CAVE Rendering Framework

Brigitte Hulliger, Jonas Walti, Stefan Broder

September 22, 2009

Agenda

Initial setup

- CAVE
  - 3 walls
  - 2 projectors per wall

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Initial setup

- CAVE
  - 3 walls
  - 2 projectors per wall
- Previous solutions
  - Chromium
  - VRML/X3D

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Task description

The goal of this bachelor thesis is the development of a CAVE Rendering Framework for C++/OpenGL applications. The framework should be built on common libraries for scene-graph based rendering such as OpenSceneGraph (www.oppensceengraph.org) or the OGRE Game Engine (www.ogre3d.org). For distributed rendering one of these libraries needs to be integrated with the Equalizer Library (www.equalizergraphics.com). The goal is to develop a CAVE Rendering Framework for simplified implementation of C++/OpenGL CAVE applications.

Goals

- Primary goals
  - Integrate a scene graph in Equalizer
  - Stereoscopic output
  - OpenGL based solution
  - Performant scene graph rendering
  - Extensibility
  - Configurability

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Goals

- Primary goals
  - Integrate a scene graph in Equalizer
  - Stereoscopic output
  - OpenGL based solution
  - Performant scene graph rendering
  - Extensibility
  - Configurability
- Secondary goals
  - Volume rendering
  - Haptics
  - Shader
  - Physics
  - Tracking

Milestones

- *M1*: Functional specification

Milestones

- *M1*: Functional specification
- *M2*: Prototype
  - Test cases
  - Project setup
  - Techniques

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Milestones

- *M1*: Functional specification
- *M2*: Prototype
  - Test cases
  - Project setup
  - Techniques
- *M3*: Development
  - Design concept
  - Implementation

Milestones

- *M1*: Functional specification
- *M2*: Prototype
  - Test cases
  - Project setup
  - Techniques
- *M3*: Development
  - Design concept
  - Implementation
- *M4*: Testing
  - Test reports
  - Demo & test application(s)

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Milestones

- *M1*: Functional specification
- *M2*: Prototype
  - Test cases
  - Project setup
  - Techniques
- *M3*: Development
  - Design concept
  - Implementation
- *M4*: Testing
  - Test reports
  - Demo & test application(s)
- *M5*: Documentation
  - Thesis report
  - Technical documentation
  - User guide & programming guide

Responsibilities

- *Brigitte Hulliger*
  - Equalizer

Responsibilities

- *Brigitte Hulliger*
  - Equalizer
- *Jonas Walti*
  - Implementation

Responsibilities

- *Brigitte Hulliger*
  - Equalizer
- *Jonas Walti*
  - Implementation
- *Stefan Broder*
  - Infrastructure

Eurographics '09 Munich

- 2-4 days

Eurographics '09 Munich

- 2-4 days
- Knowledge extension

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Eurographics '09 Munich

- 2-4 days
- Knowledge extension
- Social networking

Eurographics '09 Munich

- 2-4 days
- Knowledge extension
- Social networking
- **BOF meeting Equalizer**

Eurographics '09 Munich

- 2-4 days
- Knowledge extension
- Social networking
- **BOF meeting Equalizer**
- **eqOSG meeting**

BOF meeting Equalizer

Participants

- Stefan Eilemann, Equalizer
- University of Zurich
- University of Siegen
- Berne University of Applied Sciences
- ...

Subject

- Equalizer: Past, present and future
- *Virtual architecture with Equalizer and OpenSceneGraph*
- Performance optimizations for image compositing

eqOSG meeting

Participants

- Stefan Eilemann, Equalizer
- University of Zurich
- University of Siegen
- Berne University of Applied Sciences

Subject

- Meet & greet
- Experience exchange
- Planning of long term project

Infrastructure - CAVE topologies

- One-node setup
  - 1 node
  - 4 graphics cards
  - 8 GB memory

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Infrastructure - CAVE topologies

- One-node setup
  - 1 node
  - 4 graphics cards
  - 8 GB memory
- Six-node setup
  - 6 nodes
  - 1 graphics card per node
  - 2 GB memory per node

One-node setup

Figure: One-node setup

Six-node setup

Figure: Network setup (6 nodes)

Equalizer configuration files

- One-node configuration
  - Same physical workstation for server, application & render clients
  - Starting Equalizer server (`eqServer`)
  - Starting CRF application

