Building 3D PDFs to Visualize Geological Data

Daniel W. Eungard and David A. Jeschke

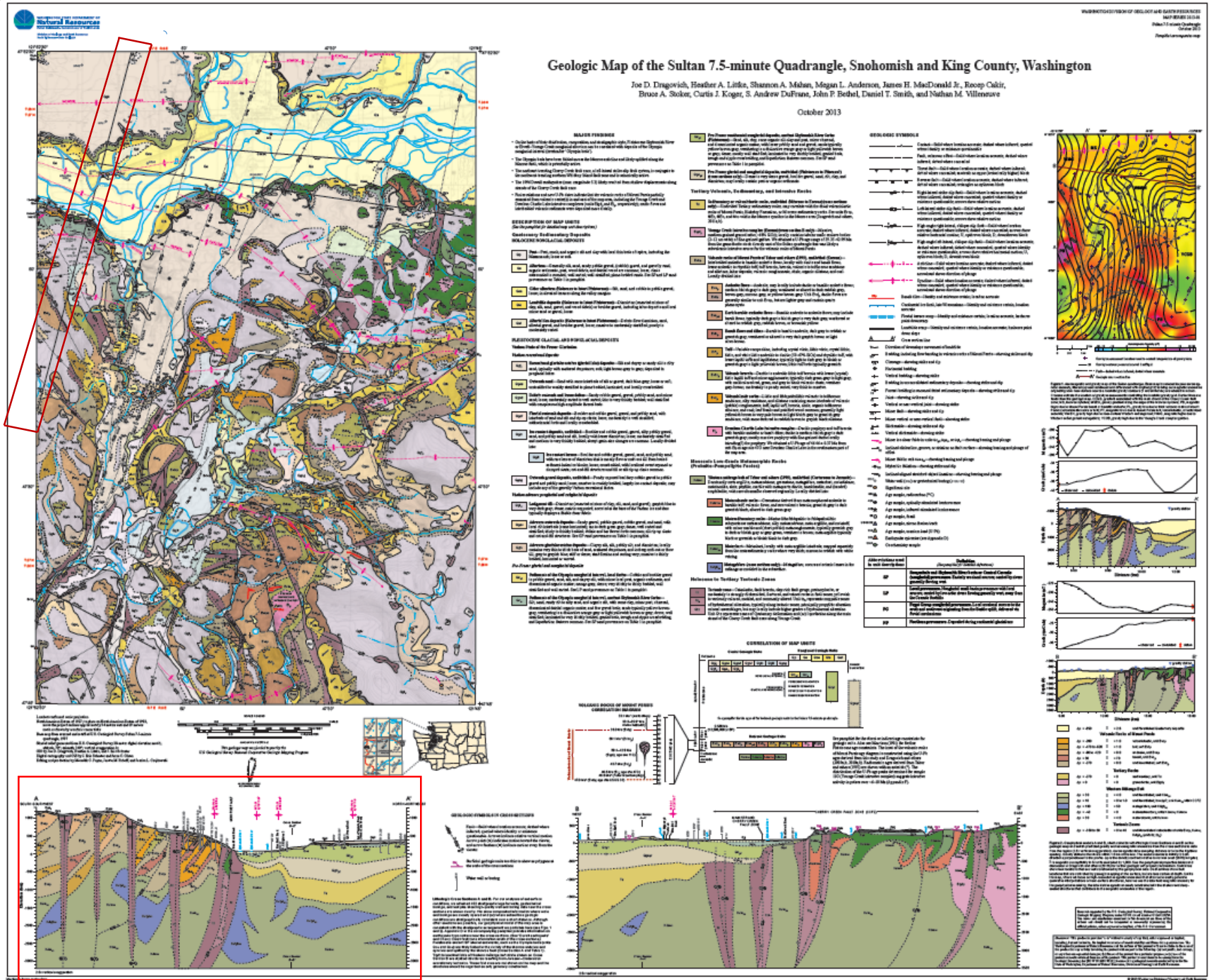
Digital Mapping Techniques 2015

