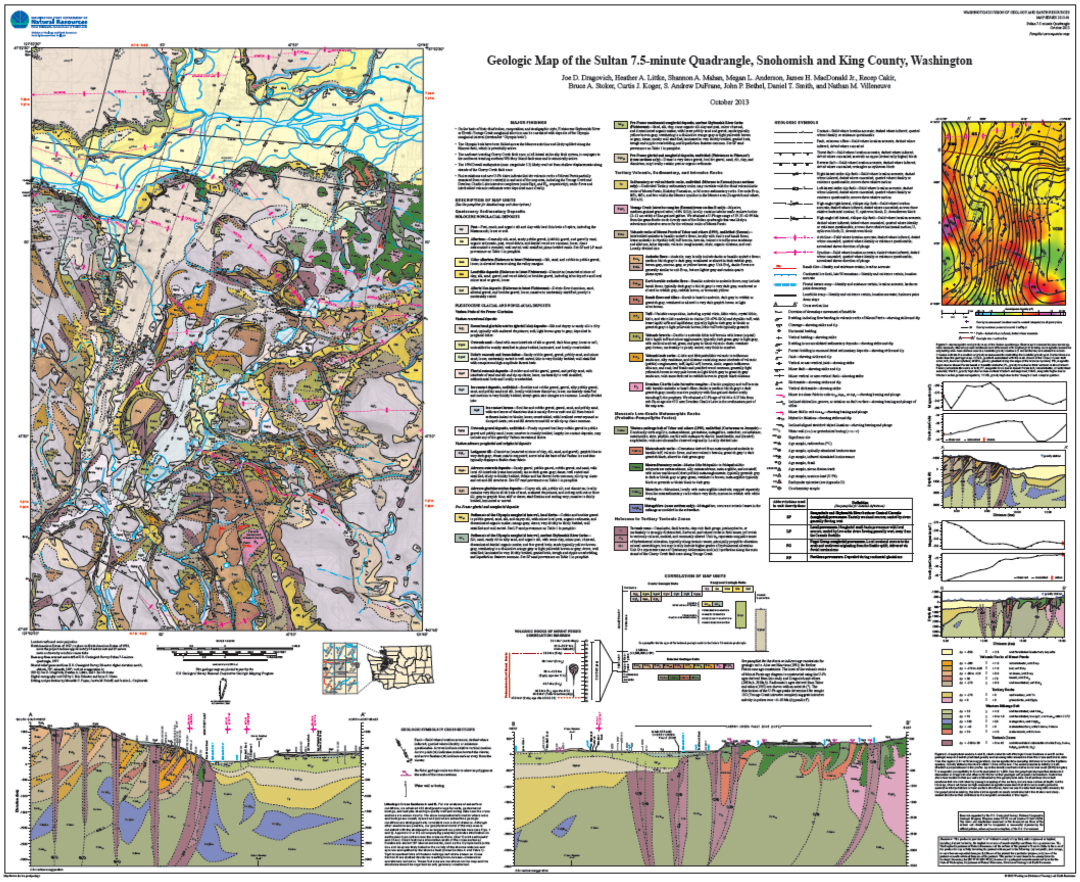


Figure 1 *- Plate of geologic map and cross sections*

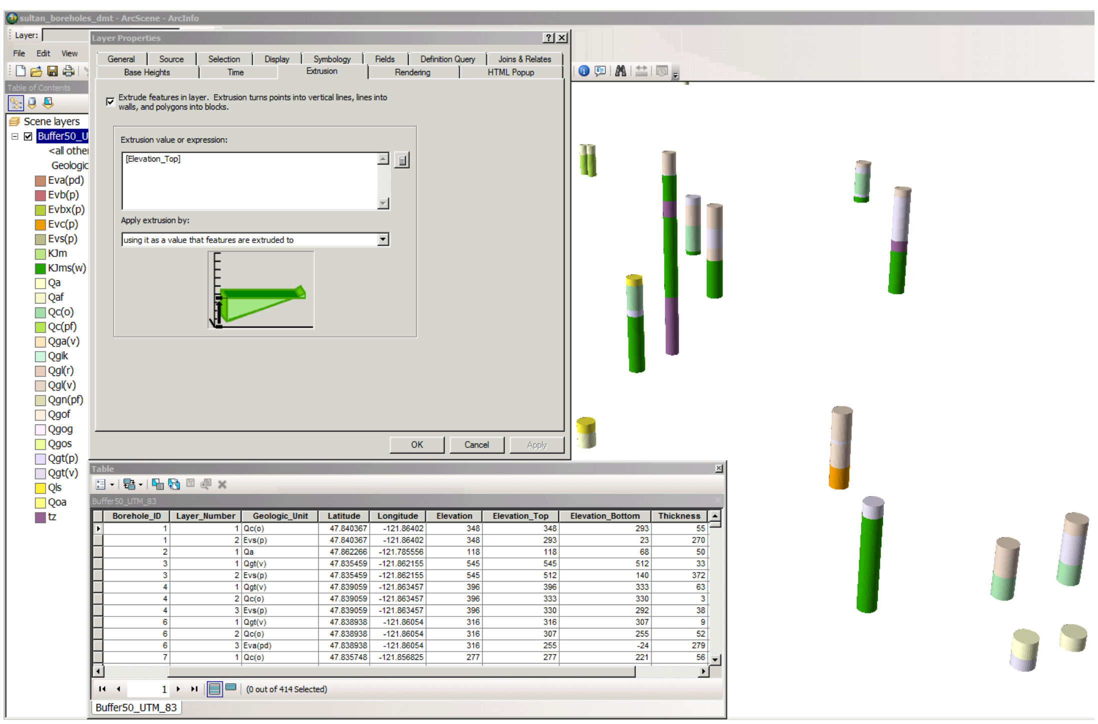


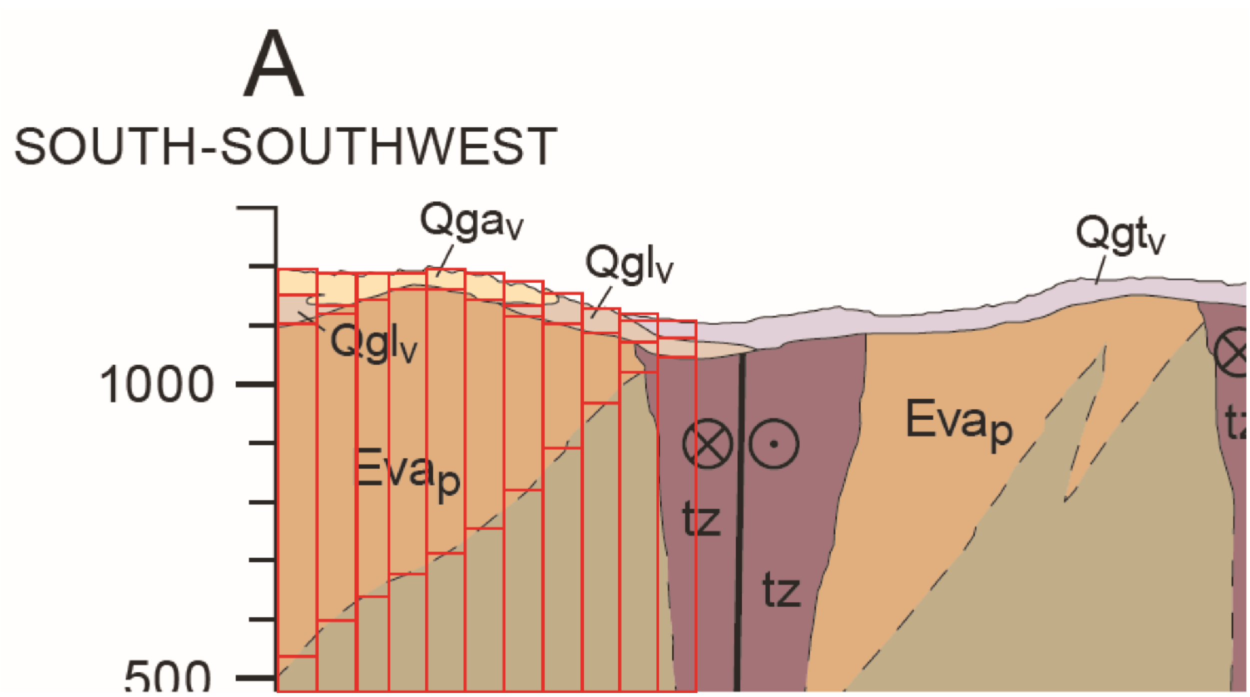
Figure 2 *- Buffered boreholes extruded in ArcScene*

A challenge presents itself however, as ArcScene will only extrude features with discrete elevations, making it impossible to draw curved surfaces. Depth values for a geologic unit changes with distance however; the polygon can have only a single elevation value. Our solution is to “discretize” the cross section polygons by slicing each into small segments (Fig. 3). This achieves an overall appearance of a curved surface when viewed at a reasonable distance while still maintaining the discrete elevations required by ArcScene. For a given “slice” of the line of cross section there will be a line created for each geologic unit encountered at depth. For a typical 7.5-minute quadrangle cross section, we find that a 10-foot interval is a good balance between appearance and file size.
The outline of the process for each major step is as follows:

- The XY coordinates are located for the vertices in the line of cross section
- The difference between X and Y coordinate is calculated for each vertex pair
- The total of 10-foot sections calculates from the line segment length
- The change in X and change in Y are calculated for each 10 foot segment
- Attributes for top and bottom elevation and the geologic unit originate from the cross section and coded into the lines while being built
- A line feature class stores the calculated line segments with applied attributes

While this is a simplification of all the steps required in the process, it is evident that this processing requires extensive calculations. We automate these calculations through a script tool written in Python. With the proper inputs provided, the script tool will create a feature class consisting of many short lines with attribute data for geologic unit symbol, top depth, and bottom depth.

