Commodity Components for High Resolution and Large Displays

Bruno Raffin

Laboratoire ID

France

Bruno.raffin@imag.fr

Laboratoire Informatique et Distribution

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Institut National Polytechnique de Grenoble

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

GRENOBLE 1 UNIVERSITÉ JOSEPH FOURIER SCIENCES,TECHNOLOGIE,MÉDÉCINE

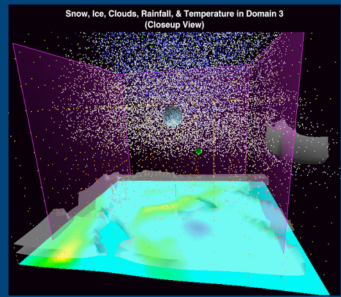
Virtual Reality 1/2

- 1965: Ivan Sutherland (MIT) :

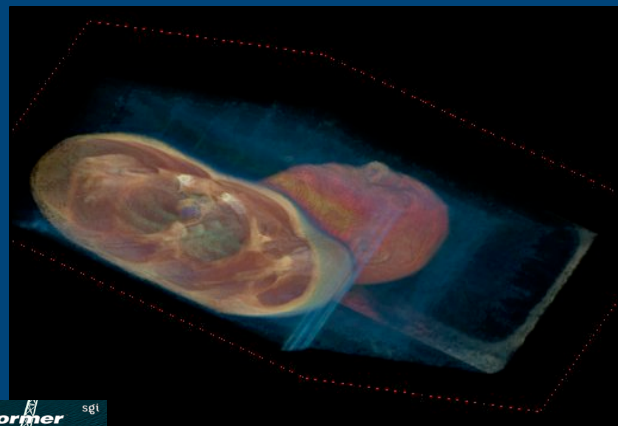
  Concept of immersion in a simulated world

  ➡ Technology not ready at that time

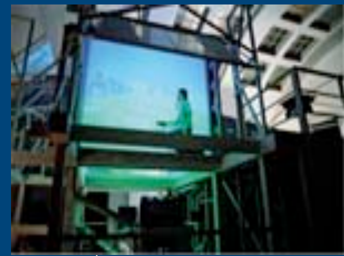
Virtual Reality 2/2

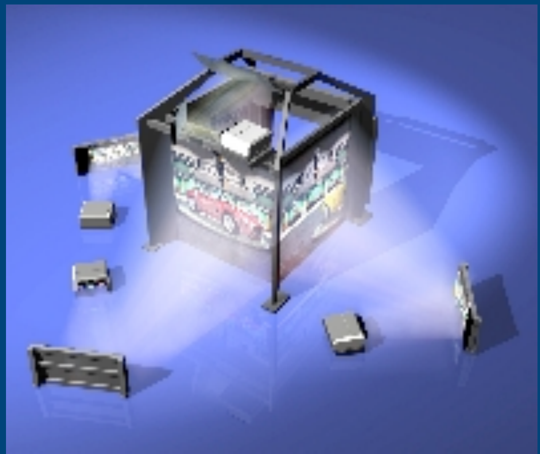


- **Today's applications:**
  - Scientific visualization
  - Engineering
  - Design, architecture
  - Games, education



Immersive Environment 1/2





- **SigGraph'92 : First Cave (Cruz-Neira)**

  -> 4 sided Cave

- **Today a few 6 sided Caves in the world** (Sweden, Germany, Japan, USA)

- **Full visual immersion**

  -> The cubic structure disappear with the stereo

- **Collaborative work possible but limited (restriction on stereo)**

- **Cost and space requirements**

Immersive Environment 2/2

- 93: First Holobench (GMD - Germany)
- The user dominates the scene rather than to be immersed
- Collaborative work limited
- Limited cost and space requirements

Appreciated for engineering Work

Classical VR Components

- Output devices:
  - High-end CRT or DLP stereo projectors
  - Active stereo glasses (LCD shutters)

- Input devices:
  - 3D tracker
  - 3D mouse or Wand

- Computer
  - Dedicated graphics supercomputers
  - Multi CPU and multi GPU
  - Classically an SGI Onyx

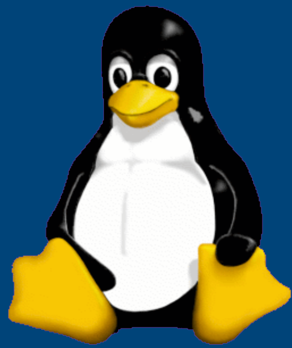
Dedicated Components

- Specifically developed for virtual reality
  - Meet the expected performance
  - Have all expected features (built-in genlock,...)
- High development and production costs, SMALL MARKET
  - High cost
  - Slow renewal rate