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’

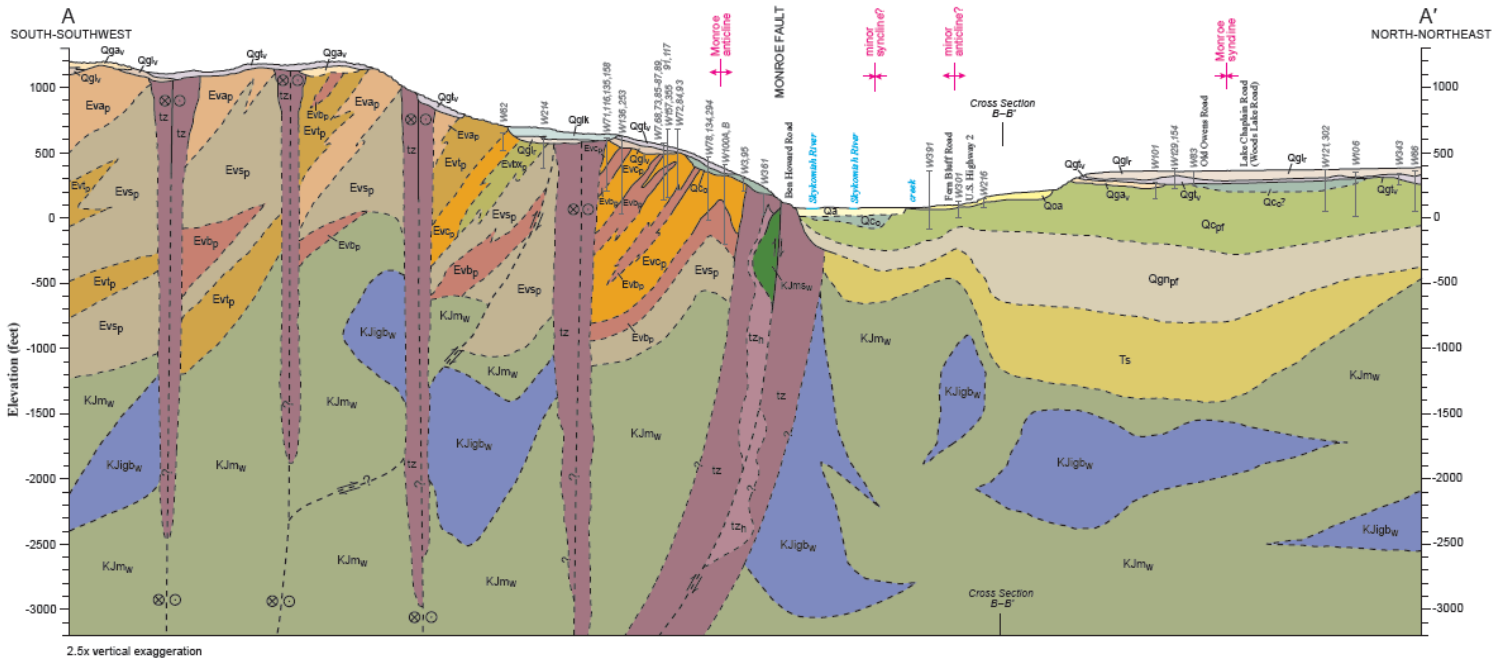
Cross section inset A-A’

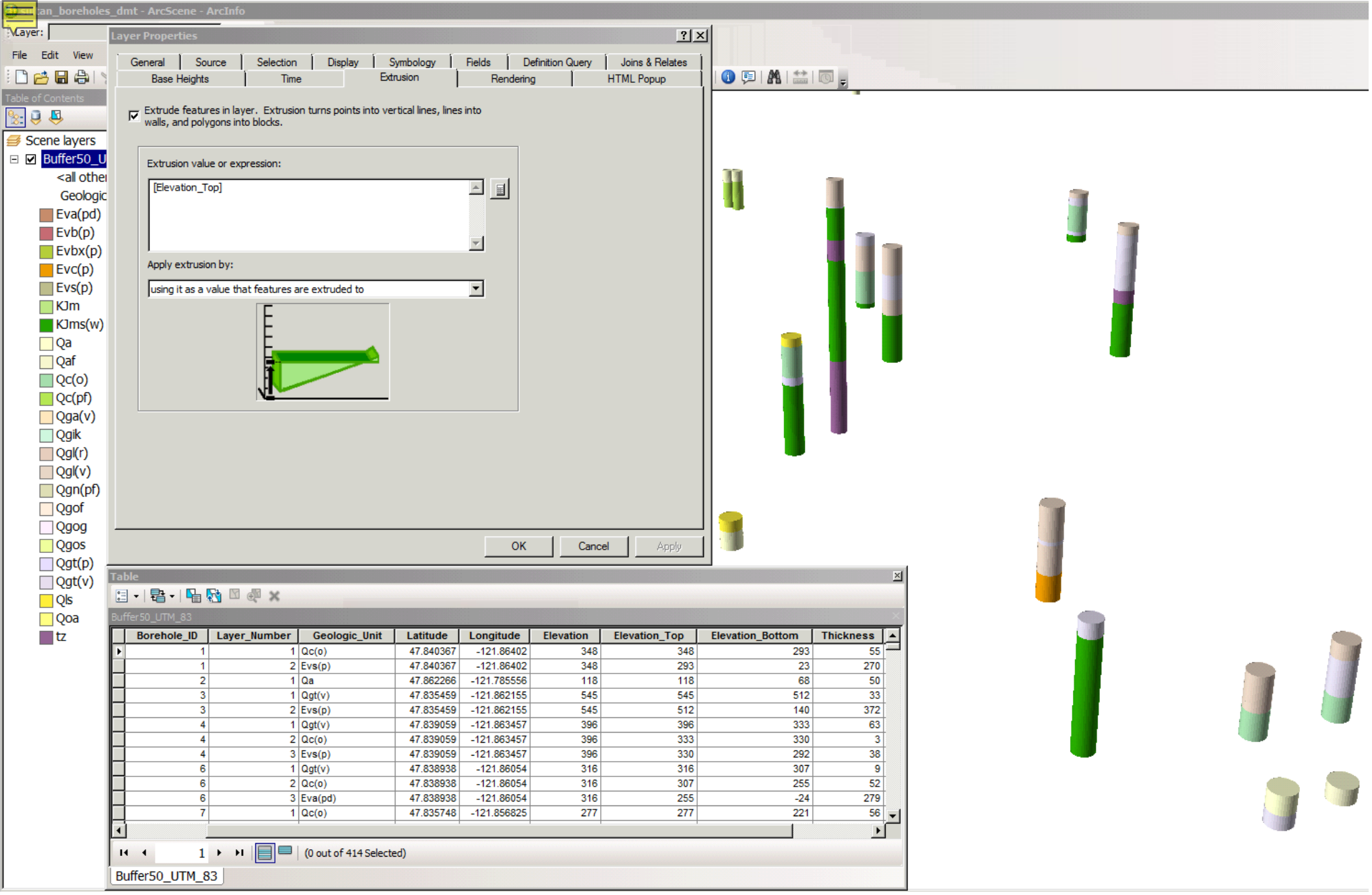
The map reader must attempt to visualize