Equalizer configuration files

- One-node configuration
  - Same physical workstation for server, application & render clients
  - Starting Equalizer server (`eqServer`)
  - Starting CRF application
- Six-node configuration
  - 6 nodes connected via SSH
  - 1 node acts as server
  - Application & data on each node
  - Starting Equalizer server (`eqServer`)
  - Starting CRF application on server node $\longrightarrow$ server starts application on all connected nodes

Equalizer configuration

config	Configuration for a server
node	Abstraction of a workstation
connection	Point-to-point connection between nodes
pipe	Abstraction of a graphics card
window	Abstraction of a OpenGL drawable
channel	Viewport within a *window*
compound	An ordered collection of tasks for a channel. Can have compound children.

Equalizer one-node configuration

Figure: One-node Equalizer configuration

Equalizer six-node configuration

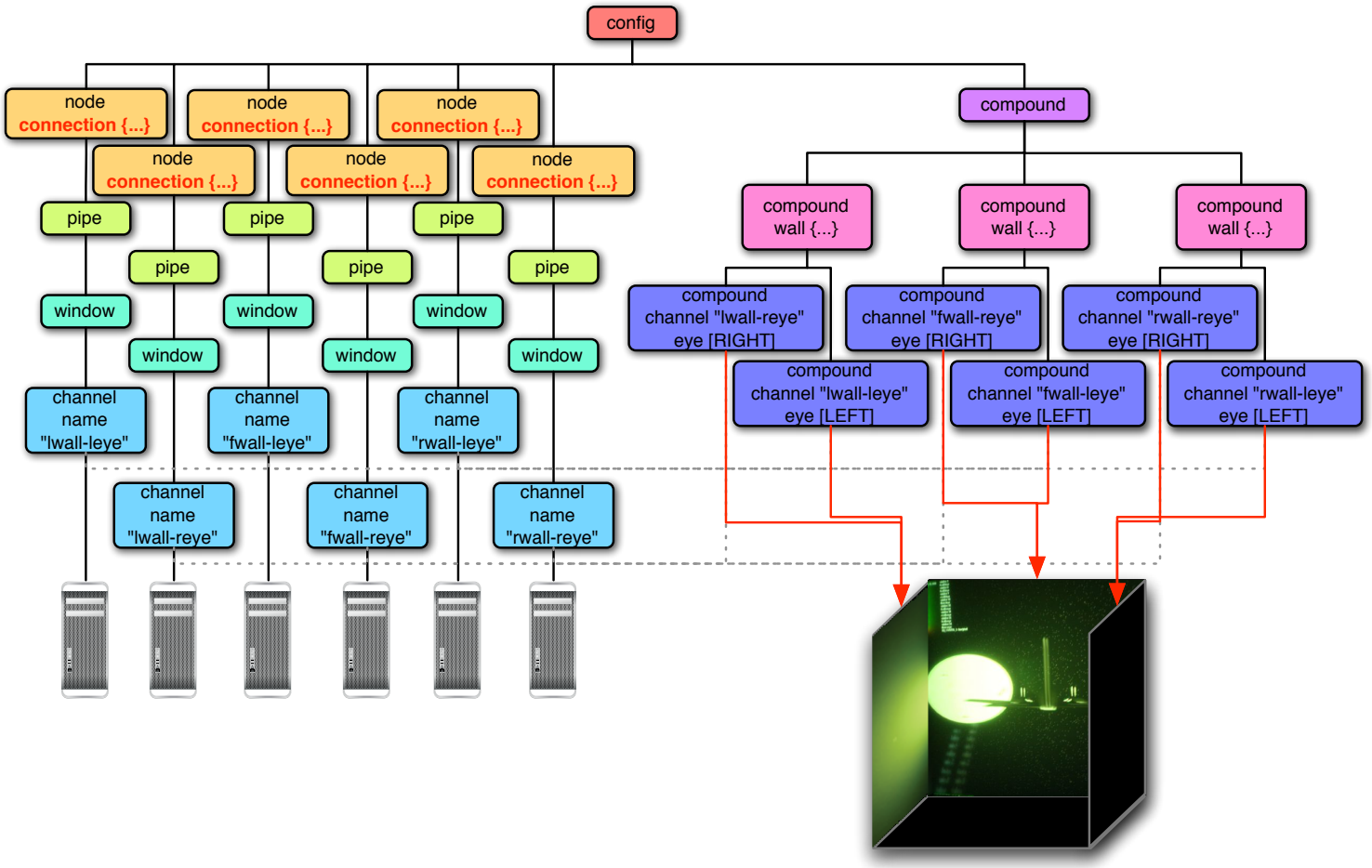

Figure: Six-node Equalizer configuration

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Equalizer conclusion

- +/- Work in progress
- \+ Configurability
- \+ Active community
- \- Combination of different hardware architectures may be a problem (32/64-bit)
- \- High maintenance for network distributed solutions

