Equalizer White Paper, April 2010

OpenSceneGraph and Equalizer

Parallel Rendering for OpenSceneGraph

OSGScaleViewer is a scalable OpenSceneGraph viewer based on the Equalizer parallel rendering framework, providing the ideal basis for building scalable high-performance 3D applications using multiple GPU's.

Introduction

The **OpenSceneGraph** (OSG) is an open source high performance 3D graphics toolkit, used widely in the vis-sim, space, scientific, oil-gas, games and virtual reality industries.

Equalizer is the standard middleware for parallel rendering. It enables OpenGL applications to benefit from multiple graphics cards, processors and computers to scale rendering performance, visual quality and display size.

OSGScaleViewer is an example application integrating OSG and Equalizer. Its purpose is to demonstrate the optimal approach for this integration and to serve as a basis for developing scalable, high-performance OSG applications.

This White Paper describes the features of OSGScaleViewer and expands on various implementation scenarios for more complex real-world applications.

OSGScaleViewer

OSGScaleViewer is a scalable, cluster-ready viewer application based on OpenSceneGraph and Equalizer. It demonstrates how to integrate OSG and Equalizer optimally to exploit the power of multi-GPU systems and visualization clusters. It supports:

- Rendering **multiple views** of the same scene graph, using software or hardware **swap synchronization**.
- Parallel, multithreaded rendering on **multi-GPU** workstations and distributed rendering on multi-GPU **visualization clusters**.
- **Scalable rendering** to aggregate the power of multiple GPUs for one or multiple views (supported modes: 2D, DPlex, 2D load-balancing, cross-segment load-balancing)
- Any combination of the above due to the flexible **run-time configuration** of Equalizer, using a simple configuration file.

Parallel Architecture

The architecture of the OSG viewer is designed for optimal performance and parallelization, while providing a straightforward programming model to developers.

Each **process** in the cluster is represented by an **eq::Node**. The node instantiates one copy of the scene graph in Node::configInit. Animation updates and other scene graph modifications are applied at the beginning of each frame in Node::frameStart.

Each **GPU** on a node is represented by an **eq::Pipe**. Each GPU runs its own rendering **thread** in parallel to the other pipes and the node main thread.

Each on-screen and off-screen **OpenGL drawable and context** is abstracted by an **eq::Window**. To render the scene graph, a customized SceneView is used. The SceneView is initialized in eq::Window::configInitGL(). Only the first window of a GPU, the shared context window, performs this initialization, while other windows reuse the scene view of their shared window.

All **rendering** operations happen in an **eq::Channel**, which represents a **2D viewport** in an eq::Window. Channel::frameDraw, which performs the actual rendering, sets up the SceneView with the rendering parameters provided by Equalizer and triggers a cull and draw traversal to render a new frame.

The rendering operations of all pipe threads on a single node are frame-synchronized, that is, they are synchronized with each other and the node's main thread. Node processes and compositing operations for scalable rendering run asynchronous to this synchronization for optimal performance. Figure 1 depicts a

Screenshots courtesy of OmniTerra (http://www.omniterra.ch)

OSGScaleViewer

A flexible and scalable foundation for developing high-performance image generators based on leading Open Source technologies.

possible synchronization when using scalable rendering to render one view using four GPU's.

Data Distribution

Equalizer provides simple, yet powerful data distribution based on the serialization of C++ objects. The objects are versioned, allowing efficient updates based on delta serialization. Object versions can easily be tied to rendering frames, keeping the database consistent across all rendering processes in a cluster.

For large clusters Equalizer provides optionally a reliable multicast implementation which efficiently distributes data to many cluster nodes.

OSGScaleViewer uses distributed objects to initialize the model filename on all processes and to synchronize the camera data. This usage is very similar to the eqPly polygonal rendering example and described in depth in the Equalizer Programming and User Guide.

OSGScaleViewer does not need to implement a mechanism to distribute changes on the scene graph itself. The following sections outline different approaches which can be taken to implement distributed OSG updates.