the terrain and subsurface geology from these two elements.

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



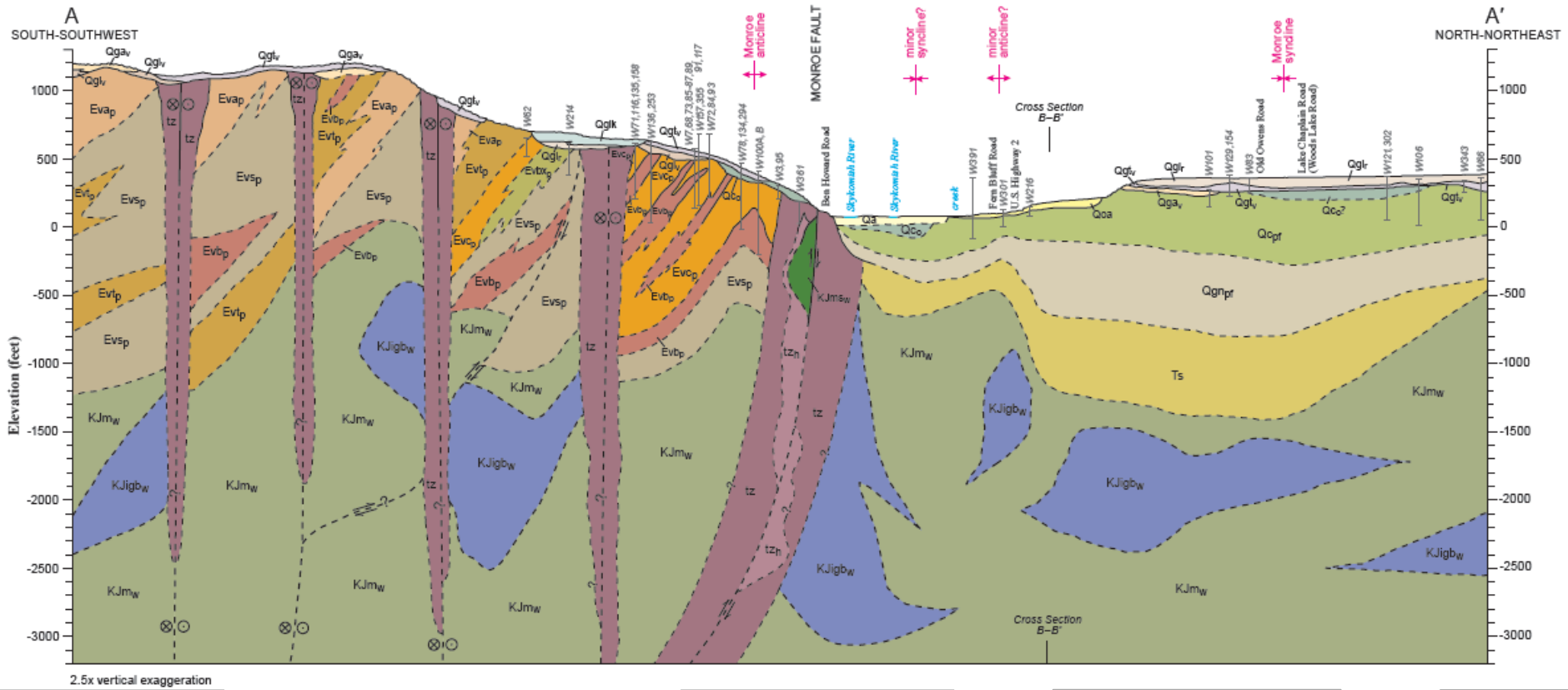


What can ArcScene read?