__Figure 3__ - Schematic example of the discretizing process



We can now add the feature class created by our custom script tool into ArcScene. Base height is set to the top elevation, extrusion is set to the bottom height (with “elevation features are extruded to” selected), and symbology set to unique values based on the geologic unit. As the image shows (Fig. 4), the cross section has the correct orientation and scale with little or no evidence of the discretizing process aside from some unusual shading from the scene’s renderer. This cross section can now be exported from ArcScene into a VRML (.wrl) file format, which stores the physical attributes (color, transparency), and 3D geometry of each object in the scene.

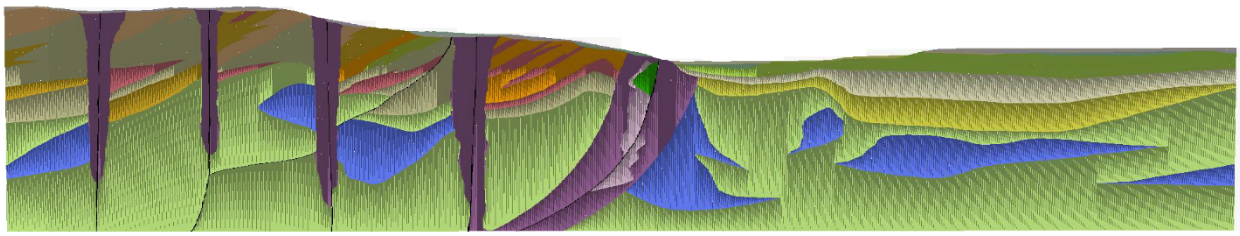


Figure 4 *- Cross section as shown in ArcScene*

Regrettably, the export process removes any attribution that the ArcScene objects had and consequently each geologic unit exists as a generic “Shape” object. We can override the name however, with a minor bit of modification to the .wrl file. Opening the .wrl in a text editor reveals human-readable Unicode text. By replacing “Shape”, with “DEF ‘your_text_here’ Shape” we rename each object to the correct geologic unit symbol. This process is tedious and highly consuming if done entirely manually; again a python script tool automates this step. The script reads the feature class that the .wrl generates from, reads the old .wrl, and writes out a new .wrl file with the instances of “Shape” replaced by the appropriate unit tag. This step now completes in 10s of seconds where it would typically take an hour or more if done by hand.

Launch the PDF3D ReportGen software and add the .wrl files. The software will read the file and convert it into an .u3d file that is readable by Adobe Acrobat. The program then embeds the .u3d into a .pdf file turning it into a 3D pdf. This 3D pdf is now viewable by any computer running Adobe Reader v.7 or higher.

PDF3D ReportGen is not limited to only importing .wrl files. It can handle most geospatial data types including (but not limited to) LAZ, DEM, IMG, SHP, ASC, KML, and TIF to name a few. This makes it a very useful tool as a data may originate from a wide variety of sources.

With the data brought into ReportGen and all the desired settings are applied, we export a “state file” for additional processing. This file holds all the commands and settings required for the ReportGen software to build a 3D model. Luckily the file is written in .xml and is human readable. Here we can add additional data to the model to replace the attribute information from shapes removed by the .wrl export from ArcScene. In this case, we replaced the data for the geologic unit name, age, and description. The data must have properly formatted xml tags to attach correctly to the model. Another python script tool automatically reads the geologic information out of a tabular dataset, formats it, and makes it ready to paste into the state file. JavaScript allows for the addition of tools or customization to the model and Adobe has several manuals easily found online which explains the use of JavaScript in both regular and 3D PDF. In this case, we chose to create our own object data tool (metadata tool) which will display a selected shape’s data instead of using the prebuilt one for aesthetic reasons.

Once converted by ReportGen, the final PDF (Fig. 5) is fully interactive and viewable. At this step, creation of custom views aid in drawing the user’s attention to important features of the model. It is strongly suggested that the final PDF (once all views and settings are applied) be saved as a “Reduced size PDF” in order to optimize the file and reduce file size. For a typical 7.5-minute quadrangle map with cross sections, file size can range from 40-70 Mb. The performance of the PDF may vary on different workstations and we suggest including a reminder to users that it may take a minute or two to load on slower computers.

At this point, there was a live demonstration provided to the audience of the DMT 2015 Conference. For those unable to attend, the Washington State DNR Division of Geology and Earth Resources has a 3D geology page (http://www.dnr.wa.gov/programs-and-services/geology/geologic-maps/3d-geology) with links to tutorial videos, instructional pdf, and a large series of 3D PDFs.

