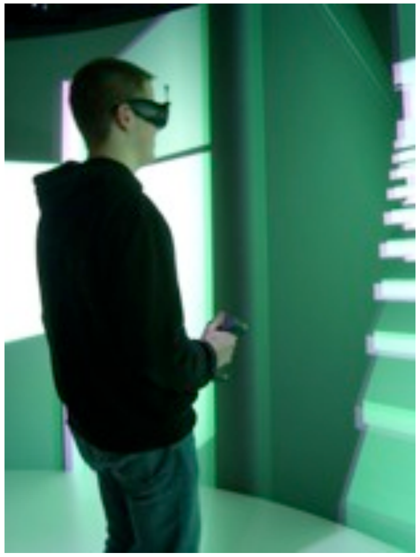
Virtual Architecture with Equalizer and OpenSceneGraph

Our team

- 11 computer science students of the University of Siegen
- Project work is a part of our study
- Duration of one year
- Many of us plan to specialize in field of computer graphics
- Four representatives of our project group are here to present our work

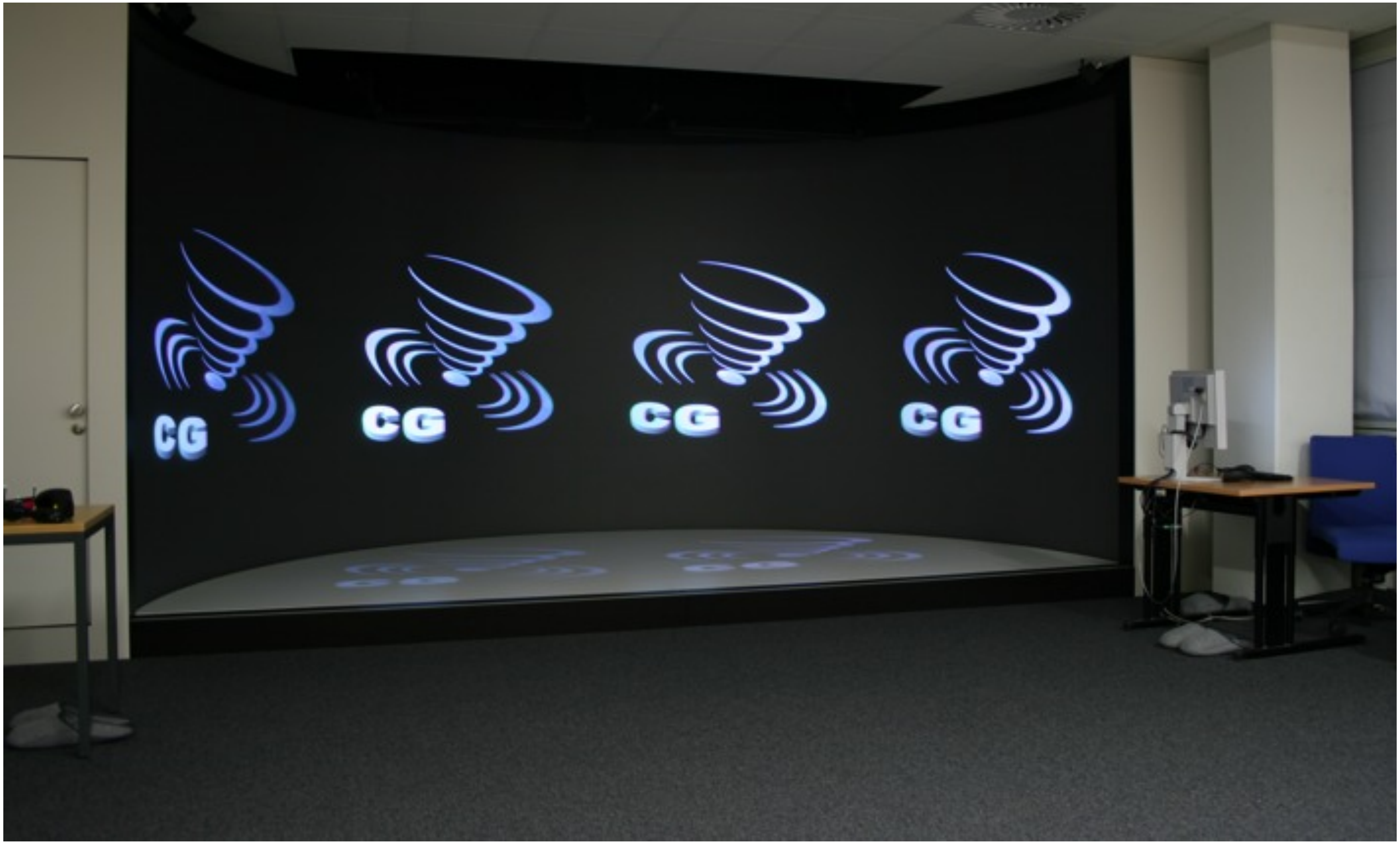
Our project

- Interactive navigation and visualisation of architectural data
- Real-time manipulation of the model structure like adding furniture or moving windows
- Collaboration with the faculty of architecture at the University of Siegen
- We visualize the data in the institute's virtual reality lab