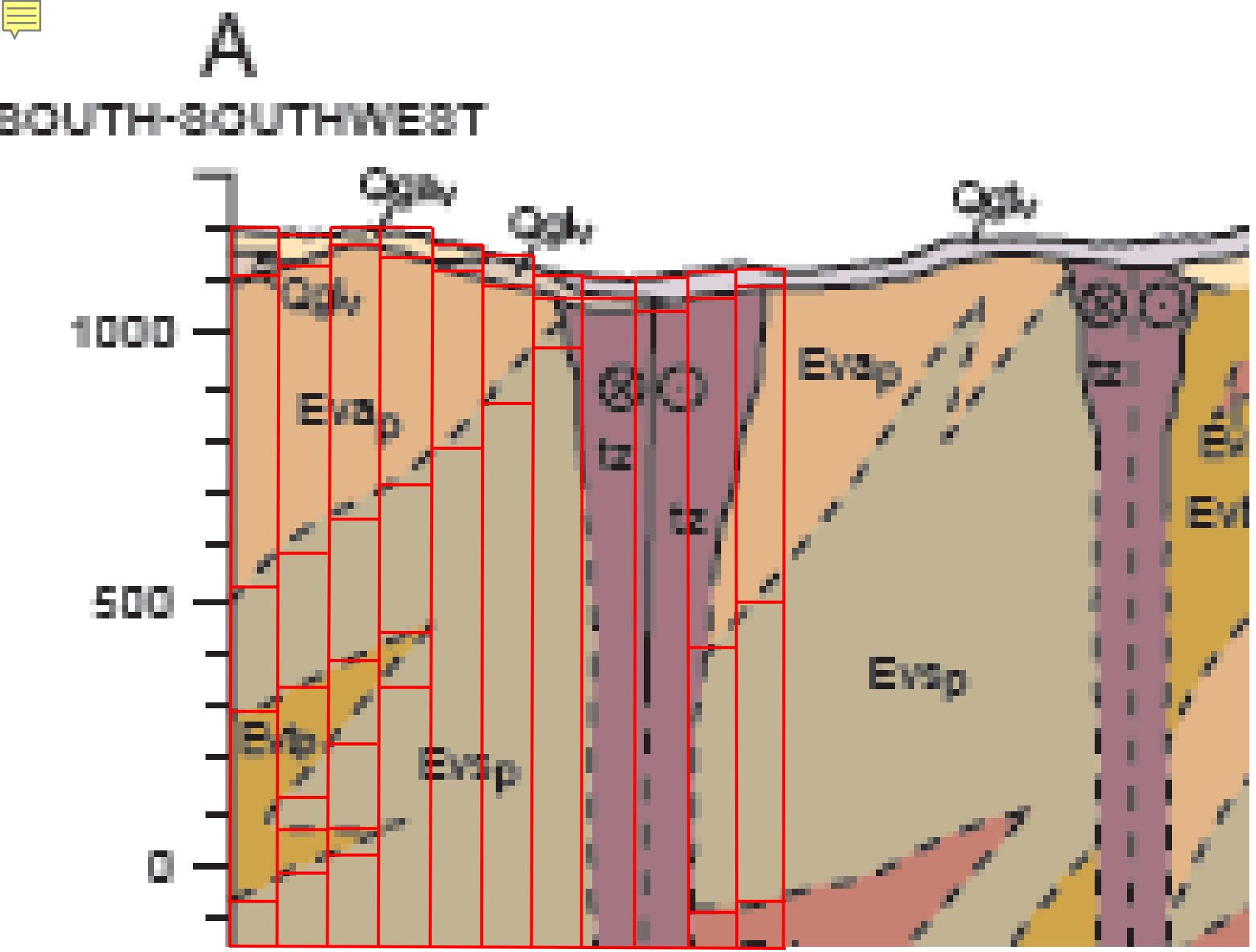
- X and Y coordinates are from feature geometry
- Z coordinates are from feature attributes
- These borings are “extruded” using elevation at the top and bottom of each layer
- Each boring is multiple polygons; each has its own top and bottom elevations, and geologic symbol

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

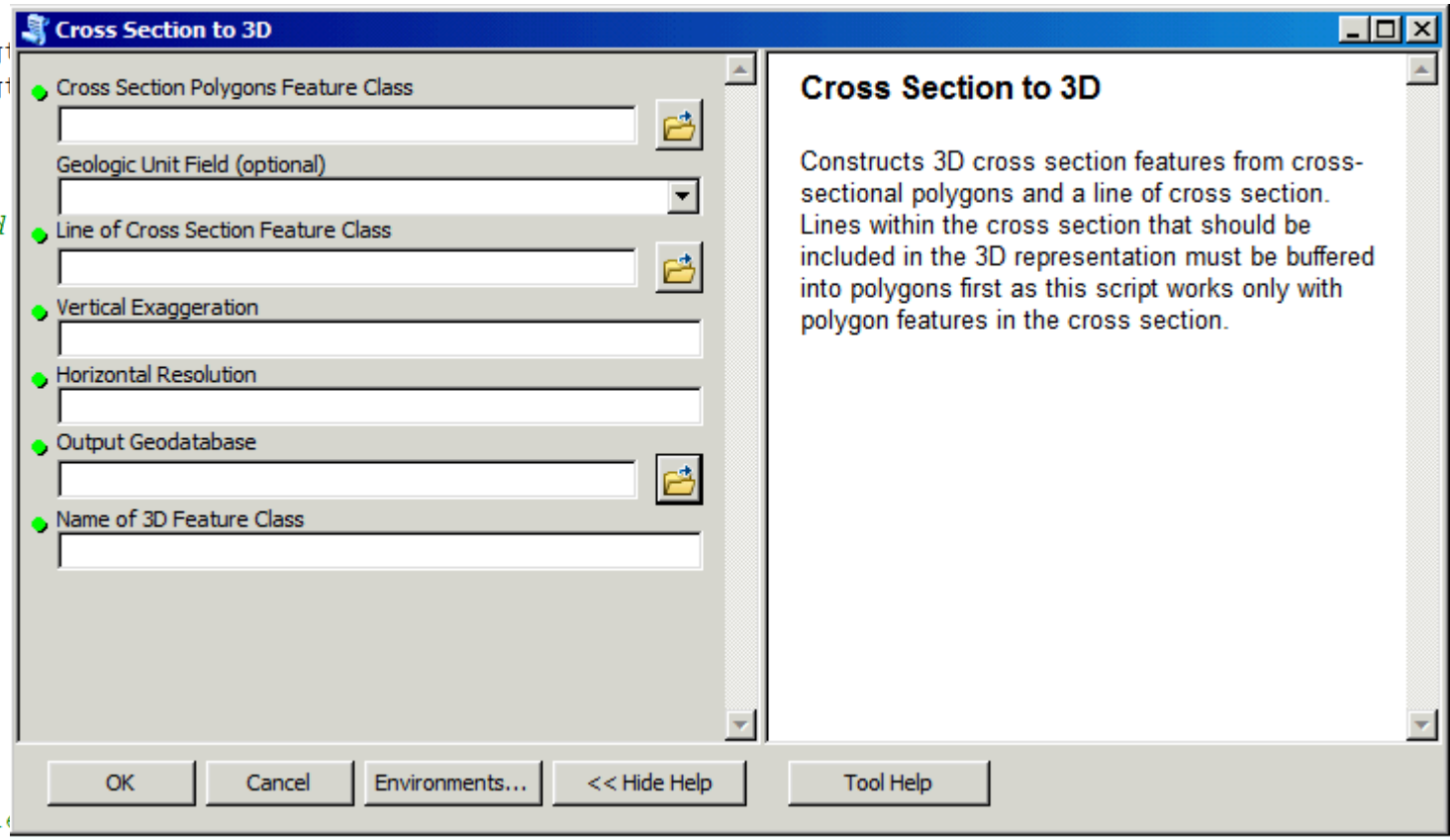
- X and Y coordinates are found for 1st and 2nd nodes in line of cross section
- The difference in X coordinates, and the difference in Y coordinates are determined
- The number of 10 foot line segments is determined
- The change in X and Y are divided by the number of 10 foot segments
- This represents 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 fi
```

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)