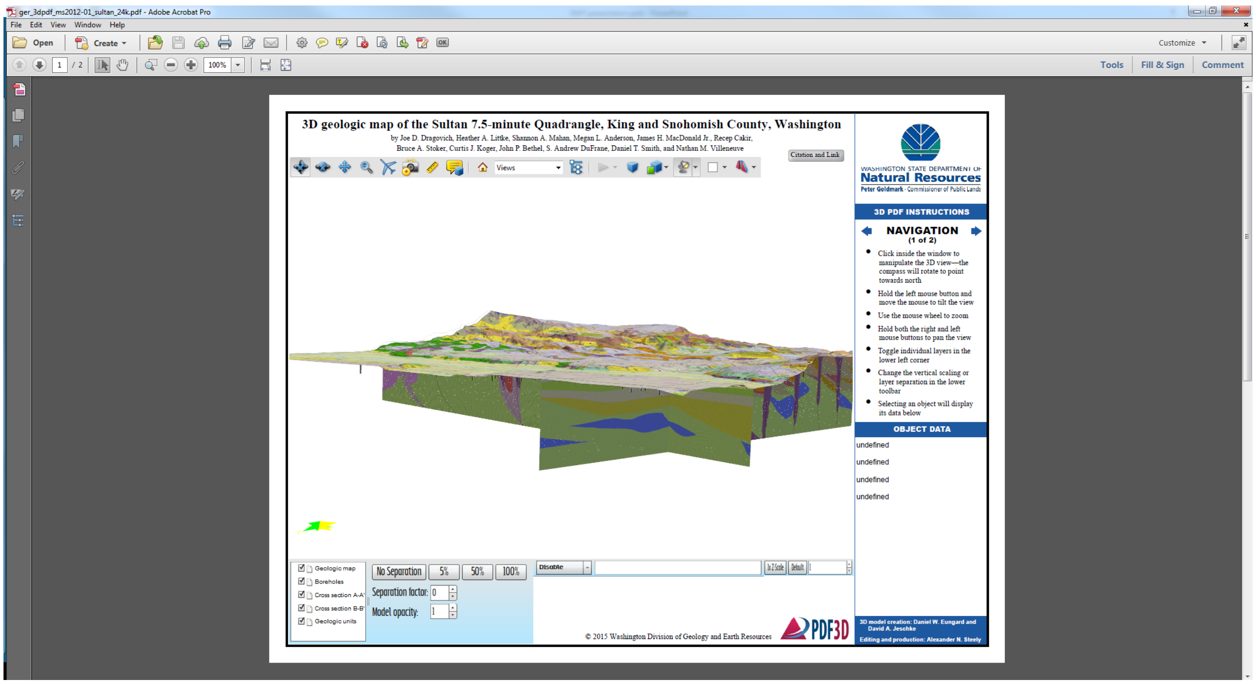


Figure 5 *- 2D capture of the final 3D model and template*

Building 3D PDFs to Visualize Geological Data

Daniel W. Eungard and David A. Jeschke

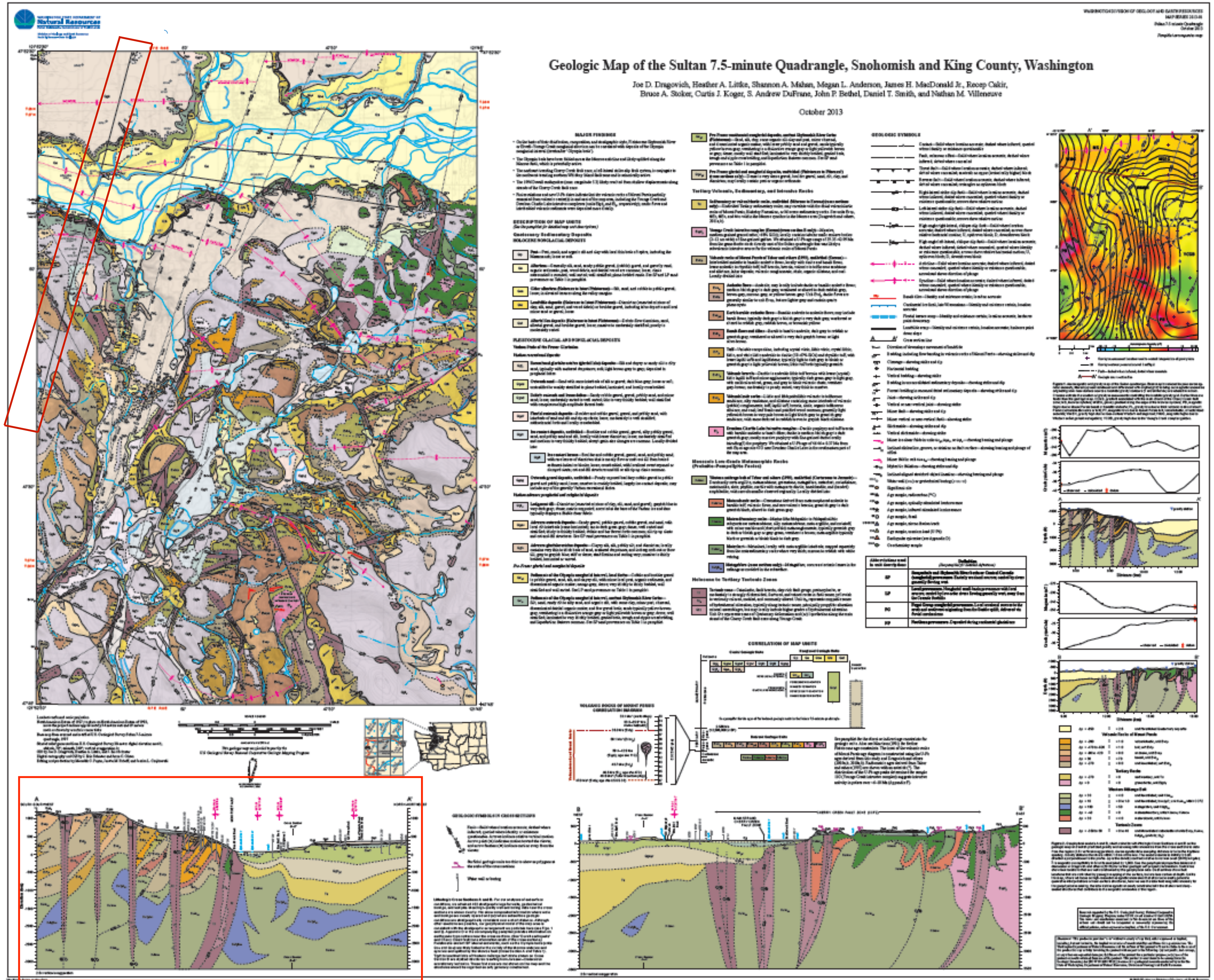
Digital Mapping Techniques 2015

WASHINGTON STATE DEPARTMENT OF
Natural Resources
Peter Goldmark - Commissioner of Public Lands

Division of Geology and Earth Resources
David K. Norman - State Geologist

Part 1: Subsurface geometry using Python and ArcScene – David A. Jeschke

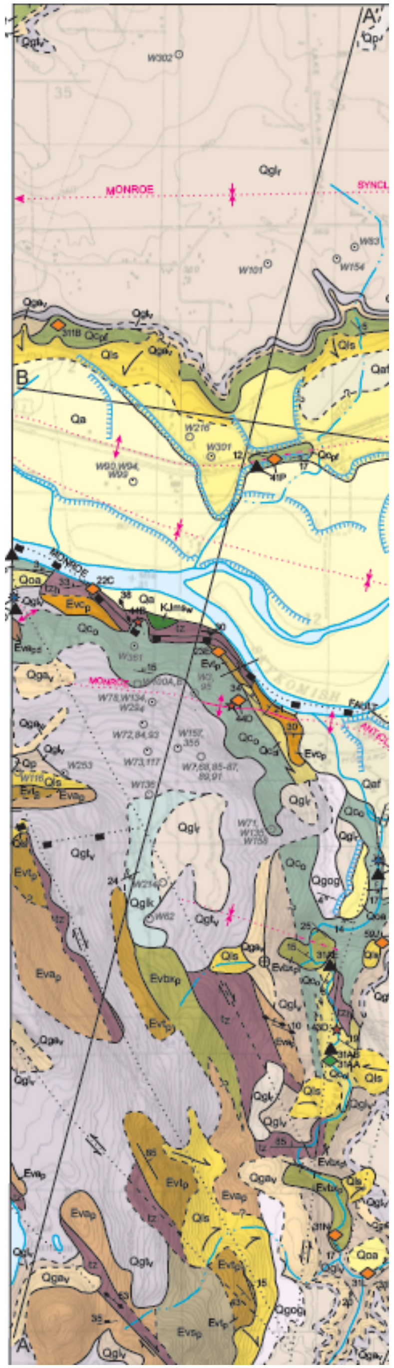
Part 2: Building professional-quality 3D PFD files based on ArcScene export files using ReportGen – Daniel W. Eungard



Line of cross section A-A’

Cross section inset A-A’

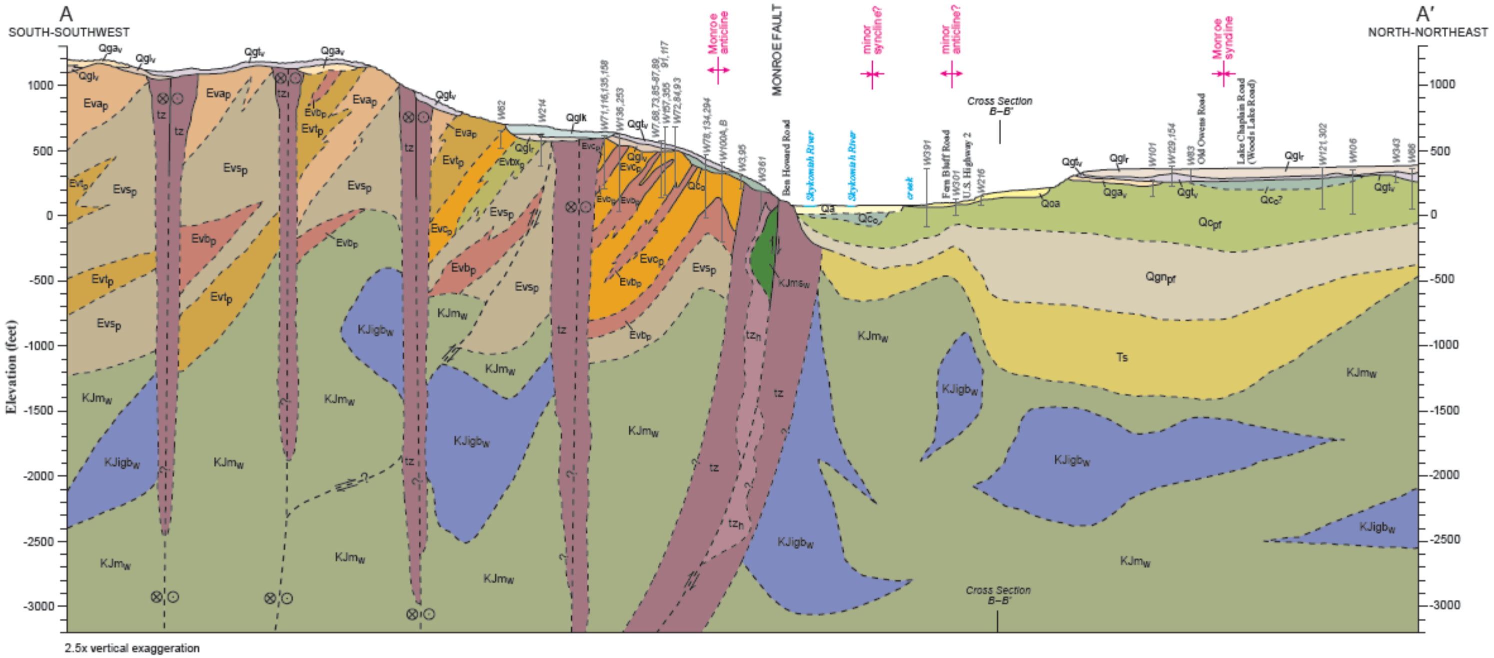
The map reader must attempt to visualize the terrain and subsurface geology from these two elements.

