**Final report for work performed under BPA Task 41 IT Development**

**3/31/2013**

First, I downloaded, configured, and built the latest version of the COLLADA DOM SDK package. This went fairly smoothly, but some subtle build bugs were introduced here that wasted a lot of time later on. The DOM SDK later turned out to have a bug of its own that I had to track down and fix as well.

I then used the previously-developed CATPart-to-VRML module as a starting point to develop the CATPart-to-COLLADA module. This module converts a single CATPart into a COLLADA document and stores it in a .dae file. This was more challenging than anticipated because the COLLADA DOM API has little documentation and few examples. I ended up using the OpenSceneGraph COLLADA importer code as a reference to figure out how to use the COLLADA DOM API.

After creating this basic converter, a test environment was needed. It turned out that I had no programs that could open a COLLADA file. I had assumed that PolyTrans supported this, but the version we own does not. Next, I attempted to use an OpenSceneGraph-based viewer but it had problems. I spent several days trying to fix this viewer. It turned out that several different build issues with the COLLADA DOM was causing OpenSceneGraph to crash when attempting to view the COLLADA files.

Once I had a working viewer, I was able to rapidly test my changes to the exporter. This accelerated the fine tuning of the exporter with basic geometry and simple materials.

I spent the final week adding texture mapping to the exporter. I had already solved the major mathematical problems with CATIA texture mapping in the VRML exporter, but the COLLADA format added additional complexity.

Because of the way CATIA materials are applied, the texture mapping code has to be able to apply texture coordinates to already-existing meshes. This means that the texture mapper must be able to read the geometric subset of the COLLADA schema. This turned out to be very complex because COLLADA documents have a very flexible layout and contain a large amount of indirection. Also, COLLADA meshes can contain polygon lists, triangle lists, triangle strips, and triangle fans. This meant that the texture mapper had to be able to remap all four different types of meshes. As a result, the texture mapper module required as much code as the part exporter itself.

Completed functionality:

* Export a single CATPart to a single COLLADA document file.
* Support CATIA graphic properties (color, transparency, hide-show)
* Support CATIA materials (ambient, diffuse, specular, transparency)
* Support all CATIA texture mapping types (planar, cubic, cylindrical, spherical)
* Support all mesh primitive types (polygon, triangle, triangle fan, triangle strip)

Missing functionality when compared to the VRML exporter:

* Export CGR and transient geometry like pipes and ergo-men.
* Export an entire CATIA assembly to COLLADA.
* Copy unique texture map images to the same directory as the COLLADA document.
* Take advantage of multi-core workstation during export to export parts in parallel
* Create a cache of already-converted COLLADA files to reduce export times

The CGR/transient item will be fairly quick to implement because it shares quite a bit of code with the CATPart exporter.

The other items will be easy to port from the VRML exporter because there is very little file-format-specific code at that level.

As the work with the DON exporter built up the capabilities of the VRML exporter, I anticipate the work on the TUT exporter will do the same for the COLLADA exporter. I am ready to provide Wyck with some example files when he starts to implement his importer for these files.