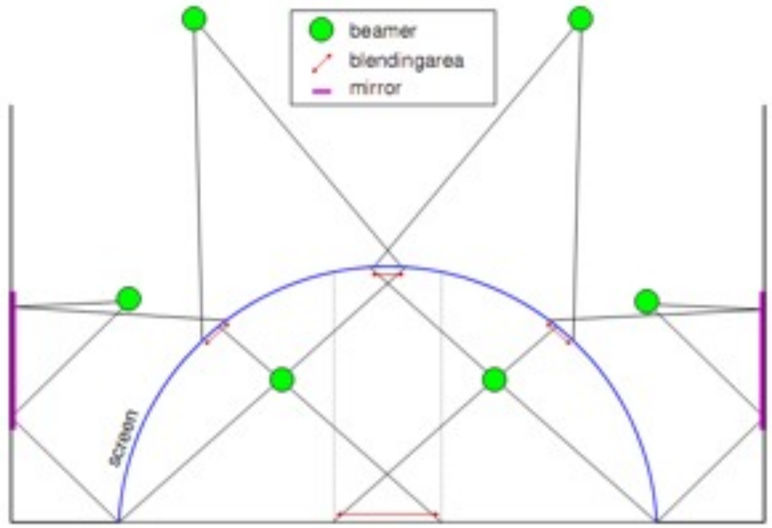
Our lab at the University of Siegen

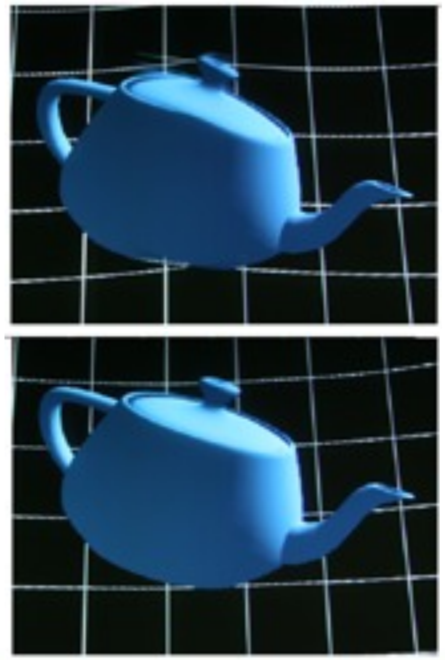
- Half cylinder with rear projection surfaces on the wall and front projection on the floor
- Unique installation: Suitable for single tracked users or presentations
- Stereo separation is done with Infitec glasses (colour separation)
- One beamer per stereo channel, 6 projection surfaces in total (12 beamers)