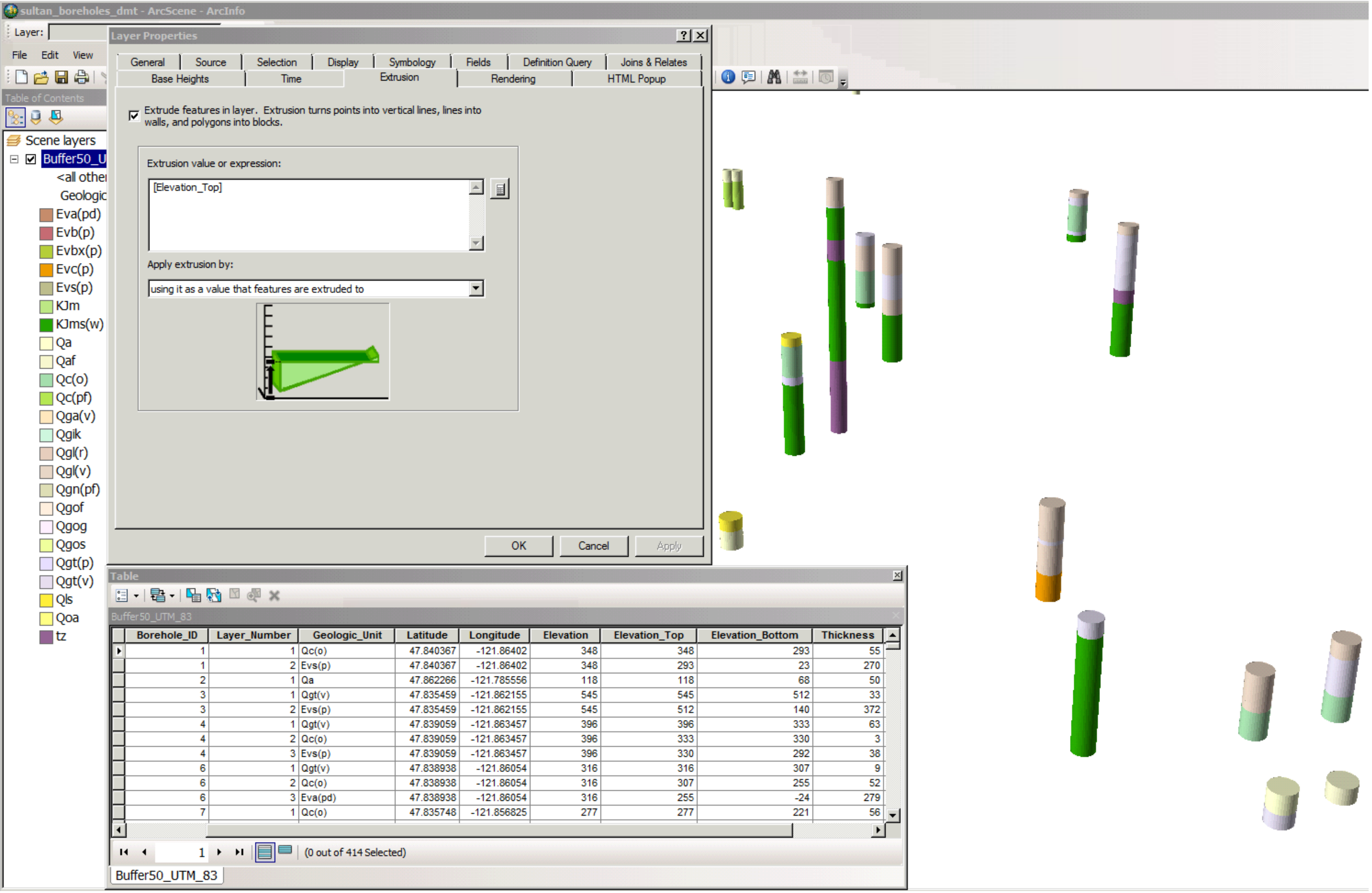


Our profile (below) begins in ArcMap and is finished in Illustrator

The profile geometry poses a challenge for 3D visualization:

- The X axis on the profile view represents distance along the line of cross section: a combination of X and Y in plan-view coordinates
- The Y axis on the profile represents the Z axis in plan-view coordinates
- These coordinates must be translated into an ArcScene readable format



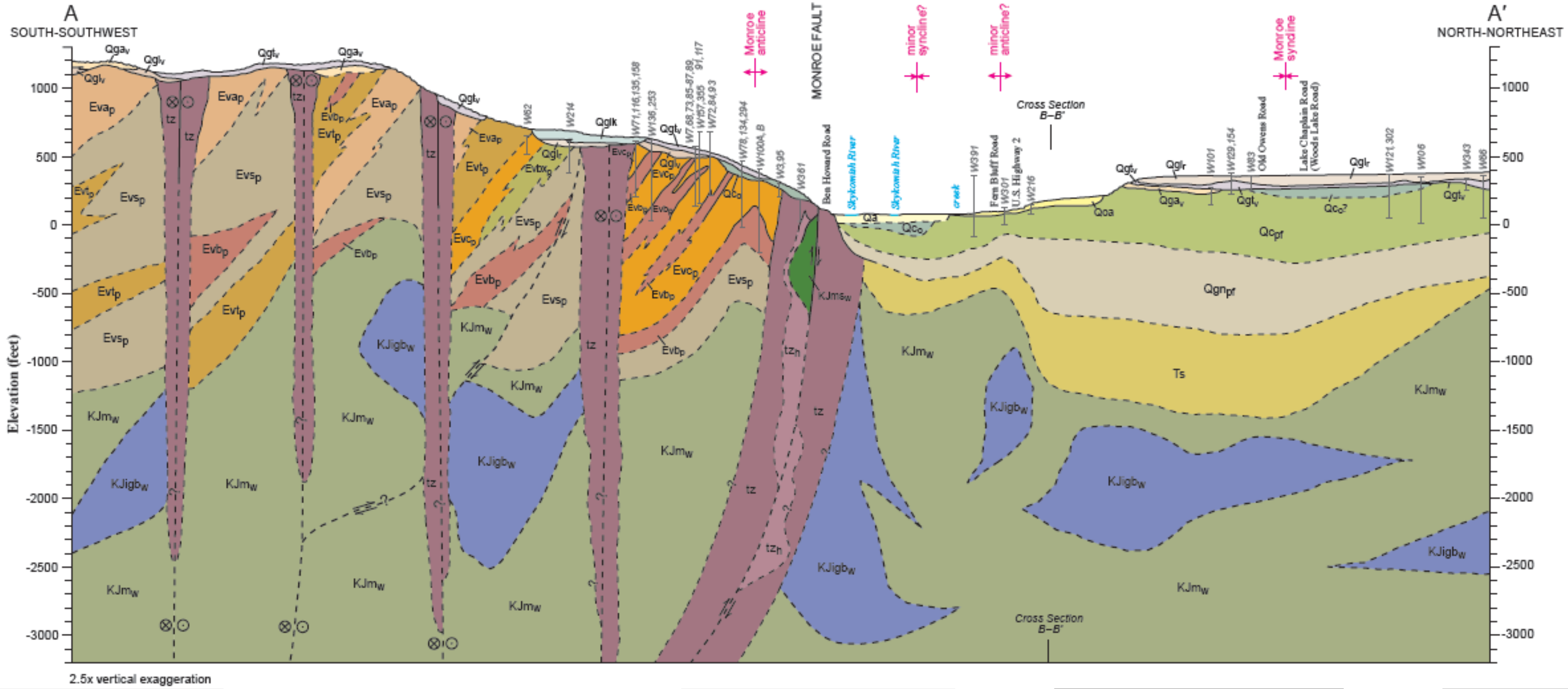


What can ArcScene read?