Scene Graph Updates

The OSG viewer is a **multithreaded** rendering application, which applies all scene graph modifications in the main thread, between rendering frames. This is the preferred design pattern for optimal performance and memory usage. The default thread synchronization of Equalizer simplifies this implementation. All scene graph changes are done in the node process at the beginning of a frame, before the pipe rendering threads are unlocked, as shown in Figure 1.

Some applications tightly integrate data updates with the rendering traversal. In this case, a **multiprocess** approach may be used. For each GPU, a separate rendering process using an Equalizer node, is instantiated. This provides protection against conflicting data updates, but increases memory requirements.

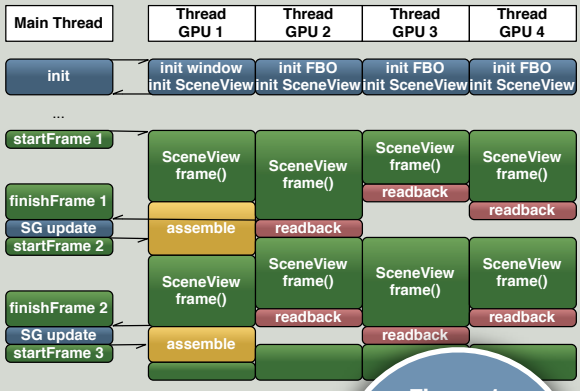

Figure 1: Asynchronous rendering tasks when updating one view using four GPU's

The multithreaded approach of osgScaleViewer fully exploits multi-GPU systems for multi-view and scalable rendering. All data modifications are done in a thread-safe manner and are directly available to the rendering threads through shared memory.

For graphics clusters applications need to implement a mechanism to update the different scene graph instances on all cluster nodes. In the following sections, two different paradigms to synchronize the databases are outlined.

Event Distribution

The least invasive, but code intensive and more fragile approach, is to apply the same operations to each scene graph instance. The initial scene graph is available to all processes, e.g., through a shared file system.

The relevant commands to modify the scene graph are distributed to all processes in the cluster. The FrameData object of the osgScaleViewer is the ideal place for this, since it provides frame-specific data correctly to all nodes. Each node reads and applies the commands in Node::frame-Start to keep the data consistent.

Data Distribution

Data distribution for OSG nodes is the most versatile and robust approach. The application process holds the master instance of the scene graph. Any changes on the scene graph are tracked internally. At the beginning of each frame, the application commits all pending changes. The render clients receive a change list and update the data in Node::frameStart.

Several design patterns can be employed to implement data distribution. Subclassing, proxies or multiple inheritance are the most common (Figure 2). The **Equalizer Programming and User Guide** describes the characteristics of the different approaches and data distribution in detail.

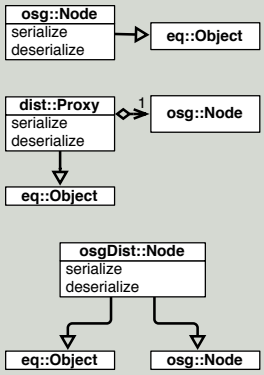

Figure 2: Subclassing, proxies and multiple inheritance for OSG data distribution

Conclusion

OpenSceneGraph and Equalizer are the ideal combination for high-performance 3D visualization. While OSG focuses on rendering efficiently, Equalizer provides scalability on multi-GPU systems and large-scale clusters. OSGScaleViewer is a template for applications combining the two open source solutions in an efficient way.

About

OpenSceneGraph is a product of OpenSceneGraph Professional Services. Please visit www.openscenegraph.org for more information.

Equalizer is a product of Eyescale Software GmbH. Please visit www.eyescale.ch and www.equalizergraphics.com for more information.

OSGScaleViewer is available as part of the Equalizer open source distribution. The Equalizer Programming and User Guide can be downloaded from the Equalizer website or ordered as a hardcopy from lulu.com.

Screenshots courtesy of OmniTerra (http://www.omniterra.ch)