What's so Hot About Alembic?

an industry op-ed by Steve Wright

Every so often a real game changer comes along for visual effects and animation. In 2003 it was OpenEXR released by ILM for the entire world to enjoy. In 2010 ILM did it again by teaming up with Sony Picture Imageworks to release yet another game changer - Alembic geometry. This mysterious new CGI creature promises to revolutionize our production workflow, but how? And what is Alembic anyway?

Alembic is a file format for exchanging 3D computer animation information between different platforms and applications. Any app can write their 3D information out in the Alembic format which can then be read in by any other app that supports Alembic, such as Nuke, my personal favorite. Beyond that, it stores 3D geometry in a format that is very efficient with a small file size that is quick to read and unpack. But wait – there's more! It also allows you to load just the frames you want or just the bits and pieces you want. You don't have to load the entire database for the whole robot if you just want to deal with the head.

Alembic distills the scene to meshes with their vertices (also called a point cache) that can be easily read in by the next application. It is perfectly analogous to lighting and rendering scenes out to an image file that can be displayed by any application that reads that image file format. A Renderman scene can only be loaded and viewed in Renderman, but the rendered exr file can be loaded and viewed by Nuke, Photoshop, After Effects, or any other app that reads exr files.

The way it does this is to "bake out" just the vertices of the geometry so you have the results of complex procedural animation without the complex procedures themselves. So a rigged character that includes deformations, for example, is baked out to just the character's points for each frame, but without the rig. If the geometry has no deformations, just simple transforms, then only one instance of the geometry need be saved along with the transform information. Very compact.