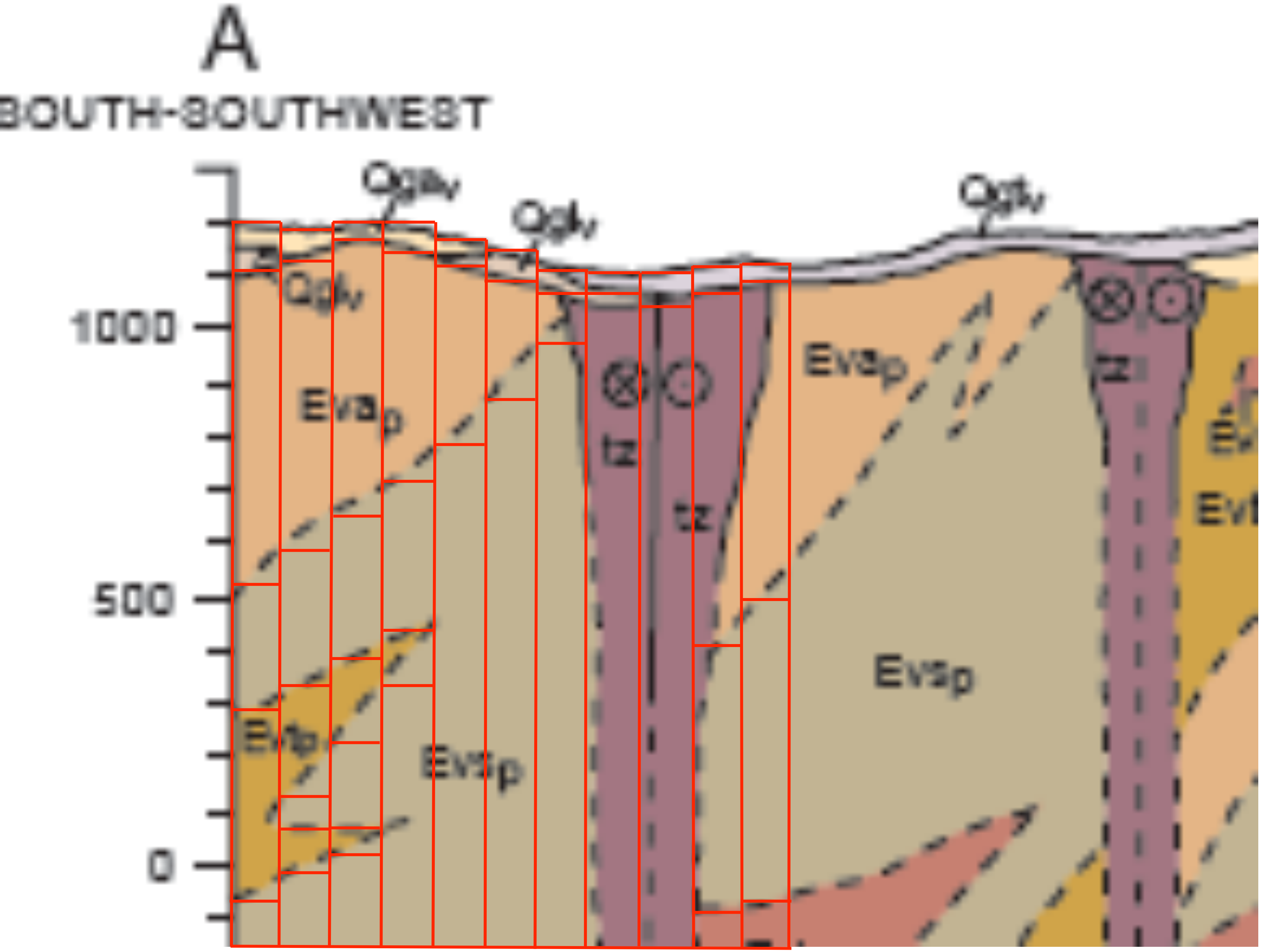
- X and Y coordinates are from feature geometry
- Z coordinates are from feature attributes
- These borings are “extruded” using elevation at the top and bottom of each layer
- Each boring is multiple polygons; each has its own top and bottom elevations, and geologic symbol

Preparing cross section data for display in ArcScene

- Depths of units vary continuously over the length of the line of cross section
- Each feature “extruded” in ArcScene must have discrete top and bottom elevations
- This presents a quandary: depth values must change and yet cannot change



Building 3D cross sections → Export VRML data → Modify VRML files → Build model in ReportGen → Modify state file → Complete model



Our solution: chop the cross section into itty-bitty vertical slices

- We build overlapping 10 foot lines along the line of cross section
- Each overlapping line represents one geologic unit
- Each has its own top and bottom depths, and unit symbol
- Curves become "discretized" into imperceptibly small horizontal lines

Building 3D cross sections → Export VRML data → Modify VRML files → Build model in ReportGen → Modify state file → Complete model

X and Y coordinates must be found for the start and end points of each 10 foot line segment

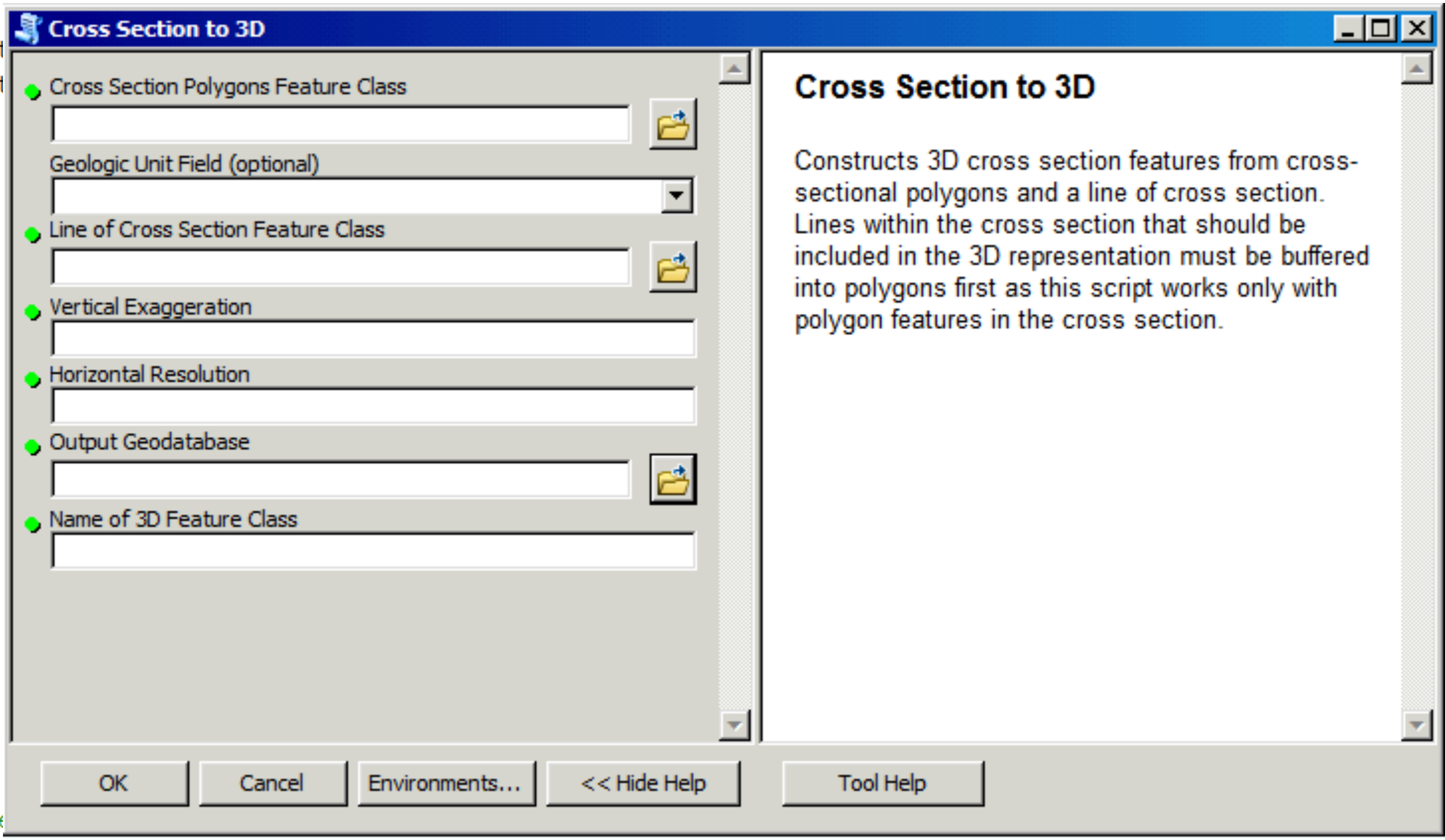
- X and Y coordinates are found for 1st and 2nd nodes in line of cross section
- The difference in X coordinates, and the difference in Y coordinates are determined
- The number of 10 foot line segments is determined
- The change in X and Y are divided by the number of 10 foot segments
- This represents the change in X, (ΔX) and the change in Y, (ΔY) for each 10 foot segment
- Lines are built in a feature class with X and Y coordinates for endpoints incrementing by ΔX and ΔY
- Attributes for Top Elevation, Bottom Elevation, and Geologic Unit are coded for each new line built