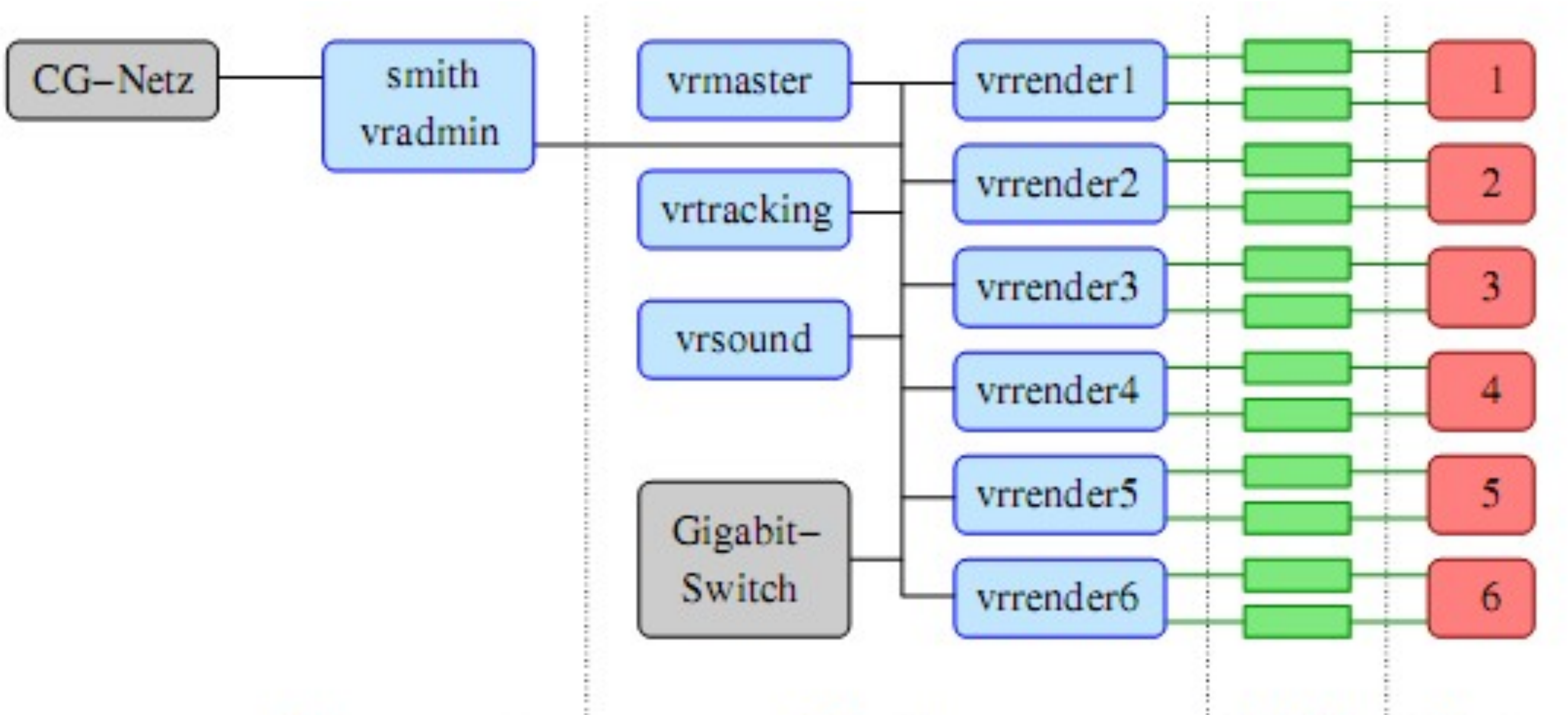

CG
CG
CG
CG

Our lab at the University of Siegen

- 6 render clients, each two graphic cards (two stereo channels each)
- Mirrors allow efficient use of available space
- Cylindrical surface creates distortion effects
  - Static warping using CineIPM video processors
- Problem: IPMs calibrated for fixed sweet spot
  - Dynamic warping for tracked users in fragment shader