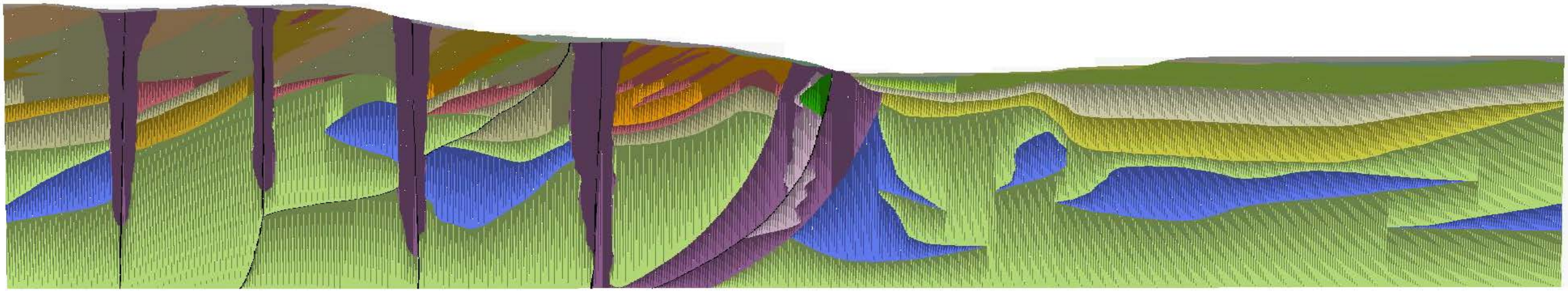


Building 3D cross sections → Export VRML data → Modify VRML files → Build model in ReportGen → Modify state file → Complete model

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

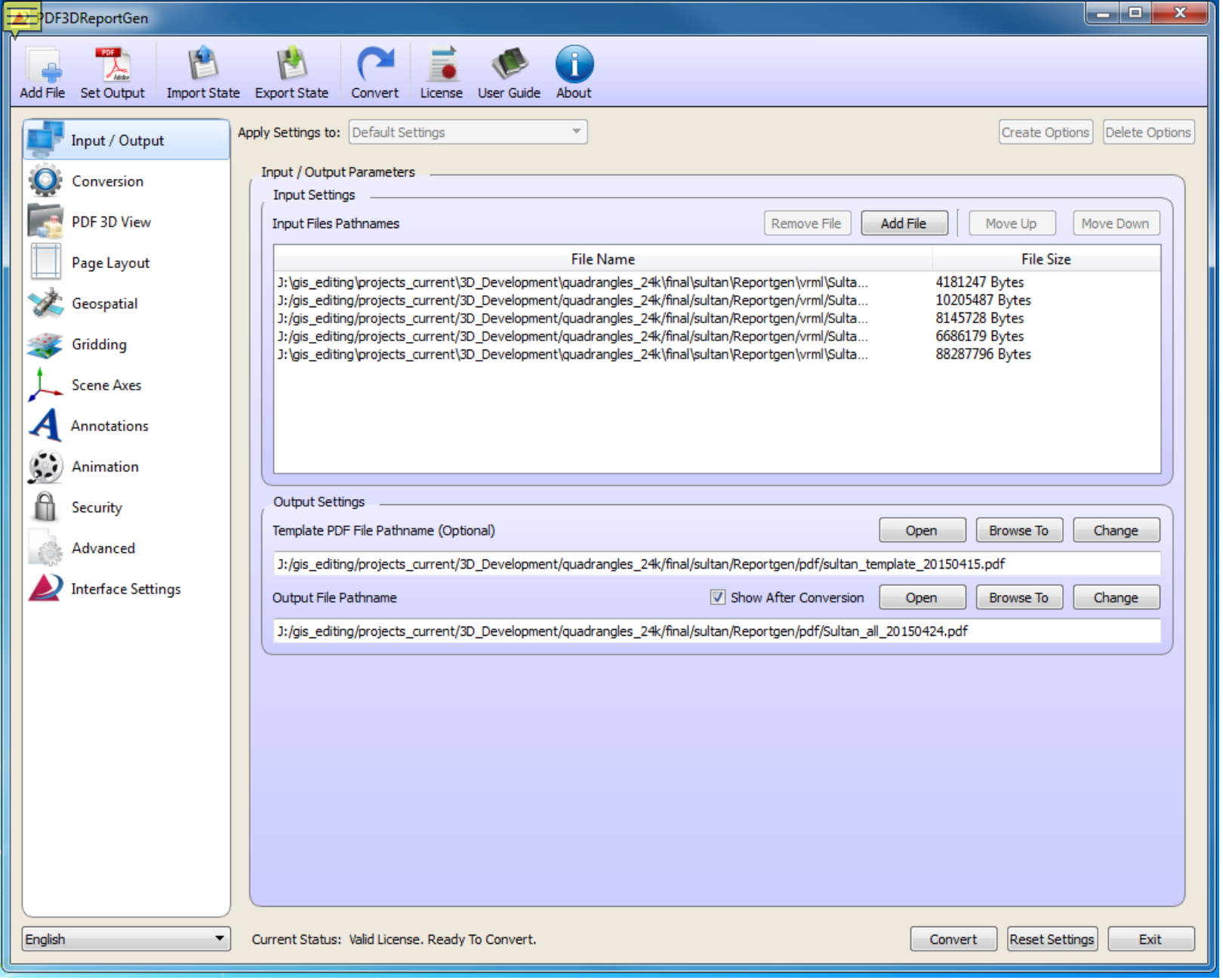
VRML files export each scene feature as a “Shape”. This means nothing to the user and is not very helpful.

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
&amp;quot;knots&amp;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
```

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!

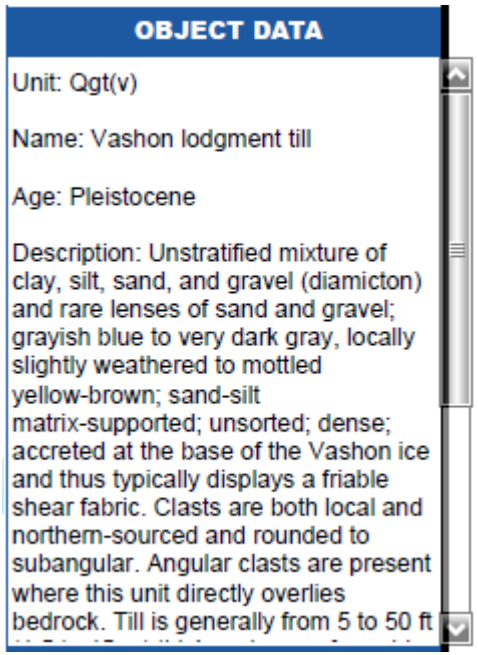
These descriptions were pulled from GIS and formatted in Excel to produce proper XML tags.

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

```
    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."