Building 3D cross sections → Export VRML data → Modify VRML files → Build model in ReportGen → Modify state file → Complete model

```
# Loop through column segments polygon features and make a new
rows = arcpy.SearchCursor("exploded_segments")
row = rows.next()

newPoint = arcpy.CreateObject("Point")
array = arcpy.CreateObject("Array")
cx3DRows = arcpy.InsertCursor(cx3D)
x=1

while row:
    feat = row.getValue("SHAPE")
    ext = feat.extent
    minx = ext.XMin - (cumulativeDistance - segmentLengt
    maxx = ext.XMax - (cumulativeDistance - segmentLengt
    topElevExag = ext.YMax
    botElevExag = ext.YMin

    # find X Y coordinates which correspond to minx and
    proportion = minx/segmentLength
    xOffset = proportion*xInterval
    yOffset = proportion*yInterval
    newPoint.X = x1+xOffset
    newPoint.Y = y1+yOffset
    array.add(newPoint)
    proportion = maxx/segmentLength
    xOffset = proportion*xInterval
    yOffset = proportion*yInterval
    newPoint.X = x1+xOffset
    newPoint.Y = y1+yOffset
    array.add(newPoint)

    # build polyline feature with elevations
    cx3DRow = cx3DRows.newRow()
    cx3DRow.shape = array
    if gUnitFieldName is not "#":    #the blank gunit fie
```

All of this processing is done using an ArcGIS script tool

This tool calls on a Python script we wrote for this purpose (a portion of the script is shown at left)

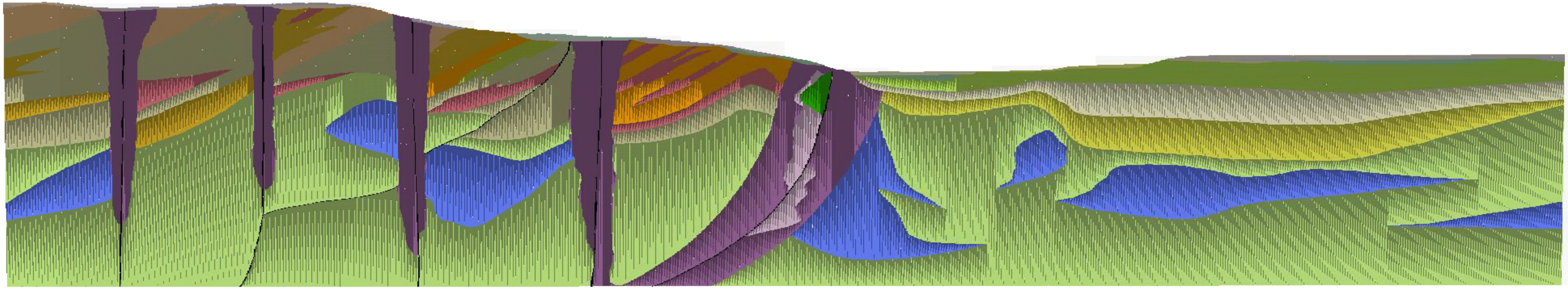


Building 3D cross sections → Export VRML data → Modify VRML files → Build model in ReportGen → Modify state file → Complete model

The 3D cross section displayed in ArcScene ☺

This is still not accessible to most users ☹

So we export it to VRML (only export option) and …



*Cross section as viewed in ArcScene. Note the discretized segments are apparent due to ArcScene's renderer.

Building 3D cross sections → Export VRML data → Modify VRML files → Build model in ReportGen → Modify state file → Complete model

```
127 Group
128 {
129  children
130  [
131   Group
132   {
133    children
134    [
135     Group   # (Layer node)
136     {
137      children
138      [
139      ]
140     }
141     Group   # (Layer node)
142     {
143      children
144      [
145       Shape
146       {
147        appearance
148         Appearance
149         {
150          material
151           Material
152           {
153            ambientIntensity 0.400
154            diffuseColor     0.651 0.710 0.522
155            emissiveColor    0.000 0.000 0.000
156            shininess        1.000
157            specularColor    0.000 0.000 0.000
158            transparency     0.000
159           }
160         }  # end appearance
161        geometry
162         IndexedFaceSet
163         {
164          ccw   FALSE
165          solid FALSE
166          coord
```

```
127 Group
128 {
129  children
130  [
131   Group
132   {
133    children
134    [
135     Group   # (Layer node)
136     {
137      children
138      [
139      ]
140     }
141     Group   # (Layer node)
142     {
143      children
144      [
145       DEF cxa_Qgt Shape
146       {
147        appearance
148         Appearance
149         {
150          material
151           Material
152           {
153            ambientIntensity 0.400
154            diffuseColor     0.651 0.710 0.522
155            emissiveColor    0.000 0.000 0.000
156            shininess        1.000
157            specularColor    0.000 0.000 0.000
158            transparency     0.000
159           }
160         }  # end appearance
161        geometry
162         IndexedFaceSet
163         {
164          ccw   FALSE
165          solid FALSE
166          coord
```

Preparation for conversion

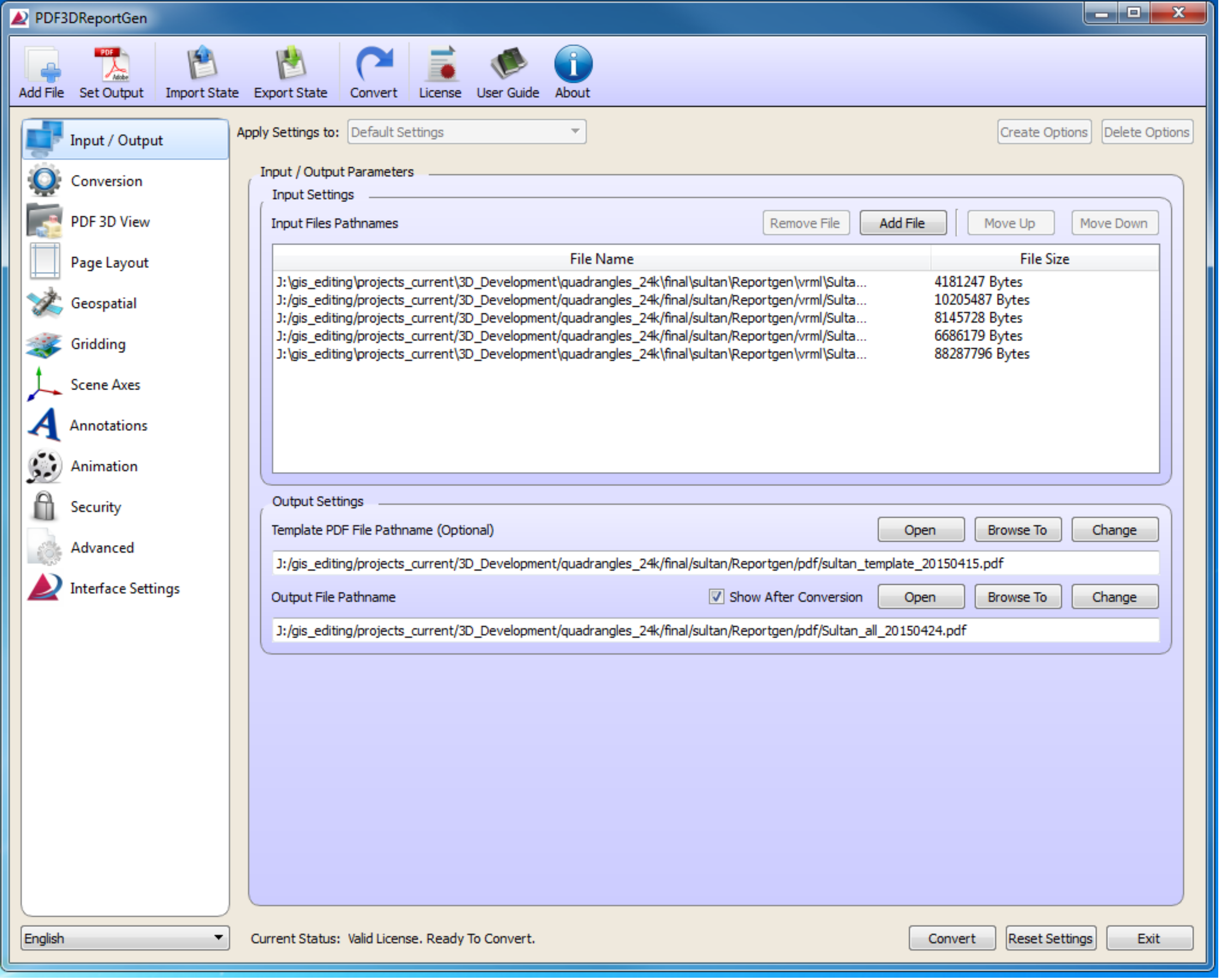
VRML files export each scene feature as a "Shape". This means nothing to the user and is not very helpful.

