WebGL, xtk and such

Random notes

Vedrana Andersen Dahl

January 20, 2020

Visualization is crucial when analyzing tomographic volumes, and already an initial investigation usually involves a slice-wise or a volumetric visualization. More interestingly, the steps of analysis pipeline and the obtained results should be visualized to support the scientific findings of the analysis.

Here I collected some notes about using WebGL for scientific visualization.

WebGL

WebGL `https://www.khronos.org/webgl/` is a cross-platform web standard for rendering interactive 2D and 3D graphics, without the use of plug-ins. WebGL is a JavaScript API, widely supported in modern browsers. WebGL serves as a platform for various web-based game engines.

The X Toolkit (xtk)

The X Toolkit `https://github.com/xtk/X#readme/` is a lightweight open-source JavaScript toolkit for building medical and scientific 3D visualization that runs within the browser. It uses WebGL to provide smooth 3D rendering with few lines of code.

The interactivity of xtk visualization is easily achieved by using dat.gui `https://github.com/dataarts/dat.gui`, a lightweight controller library.

For surface models, xtk supports the common formats (.stl, .vtk, .obj). A minor issue with xtk is that it supports primarily DICOM (Digital Imaging and Communications

in Medicine) volume files, very common in medical image analysis, but not in materials science. I used xtk with .nrrd (nearly raw raster data) volumes.

For easy start in using xtk, I prepared a small example for visualizing a volume and a mesh. This includes:

- Main file: index.html. In the header, this file defines page style using internal CSS `https://www.w3schools.com/html/html_css.asp`. In the body, the file includes three JavaScript scripts `https://www.w3schools.com/html/html_scripts.asp`.
- Two .js files: xtk.js and dat.gui.min.js. Those can be obtained from `https://github.com/xtk/X/` and `https://github.com/dataarts/dat.gui`. You can also find latest versions of xtk, with changed API, but I here stick to older versions which work fine for my example.
- Volumetric and mesh data in one .nrrd and two .obj files. The data in this example is saved from Matlab, and instead of providing the actual data I provide a script which produces the data and saves it using a rudimentary obj writer for meshes and nrrd writer for volumes.
- Main code: citrus.js file. In this script I define onload function which creates a xtl 3D renderer, associated objects and a dat GUI. The objects are a volume (which points to the .nrrd file) and two meshes (which point to .obj files) and those are added to the renderer. The GUI contains two folders, one interacting with the properties of the volume, and another interacting with the meshes.

In my experience, xtk works well if your needs match some of the visualization pipelines shown in the lessons `https://github.com/xtk/X#lessons/`, but if you want to extend or modify the functionality, you may spend a lot of time trying.

Visualization Toolkit (vtk.js)

The Visualization Toolkit (VTK) JavaScript library `https://kitware.github.io/vtk-js/` is an open-source library for scientific visualization on the web. It uses WebGL and supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods.

It should be able to suppors .vtp geometry, .obj and .mtl meshes, .vtkjs scenes exported by ParaView and .vti volumes. I did not test this.

Browser issues

For security reasons, your internet browser will probably prevent loading WebGL content from your local file system. Instead of changing your security policy, you can run a local server, e.g. Python's built-in http server.

1. From the command line, run `python -m http.server` (this is for Python 3.x).
2. In a browser, open `http://localhost:8000/` and navigate to your file.

Source, and more info: `http://www.acute3d.com/local-webgl/`

Another issue is that your browser will probably attempt loading a cached version of your web side, ignoring recent changes you may have made. You need to check how to force your browser to load a fresh content. In Crome on mac, hold down `Shift` while reloading.

Debugging

Figuring out what is wrong with your JavaScript and/or html code can be really, really difficult. You normally only see that the output is not as intended – usually an empty window. To read the error messages, activate debugging in your browser (F12 key in Chrome) and look at the console. You can also display JavaScript values in the console, to assist debugging. Source, and more info: `https://www.w3schools.com/js/js_debugging.asp`

Meshes and xtk

A few additional notes on preparing the meshes for visualization using xtk – here I do not consider volumetric data.

Firstly, I tried using meshes in .obj and .vtk format. It turns out that those two formats get displayed differently in xtk – despite the exactly same mesh geometry (vertices and faces). My conclusion is that for .obj files normals get computed for flat shading (i.e. every vertex is copied for each face it contributes to, and face normal is assigned to all of face points), while for .vtk files normals get computed by normalizing vertex coordinates. This gives a smoother, but wrong, appearance when using .vtk files.

Secondly, and in connection to the first comment, for best smooth shading we would like per-vertex normals (and smooth appearance when using wrong .vtk normals suggests that interpolation works well). Hovewer, even when .obj file contains per-vertex normals, xtk does not seem to load these! Not to mention per-vertex color. After a whole Saturday trying to assign color to mesh vertices, I gave up. My last success before giving up involved setting a vector of per-point scalars (a point is collection of triangle points, every mesh vertex occurs multiple times in points), and then color would be interpolated between the two rgb values according to scalars.

Thirdly, I experienced problems when setting up-vector for xtk (both x and z direction seem to give the same orientation). I ended up fixing the size and orientation of the mesh in MeshLab `http://www.meshlab.net/`. For this I used options under

Filters → Normals, Curvatures and Orientation → Transform: various options. Here you can set origin so that the mesh is centered on bounding box, scale to unit box (I would then upscale again by a fixed factor), and swap or flip axis to get the desired orientation. (For some reason, trying to export the transformed mesh would sometimes save the original mesh, I don't know why and when this happened.) The initial view of MeshLab is the same as the view of xtk when you in xtk set up-vector to $(0,1,0)$ and camera position to $(0,0,z)$.

Gallery

Here are links to my xtk-based visualizations.

- A minimal example accompanying this note can be seen by following a link under WegGL and such `http://people.compute.dtu.dk/vand/notes.html`
- Visualization of peripheral nerves extracted from volumetric data `http://people.compute.dtu.dk/vand/nerves/`
- Some skulls `http://people.compute.dtu.dk/vand/skulls/`
- An interactive bust of Hans Christian Ørsted `http://hco.compute.dtu.dk/`