999 <MetadataItem key="Age" value="Tertiary to Holocene"/>
1000 <MetadataItem key="Name" value="Tectonic zone"/>
1001 <MetadataItem key="Unit" value="tz"/>
1002 </Metadata>
1003 <Metadata nodeName="cxb_tz(h)">
1004 <MetadataItem key="Description" value="Hydrothermally altered tect
    tz(h) contains principally low-temperature carbonate (calcite) min
1005 <MetadataItem key="Age" value="Tertiary to Holocene"/>
1006 <MetadataItem key="Name" value="Low-temperature, hydrothermally al
1007 <MetadataItem key="Unit" value="tz(h)"/>
1008 </Metadata>
1009 <JavaScript>
1010 var _ts3dhp_Attributes =  host.getField("A3DR_Text");
1011 var myarray = [];
1012 function findAndShowAttributes(node)
1013 {
1014 _ts3dhp_Attributes.value = "";
1015 if (node.metadataString != "")
1016 {
1017 myarray = [];
1018 var localxml = new XML( node.metadataString );
1019 var xItems = localxml.item;
1020 for (var j=0;xItems.length()>j;j++)
1021 {
1022 var name = xItems[j].@name.toString();
1023 var value = xItems[j].@value.toString();
1024 var attribute = name + ": " + value;
1025 myarray.push(attribute);
1026 }
1027 _ts3dhp_Attributes.value = myarray[3] + "\r\r" + myarray[2] + "\r\
1028 }
1029 }
1030 if (_ts3dhp_Attributes != null)
1031 {
1032 var _ts3dhp_PartAttributesSelect = new SelectionEventHandler();
1033 _ts3dhp_PartAttributesSelect.onEvent = function( event )
1034 {
1035 if (event.selected)
1036 findAndShowAttributes(event.node);
1037 else
1038 _ts3dhp_Attributes.value = myarray[3] + "\r\r" + myarray[2] + "\r\
1039 }
1040 runtime.addEventHandler(_ts3dhp_PartAttributesSelect);
1041 }
1042 </JavaScript>
1043 </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.

