An Open Source Framework For Visual Effects Software Development

John Haddon* Image Engine
Carsten Kolve† Dr D Studios
Roberto Hradec‡ Radical

Abstract

Cortex [1] is an open source framework applicable to a broad gamut of software development for visual effects. It can be distinguished from projects such as Alembic [2] and GTO [3] in that it provides a cross-application framework for computation and rendering in addition to file IO. Cortex has undergone continuous development at Image Engine since 2006, forming the foundation for work on feature films such as District 9 and Twilight Eclipse. It has recently seen further use and development at a number of other studios.

1 Basic Functionality

Cortex builds upon the popular Imath and Boost C++ libraries, adding higher level data types such as curves, meshes and images, along with a suite of algorithms for operating on those types. It provides support for reading and writing many common graphics file formats and provides native formats for object serialisation and random access caching. Additionally, conversions are provided between Cortex types and the 3rd party APIs for Maya, Nuke, Houdini, RenderMan and OpenGL. Most functionality is implemented in C++ for speed, but also made available in Python via a thin layer of bindings. This is of particular relevance to TDs, and allows for the rapid development and deployment often necessitated by real world production scenarios.

2 Write Once, Deploy Anywhere

A key feature of Cortex is that it allows code to be written once using native types, and then deployed in multiple 3rd party applications without change. Such code comes in two principal forms - ops, which calculate a result given some inputs, and procedurals, which describe geometry and shading through a renderer agnostic interface. In both cases inputs are described as parameters, which define a required type (e.g. an integer), a set of additional constraints (e.g. an allowable range) and further information such as user help and UI hints. Different host layers then map these inputs to a form appropriate for a given application. A simple command line host maps inputs to command arguments and Maya, Nuke and Houdini hosts map them to attributes on native nodes, performing automatic conversion between the host and Cortex data types. This makes ops and procedurals usable in many different environments.

*e-mail: john@image-engine.com
†e-mail:carsten.kolve@drdstudios.com
‡e-mail:rhradec@radical.ca

3 Case Studies

3.1 Lighting and Rendering

In Image Engine's lighting pipeline all geometry is represented as Cortex procedurals. These pull in cached models and animation, and combine them with layered shaders and other appearance modifiers. Look development consists of building such procedurals dynamically from an extensible toolbox of components. Visualisation in Maya is achieved by evaluating procedurals through an OpenGL translation layer and rendering with 3delight uses an equivalent RenderMan layer. Recent developments allow procedural hosting in Nuke, with rendering via 3delight embedded in a custom node.

3.2 Crowd Simulation

Dr D Studios are using Cortex for crowd layout, simulation, visualisation and rendering. Cortex extensions allow large volumes of motion capture to be represented and cached efficiently. Layouts are generated procedurally using scripted ops hosted in Maya, with Cortex features such as mesh evaluation and fast neighbour queries helping with terrain matching, clumping and collision detection. Visualisation is achieved using procedurals hosted in Houdini, and these also perform geometry instancing and skinning for rendering with 3delight. The geometry can also be output at runtime within Houdini to drive environment interaction effects.

3.3 Fur and Hair

Image Engine's fur and hair system builds directly upon the Cortex C++ libraries for best performance. Seeds are generated on meshes using high quality point distributions and curves are then variously grown, groomed, clumped, frizzed and mutated using algorithms sitting atop Cortex's geometry and image processing routines. This system is multithreaded and happily processes, displays and renders hairs on the order of tens of millions for multiple creatures.

3.4 Baking Lightmaps

Cortex is used at Radical to perform baking of volumetric ambient occlusion and bounce light for all game geometry. The process is triggered any time a change to geometry, lighting or texture is published. Extensions to Cortex read in the assets and use the RenderMan support layer to describe them to 3delight. Here the lighting is computed before being stored in a custom encoded 8bit RGBA image. The automated system replaces a previously manual process, reducing baking times from 14 hours to approximately 30 minutes.

4 Summary

Cortex has been used successfully in a broad range of production scenarios. We hope it will find further uses in the wider community.

5 References

[1] Cortex Project Website, http://code.google.com/p/cortex-vfx
[2] Alembic Project Website, http://www.alembic.io
[3] GTO Project Website, http://code.google.com/p/open-gto