2 April 2011

General design process for converting lots of data into X3D visualizations

Focused Brainstorming

1. ***Concept***. Come up with a great idea about how your data might look when visualized in 3D. Draw some simple sketches to show exemplars of interest.

2. ***Notional examples***. Create one or two simple *X3D* models to illustrate your sketches in 3D.

3. ***Candidate examples***. Modify your *X3D* model scene to include some actual (or representative) sample data. Does this example help illustrate the relationships that you want to visualize?

Data Representations and Conversions

4. ***Correspondences***. Map the data structures of the original information to parameters within the X3D model. How are you visualizing the information important to you? What conversion or transformation functions are needed? What filtering and error checking can be applied to input data, thus avoiding garbage-in garbage-out (GIGO) problems? What boundary values, default initialization values and constraint rules can be defined for the results used in your X3D scene?

5. ***XML representation***. (Optional) If needed or available, find a conversion program that can translate the raw data into some form of XML. Can the raw data or smoothed XML be plotted using existing tools?

6. ***Brute-force conversion to X3D***. (Optional) Write a conversion program to translate raw data into custom X3D scenes that match the original example pattern. Such scenes typically have literal data placed in various nodes scattered around a scene graph. However this kind of customized scene is not easily modified afterward, so a prototype approach is preferable for large data sets.

Build X3D Prototype for Repeatability

7. ***Prototype declaration***. Wrap a *ProtoDeclare* definition around your initial model, and expose key parameters of interest as *field* definitions. Identify the key data nodes and fields used for visualization inside your X3D model. For each of these nodes and fields, define corresponding *IS*/*connect* links that utilize the parameters exposed in the *field* interface. The prototype (with connected interface and body) provides a cookie-cutter template to build many example results.

8. ***Prototype instance examples***. Build some examples by creating *ProtoInstance* copies, using actual data as *fieldValue* initializations for the prototype. Then review, refine, and repeat: modify the *ProtoDeclare* template as needed to draw your *ProtoInstance* examples correctly, producing useful visualizations. To avoid unexpected errors, ensure that all validation tests pass.

9. ***Stylesheet conversion***. Write a simple XSLT stylesheet to convert XML-ized raw-data inputs into X3D *ProtoInstance* outputs. These example outputs can simply use an *ExternProtoDeclare* to refer to the file holding the master *ProtoDeclare* template, which you can continue to improve.

10. ***Spiral improvement***. Inspect the output visualizations, and continue to refine your visualization design patterns. Note that each improvement to the *ProtoDeclare* definition gets applied equally to all of of the *ProtoInstance* examples. This separates function and presentation well: simple prototype refinements let you improve the visualization of all data examples simultaneously, without needing any brute-force regeneration of all the individual visualization models.

Addition process considerations

- Consider the raw data structures corresponding to your data.
    - Is there XML already? Is it well defined by an XML Schema or possibly an XML DTD?
    - If multiple file formats exist for the datasets of interest, then how do they compare?
    - Are the data structures sufficient for your work?

- Elaborate model view controller (MVC) design issues
    - Separation of data model, presentation view and user-interface controller simplifies evolution
    - In addition to typical typed values, other data and metadata types are needed such as images, audio/video/3D, url addresses, file names, and consistently provided annotation descriptions

- Consider what animation techniques might be suitable
    - Variations animated over time, sometimes modifiable by color or transparency selections
    - Billboard images facing user direction
    - Links from visualization components to external documents, images, videos, 3D, etc.
    - User-directed viewpoint tour, predefined animated tour

- Similar outputs might be valuable using KML for geospatial display
    - Co-evolution of complementary, mutually supporting X3D and KML mappings
    - Also some shared standards for geopositioning, time stamps, url handling, MIME types, etc.

- If your raw data is itself an X3D export from another tool, consider using that as input template for a prototype design.
    - If there is a good match, the autogenerated X3D can simply be considered as another XML input format suitable for styling.
    - Cooperative relationship with tool builders

Archival publishing considerations

- Motivation: provide the ability to perform guided visualization explorations as part of publishing datasets of interest
    - Take advantage of original data filtering and analysis techniques
    - Provide stepping-stone to future improvements or variations in analysis of the dataset
- 3D visualization is important for social network analysis
    - Can explore and understand data interrelationships in a deeper, broader way
- Software visualization tools such as Pajek, UCINet, and ORA are valuable and important
    - Can explore relationships in a deep way
    - Can capture and publish an exploration via compatible dataset, but no data standards yet exist
    - Can capture and publish explorations as 2D images and as video, which show demonstrations, but does not allow re-exploration of the data in other terms during future analysis
    - Software tools evolve, future tool availability or data compatibility is not guaranteed
- Saving as X3D allows archival publishing of 3D models
    - No longer dependent on software consistency, stability, availability
    - Can publish baseline visualization exported from tool
    - Metadata should also be included to record data origin, rights, parameters, etc.
    - Now assessing whether we can include user-directed manipulations and animations of a key visualization parameters controlling the social network being examined

Social network data visualization case study

- Choose representative exemplar dataset
- Show various toolset paths for exploring, analyzing and visualizing datasets
- Comparison of social-network data format characteristics, capabilities
- Build a case study comparing metrics of interest using each type of display, show list of capabilities (relative strengths, weaknesses) in a comparison table
  - 2D images
  - 2D+time video demonstrations
  - examinable X3D visualizations of a static 3D model
  - explorable X3D+time visualization demonstrations
- Another comparison table for 3D file-size reduction using prototype-based approach:

Conversion type	original file size for social network data	X3D export using brute-force repetitive approach	X3D export (or conversion) as prototype templates	units
ORA text (or XML) source		-	-	MB
UCINet text (or XML) source		-	-	MB
Pajek text (or XML) source				MB
.zip compression				MB
gzip compression				MB
.exi compression using Efficient XML Interchange				MB
.x3db Compressed Binary Encoding	-			MB
visualization demo: 3D graphics performance (frame rate)	-			fps

Pajek-specific tool comments

- Need to embed useful viewpoints of interest in output file
- Arcs should have parent name/comment and placed together in single Group
- DEF/USE Appearance/Material, Cone (arrowheads), possibly Cylinder if scaled by Transform
- <FontStyle family='Arial Unicode MS' should have double quotes
- Cylinder for arc (i.e. ball+stick model) is longer than needed

Acknowledgments

- Elaine Reid, Ying Zhao, Doug MacKinnon