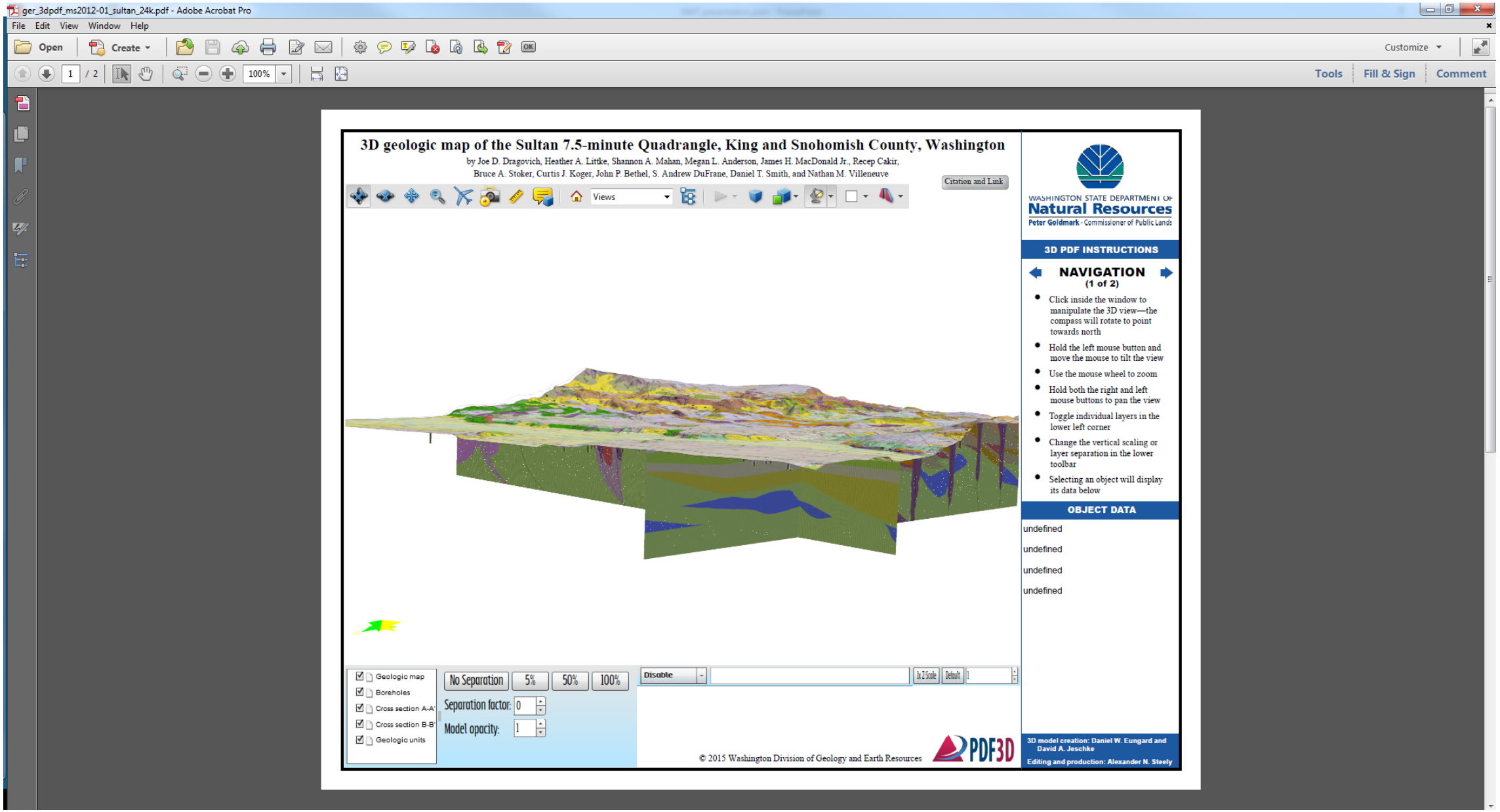
The last step for us is to add a custom Javascript which will display our data on the PDF page when a unit is clicked.



*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.