We can open the VRML in a text editor and give each Shape a name.

As ArcScene groups shapes by color and transparency, each geologic unit is represented separate from the rest.

*Before and after of VRML file in Notepad++, notice the change in line 145

Building 3D cross sections → Export VRML data → **Modify VRML files** → Build model in ReportGen → Modify state file → Complete model



We bring it into ReportGen, a software developed by PDF3D for creating 3D pdfs from spatial data.

This allows us to take an inaccessible VRML file and convert it into a very accessible PDF

* ReportGen software with model built.

Building 3D cross sections → Export VRML data → Modify VRML files → **Build model in ReportGen** → Modify state file → Complete model

PDF3D User Manual

Supported File Formats

- 3D GameStudio 3DGS Format (*.mdl)
- 3D GameStudio 3DGS Terrain Format (*.hmp)
- 30 Studio MAX Format (*.3ds)
- 3ds Max ASE Format (*.ase)
- AC3D Inivis Format (*.ac)
- AVS/Express UCD Format (*.inp)
- ArcGIS Geospatial Grid Format (*.asc)
- AutoCAD Export DWF Format (*.dwf)
- AutoCAD Generic DXF Format (*.dxf)
- AutoCAD Native DWG Format (*.dwg)
- Bentley-Intergraph Microstation DGN Format (*.dgn)
- Blender 3D Format (*.blend)
- BlitzBasic 3D Format (*.b3d)
- CATIA* v5 (*.catpart)
- Collada COLLAborative Design Activity Format (*.dae)
- COMSOL Simulation Results, Grid Format (*.dat, *.txt)
- Design Workshop Database (*.dw)
- DirectX 3D Model Format (*.x)
- DirectX X Format (*.x)
- Doom 3 Format (*.md5)
- ESRI ArcGIS Geospatial Vector Shapes (*.shp)
- Flash Animations or Movie clips (*.swf)
- GeoTIFF Geospatial Image and Grid Data Format (*.tif)
- GeoVRML Geospatial Format (*.wrl)
- GSI3D Geological Models(*.gsipr, *.gxml)
- IGES* 3D Model Interchange (*.igs)
- Image Files (*.PNG, *.JPG, *.TIF, *.BMP)
- IMAGINE Multi-Channel Image or Elevation Grids (*.IMG)
- Industry Foundation Classes IFC/STEP (*.ifc)
- Irricht Mesh Format (*.irrmesh)
- Irricht Scene Format (*.irr)
- IVE OpenSceneGraph File Format (*.ive)
- KML Geospatial Vector Features (*.kml)
- Nendo Format (*.ndo)
- LAS Well Log (Subset) (*.las)
- LiDAR LAS File Format v1,2,3 (*.las)
- LiDAR LAZ Compressed Format (*.laz)
- LightWave Object Format (*.lwo)
- LightWave Scene Format (*.lws)
- Milkshape 3D Format (*.ms3d)
- Modo Format (*.lxo)
- Movie.BYU Geometry Format (*.byu)
- Neutral File Format (*.nnf)
- OSG Extendable ASCI Format (*.osgt)
- OSG Extendable Binary Format (*.osgb)
- OSG Extendable XML Format (*.osgx)
- OSG Native Format (*.osg)
- OSGTGZ Compressed Format (*.osgtgz)
- Object File Format (*.off)
- Ogre Graphics Engine XML Format (*.xml)
- OpenFlight Format (*.flt)
- OpenInventor 2.1 Compressed Format (*.iv.gz)
- OpenInventor 2.1 Format (*.iv)
- Point Cloud Formats (*.csv,*.pts,*.xyz)
- Point Cloud with Color (*.xyzrgb,*.xyzi)
- Polygon File Format Stanford (*.ply)
- PovRAY Raw Format (*.raw)
- PRC Product Representation Compact Format (*.prc)
- Protein Data Bank Molecular Format (*.pdb)
- Quake I Format (*.mdl)
- Quake II Format (*.md2)
- Quake III Map/BSP (*.pk3)
- Quake III Mesh Format (*.md3)
- Quick3D Format (*.q3s)
- Return to Castle Wolfenstein Format (*.mdc)
- Sense8 WorldToolKit Format (*.nff)
- Starcraft II M3 Format (*.m3)
- STEP* 3D Model Interchange (*.stp)
- Stereolithography ASCII Multi-part File Format (*.stla)
- Stereolithography Binary File Format (*.stlb)
- Stereolithography STL File Format (*.stl)
- Surfer Grid Format (*.grd)
- Surfer Colormap Format (*.clr)
- Terragen Terrain Format (*.ter)
- TrueSpace Format (*.cob,*.scn)
- USGS DEM Geospatial Grid File Format (*.dem)
- Unreal Game Format (*.3d)
- VRML Compressed Format (*.wrz, *.vrml.gz)
- VRML Uncompressed Format (*.wrl, *.vrml)
- VTK PolyData Format (*.vtp)
- VTK Model File Format (*.vtk)
- Valve Model Format (*.smd, *.vta)
- VOXLER 2, Scene format (*.iv)
- Wavefront Object Format (*.obj)
- XGL, XGL Format (*.xgl, *.zgl)
- ZMapPlus Geospatial Grid Field Format (*.dat)

An Aside:

Q: I don’t have or like ArcScene, do I need to use it?

A: No, you don’t. We chose to use it as we can visualize our models prior to conversion and for the ease of symbolization.

However, as you can see, ReportGen can handle many file formats directly including Shapefiles, ASC, XML, KML and many more…

Building 3D cross sections → Export VRML data → Modify VRML files → **Build model in ReportGen** → Modify state file → Complete model

```
        <CellSizeParameters width="100" geospatialWidth="1" squareCells="false" height="100" geospatialHeigh
    </GrdParameters>
    <Subsampling skipFullGrid="false" level="1"/>
    <BandBinding greenChannel="1" bindingMode="Automatic" attributeAutomatic="true" redChannel="0" greenAuto
    attributeChannelName="" redAutomatic="true" blueChannel="2"/>
    <ColormapBinding labelLegend="Elevation" usingTexture="true" filename=""/>
    <ElevationBinding bindingMode="Automatic" elevationAutomatic="true" elevationChannel="0"/>
    <Position autoPosition="false" useActualCenter="false">
        <ManualPosition x="0" y="0" z="0"/>
    </Position>
    <Scale generalScale="1" autoScale="false">
        <ScalePerComponent x="1" y="1" z="1"/>
    </Scale>
    <Visible value="true"/>
    <TerrainMode mode="Disabled"/>
</DefaultAssemblyProperties>
<PointsSubstitution substituteType="Automatic">
    <LinesLength value="1"/>
</PointsSubstitution>
<Metadata nodeName="gunitp_wtr">
<MetadataItem key="Description" value="Condensate of dihydrogen oxide which has undergone a vertical transla
quantities of dissolved salts and minerals."/>
<MetadataItem key="Age" value="Holocene"/>
<MetadataItem key="Name" value="Water"/>
<MetadataItem key="Unit" value="wtr"/>
</Metadata>
<Metadata nodeName="gunitp_fault">
<MetadataItem key="Description" value="A planar or gently curved fracture in the rocks of the crust, where c
the opposite sides of the fracture."/>
<MetadataItem key="Age" value=""/>
<MetadataItem key="Name" value="Fault"/>
<MetadataItem key="Unit" value="fault"/>
</Metadata>
<Metadata nodeName="gunitp_Ei(p)">
<MetadataItem key="Description" value="Uniquely textured medium-K calc-alkaline dacite flows (~68% SiO2) wit
and dacitic bomb breccia; rocks are bluish gray to gray. These flows, fragmental volcanic rocks, and possibl
Lake Fontal on the SE highlands in the east-central part of the map area. The flows or possible hypabyssal i
&quot;knots&quot; that define a subvertical mafic mineral lineation suggestive of vertical flow; however, th
requires further study. The flows are mostly holocrystalline and contain blocks of euhedral to microlitic pl
contain a variety of clasts, including pumice, basalt, hornblende-augite dacite, and hypidiomorphic intrusiv
```


Reportgen produces a “state file” for the conversion written in XML.