Framework - overview

- Synchronisation

Framework - overview

- Synchronisation
- Event handling

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Framework - overview

- Synchronisation
- Event handling
- Use of framework

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Framework - overview

- Synchronisation
- Event handling
- Use of framework
- Conclusion

Synchronisation

- One OSG viewer per Equalizer pipe

Synchronisation

- One OSG viewer per Equalizer pipe
- Each of these viewers holds a complete scene graph

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Synchronisation

- One OSG viewer per Equalizer pipe
- Each of these viewers holds a complete scene graph
- Synchronisation has to be respected
  - Frame based animations
  - Time based animations
  - Event based animations

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Event handling

- Equalizer event handling
  - Camera handling
  - Statistics

Event handling

- Equalizer event handling
  - Camera handling
  - Statistics
- Event propagation
  - Selected Equalizer events are distributed
  - Currently: Mouse- and most keyboard events
  - Converted on every Equalizer pipe

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

Event handling

- Equalizer event handling
  - Camera handling
  - Statistics
- Event propagation
  - Selected Equalizer events are distributed
  - Currently: Mouse- and most keyboard events
  - Converted on every Equalizer pipe
- OSG event handler
  - OSG viewer(s) receive native OSG events
  - Thus, common OSG event handlers can be used

Use of framework

- Abstraction
  - Just basic Equalizer skills needed
  - OSG applications can be easily ported to the CRF
  - Just a few limitations

Use of framework

- Abstraction
  - Just basic Equalizer skills needed
  - OSG applications can be easily ported to the CRF
  - Just a few limitations
- Facade pattern
  - Hides the Equalizer initialisation from the CRF user
  - Only two lines of code needed for a basic example

Use of framework

- Abstraction
  - Just basic Equalizer skills needed
  - OSG applications can be easily ported to the CRF
  - Just a few limitations
- Facade pattern
  - Hides the Equalizer initialisation from the CRF user
  - Only two lines of code needed for a basic example
- Overriding framework functions
  - For more complex applications
  - Well documented with examples
  - Unlimited possibilities for further framework extensions

Framework conclusion

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

- Testing!
  - Testing in real environments
  - C++ is not easy!
  - Multithreading can cause unexpected behaviours or even errors
  - Black box tests for more improvements

Framework conclusion

Brigitte Hulliger,
Jonas Walti,
Stefan Broder

- Testing!
  - Testing in real environments
  - C++ is not easy!
  - Multithreading can cause unexpected behaviours or even errors
  - Black box tests for more improvements
- Extensibility
  - Numerous new features possible
  - This is just a basic solution

(Un-)Achieved goals

- Primary goals
  - Integrate a scene graph in Equalizer ✔
  - Stereoscopic output ✔
  - OpenGL based solution ✔
  - Performant scene graph rendering ✔
  - Extensibility ✔
  - Configurability ✔

(Un-)Achieved goals

- Primary goals
  - Integrate a scene graph in Equalizer ✔
  - Stereoscopic output ✔
  - OpenGL based solution ✔
  - Performant scene graph rendering ✔
  - Extensibility ✔
  - Configurability ✔
- Secondary goals
  - Volume rendering ✖
  - Haptics ✖
  - Shader ✔
  - Physics (✔)
  - Tracking ✖

Personal summary

- OSG vs. OGRE
  - OSG: Unorganised documentation
  - OSG/OGRE: Big community

Personal summary

- OSG vs. OGRE
  - OSG: Unorganised documentation
  - OSG/OGRE: Big community
- Preparation project
  - Setup of a real test environment
  - Gather basic knowledge first

Personal summary

- OSG vs. OGRE
  - OSG: Unorganised documentation
  - OSG/OGRE: Big community
- Preparation project
  - Setup of a real test environment
  - Gather basic knowledge first
- Time consuming knowledge preparation