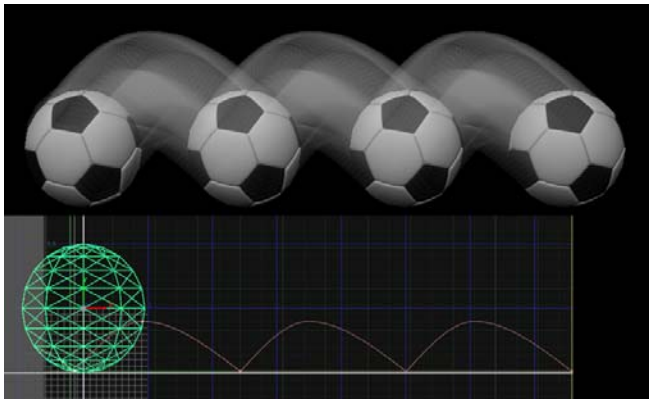


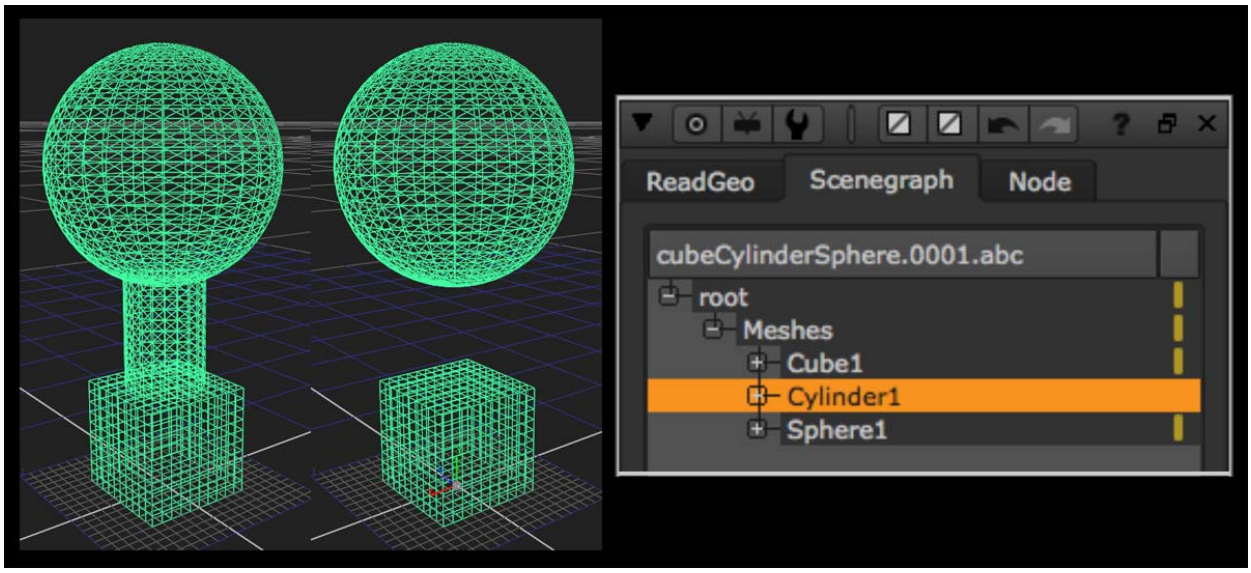
Figure 1 – a simple case

This approach introduces great efficiencies and time-savers in the CGI production pipeline. It creates an asset-based parallel CGI pipeline so that multiple groups can be working on the same scene elements at the same time. It also makes it far simpler to pass work between groups (or facilities) eliminating the need to convert data between platforms. To date, Nuke's version of Alembic supports meshes, their vertices, normals, UVs, and vertex colors, plus particles, point clouds and cameras. No lights or materials yet, but the format is still young and in development.

Figure 1 illustrates the Alembic representation of a scene in its simplest form. The top part of Figure 1 shows the shot we want to make, a bouncing soccer ball. The bottom part illustrates the Alembic representation of that scene, just one mesh with vertices for the ball plus the transforms required to make it bounce. Not only is it a very small file size, but a 1000 frame shot would hardly be any larger than a 24 frame shot.

Note that the transforms are in a verbose frame-by-frame format rather than just a few keyframes defining a spline path of motion. The reason for this is that every application would draw the spline somewhat differently but verbose frame-by-frame transformations are easily defined and exquisitely repeatable. The geometry and animation will be identical regardless of which app uses it.

One of the other key features of the Alembic format is illustrated in Figure 2, the ability to just load selected parts of the database. This dramatically lowers data transfer and load times as well as render times. The wireframe model on the left shows the entire elegant structure on which we are working. However, if we don't need the middle cylinder we can just deselect it and it is never fetched, loaded, or rendered. How this is done is illustrated using Nuke's implementation called a "Scenegraph". Each object is listed in the "inventory list" and you just disable those components you don't want at the moment. In this example the second wireframe model is missing the cylinder because it was disabled in the Nuke Scenegraph.



all geometry **cylinder not loaded** **cylinder disabled in the Scenegraph**

Figure 2

In the real world of production where you are dealing with more complex 3D structures than my elegant example here the gains are huge. Let's say you are working with an articulated robot. The Scenegraph will display a hierarchical list of all the fiddly bits of your robot so that you could, for example, turn on just the head which would include the eyes, mouth, and death ray. Or you could click off the head and just turn on the eyes, or just the left eye.

The key here is that you only pay for what you actually use in memory space, file transfer time, and render time. This is not true of the FBX file format, the other 3D scene file interchange format. With FBX you must load the entire database even if you want to deal with just one part. You would then turn off the unneeded parts in the application after having suffered the overhead of loading them all in. Better to not load them in the first place. Further, because FBX supports more sophisticated representations of the 3D scene there are interpretation differences between applications that introduce problems that Alembic does not have.

Another cool attribute of the Alembic format is subframe support for motion blur. Let's say the geometry is in one location on frame 3 and another on frame 4. You could, for example, load the geometry on frame 3.2, 3.4, 3.6, and 3.8 to render motion blur. The Alembic format will return these in-between subframe positions automatically without intervention from the application program.

My favorite application, Nuke, now supports Alembic geometry as of Nuke 7.0. I am sure your favorite application will too as this is a real game changer in the CGI production pipeline introducing efficiencies and inter-platform operability that we all can use when creating visual effects shots and animation.

Links

Here's the link to the official Alembic web site: http://www.alembic.io

However, the real action seems to be here at the Alembic Google group:

http://code.google.com/p/alembic/

Steve Wright