This means that we can modify it to change some parameters and add data such as unit symbols, descriptions, ages, you name it!

These descriptions were pulled from GIS and formatted in Excel to produce proper XML tags.

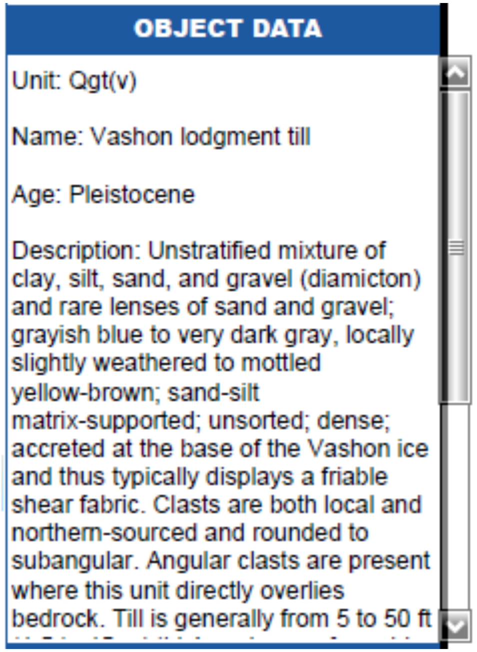
An entire map’s worth of data can be converted in <10 minutes.

Building 3D cross sections → Export VRML data → Modify VRML files → Build model in ReportGen → **Modify state file** → Complete model

```
colored, mottled, and veined as a result of local hydrothermal alt
of the map area. Sherrod and others (2008) map many strands of the
echelon vein arrays, suggest right-lateral strike-slip or oblique-
the Johnsons Swamp fault zone, and the Fontal Road reverse fault."
<MetadataItem key="Age" value="Tertiary to Holocene"/>
<MetadataItem key="Name" value="Tectonic zone"/>
<MetadataItem key="Unit" value="tz"/>
</Metadata>
<Metadata nodeName="cxb_tz(h)">
<MetadataItem key="Description" value="Hydrothermally altered tect
tz(h) contains principally low-temperature carbonate (calcite) min
<MetadataItem key="Age" value="Tertiary to Holocene"/>
<MetadataItem key="Name" value="Low-temperature, hydrothermally al
<MetadataItem key="Unit" value="tz(h)"/>
</Metadata>
<JavaScript>
var _ts3dhp_Attributes =  host.getField("A3DR_Text");
var myarray = [];
function findAndShowAttributes(node)
{
_ts3dhp_Attributes.value = "";
if (node.metadataString != "")
{
myarray = [];
var localxml = new XML( node.metadataString );
var xItems = localxml.item;
for (var j=0;xItems.length()>j;j++)
{
var name = xItems[j].@name.toString();
var value = xItems[j].@value.toString();
var attribute = name + ": " + value;
myarray.push(attribute);
}
_ts3dhp_Attributes.value = myarray[3] + "\r\r" + myarray[2] + "\r\
}
}
if (_ts3dhp_Attributes != null)
{
var _ts3dhp_PartAttributesSelect = new SelectionEventHandler();
_ts3dhp_PartAttributesSelect.onEvent = function( event )
{
if (event.selected)
findAndShowAttributes(event.node);
else
_ts3dhp_Attributes.value = myarray[3] + "\r\r" + myarray[2] + "\r\
}
runtime.addEventHandler(_ts3dhp_PartAttributesSelect);
}
</JavaScript>
</pdf3d:InputParameters>
```

Javascript is Acrobat's bread and butter, with it we can make tools to change the behavior or provide additional tools to the user.

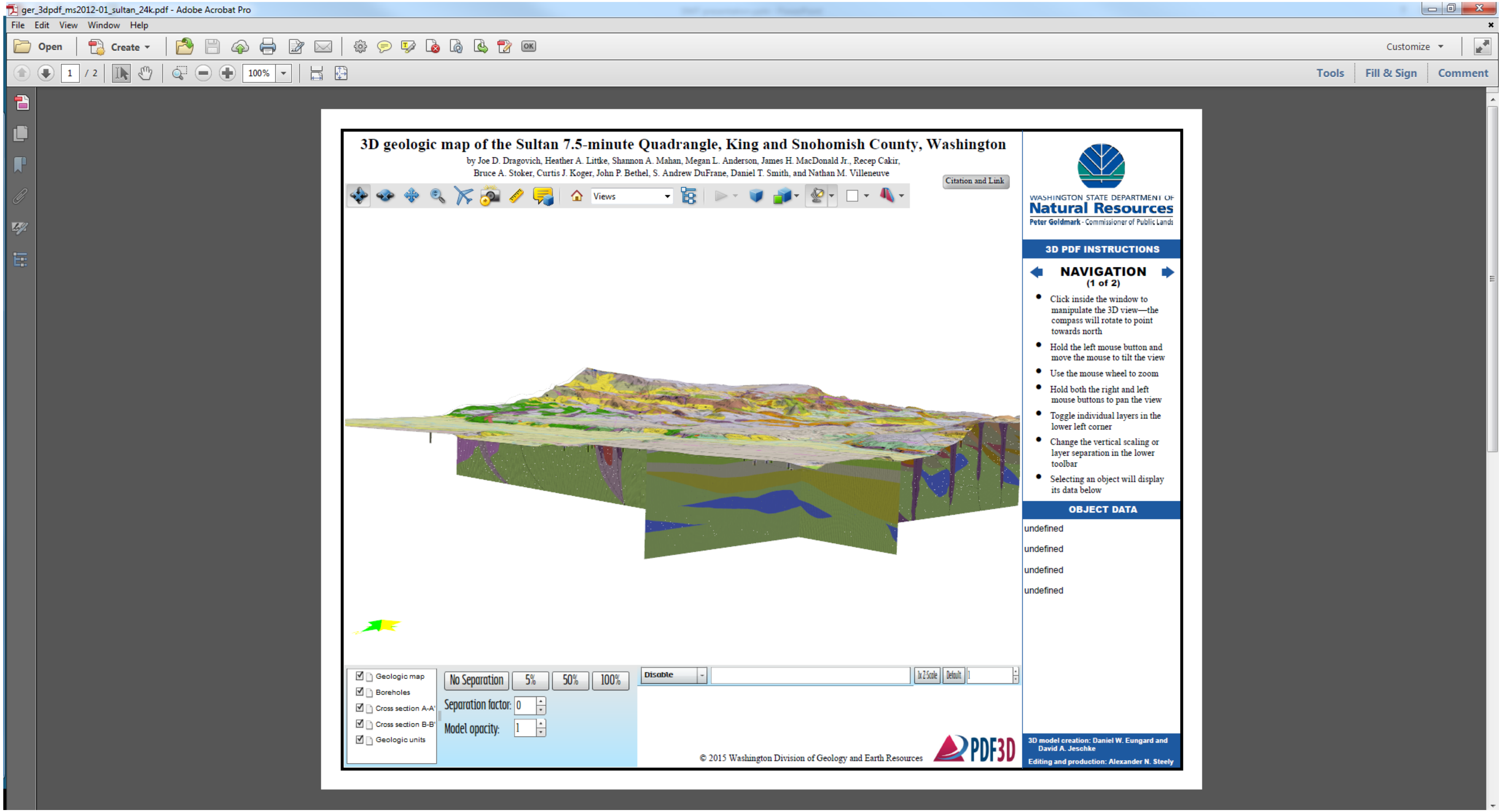
The last step for us is to add a custom Javascript which will display our data on the PDF page when a unit is clicked.



*unit description provided when unit is clicked in model

Building 3D cross sections → Export VRML data → Modify VRML files → Build model in ReportGen → **Modify state file** → Complete model

Final Product!!!



Building 3D cross sections → Export VRML data → Modify VRML files → Build model in ReportGen → Modify state file → Complete model

End of slideshow and beginning of live demonstration

For those who couldn't attend, follow along using any 3D model and the tutorials available at:

WA DGER 3D Geology webpage:
http://www.dnr.wa.gov/programs-and-services/geology/geologic-maps/3d-geology

Tutorial PDF:
https://fortress.wa.gov/dnr/geologydata/cartography/3d/3D_PDF_geologic_map_user_guide.pdf

Tutorial Video:
https://www.youtube.com/watch?v=b6wyGGqBVWM

Links are subject to change, see dnr.wa.gov if links here are broken.