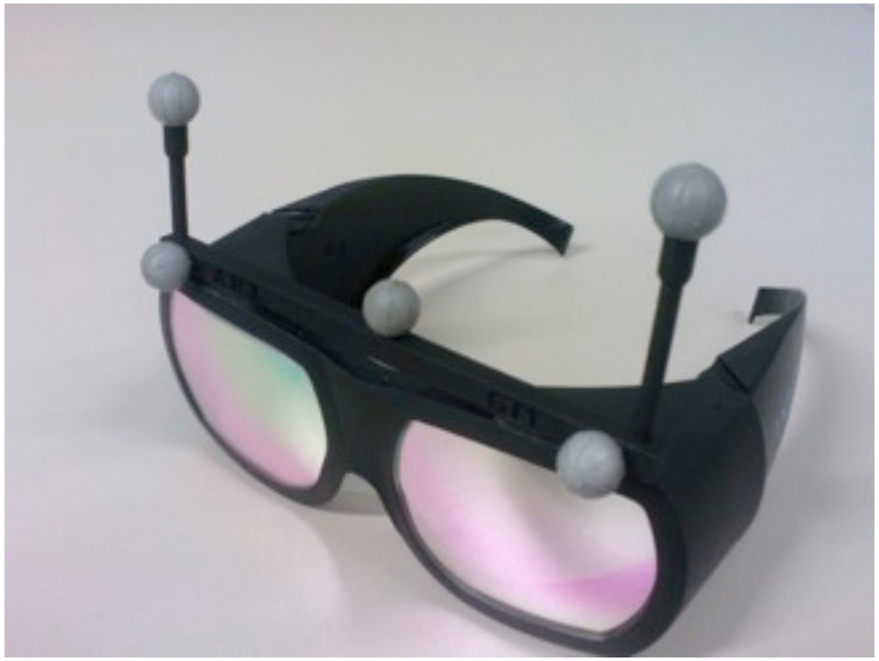




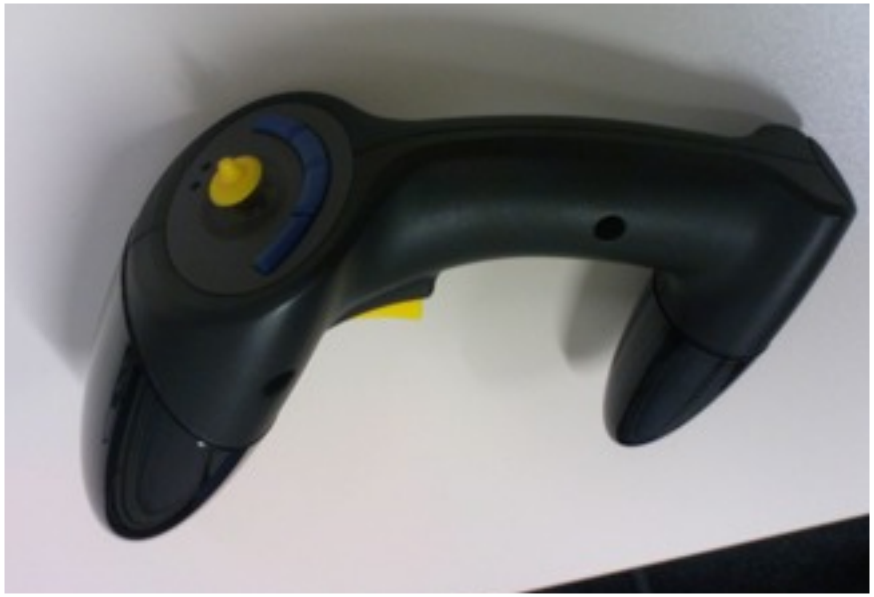
Control Cluster IPM Beamer pairs

Tracking

- ○ Infrared reflection of markers (A.R.T.)
- ○ Recorded by four IR-cameras



Head Tracking



Navigation

Why have we chosen Equalizer?

- Best VR framework for our purpose
- No restriction for OpenGL programming
- Supports distributed rendering (essential for the VR-lab)
- Easy configuration for different systems (VR-lab and single-user PC)
- Operation system independent (Linux, Windows and OS X)
- Active development and support
- Open Source

Why do we use OpenSceneGraph?

- Offers all the functionality we need
  - Lots of different file formats are supported
  - Very flexible and extensible
- Large community (tutorials, documentation, ...)
- Open Source and platform independent, like Equalizer

Integration between Equalizer and OpenSceneGraph

- Tell OSG to use context provided by Equalizer
  - ➢ Now OSG doesn’t create its own window

```
Osg::Camera->setGraphicsContext( new OsgViewer::GraphicsWindowEmbedded );
```

- Pass the viewport and frustum information to OSG

- ▪ Viewport:

```
const eq::PixelViewport& viewport = eq::Channel->getPixelViewport();
Osg::Camera->setViewport( 0, 0, viewport.w, viewport.h );
```

Integration between Equalizer and OpenSceneGraph

- Frustum:

```
const vmml::Frustumf& frustum = eq::Channel->getFrustum();
osg::Camera->setProjectionMatrixAsFrustum( frustum.left,
     frustum.right, frustum.bottom, frustum.top,
     frustum.nearPlane, frustum.farPlane );
```

In Channel::frameDraw(), multiply view matrix with head matrix

```
osg::Matrix headView = getViewMatrix();
headView.postMult( getHeadTransform() );
osg::Viewer->getRenderCamera()->setViewMatrix( headView );
```

Multipass Rendering

- One osg::Viewer per Channel renders scene to texture
- Warping shader corrects distortion in texture
- Texture is rendered onto quad

Distribution of the scene graph

○ No special decomposition used
- ➢ Complete scene graph is available on each node

○Shared file system to load the same model on all nodes

○FrameData is used to pass camera position to the nodes

○Nodes are told via the FrameData about dynamic updates and have to manipulate the scene graph themselves
- ➢ like "Place model 'table.3ds' at position (x, y, z)“

○Collision detection runs on the application node
- ➢ render clients really do only rendering

eqOSG

- Snapshot of earlier VIA version, now in Equalizer SVN
  - Minimal example code based on the above principles
  - As generic as possible *(no head tracking, no warping shaders, etc.)*
  - We hope to extend eqOSG and collaborate with others
  - For example :
    - Camera controls or collision detection
    - Support for dynamic scene graph updates

Thank you for your attention.

Are there any questions?

http://www.cg.informatik.uni-siegen.de/Teaching/ProjectGroups/VIA