Commodity Components

- High development and production compensated by a HUGE MARKET
  - Low cost
  - Fast renewal rate
- Standard
  - Better interoperability between components
- But not designed for the use we target
  - -> some features may be missing

Software is the Key



- **Idea: to develop software to compensate for the missing features.**

- **Relies on Open Source software:**
  - Access to source code at no cost
  - Already large variety of software available (Linux)

"Easy" to improve an existing piece of software

Classical VR Components

- Output devices:
  - High-end CRT or DLP stereo projectors
  - Active stereo glasses (LCD shutters)

- Input devices:
  - 3D tracker
  - 3D mouse or Wand

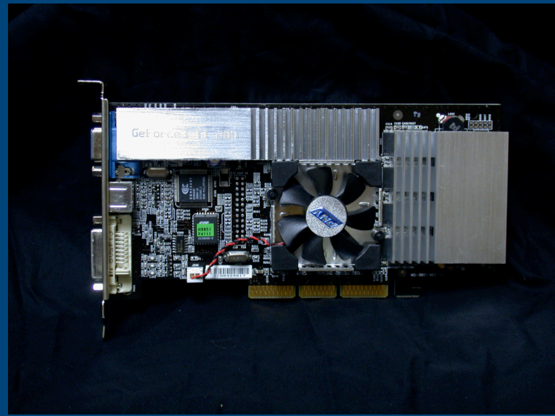
- Computer
  - Dedicated graphics supercomputers
  - Multi CPU and multi GPU
  - Classically an SGI Onyx

PC Cluster

- PCs + local network
- Started with Beowulf Project, at NASA, in 1994.
- Today common for high performance computing
- Clusters up to 10 000 nodes

I-cluster at Lab ID

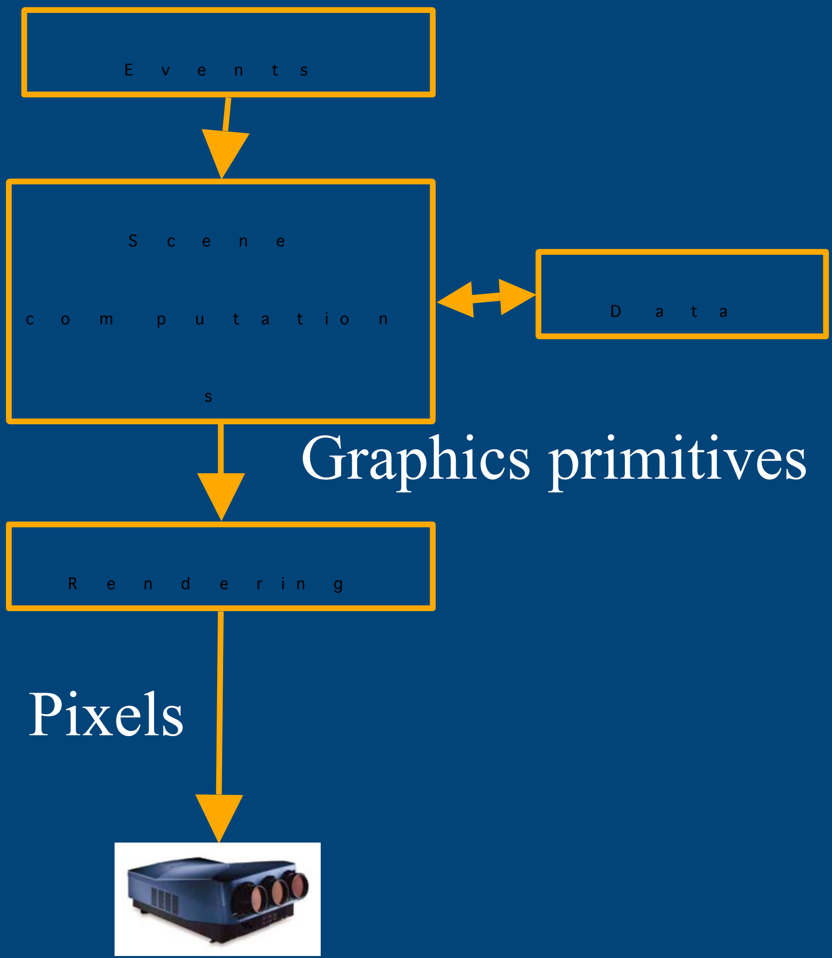
PC Cluster for VR



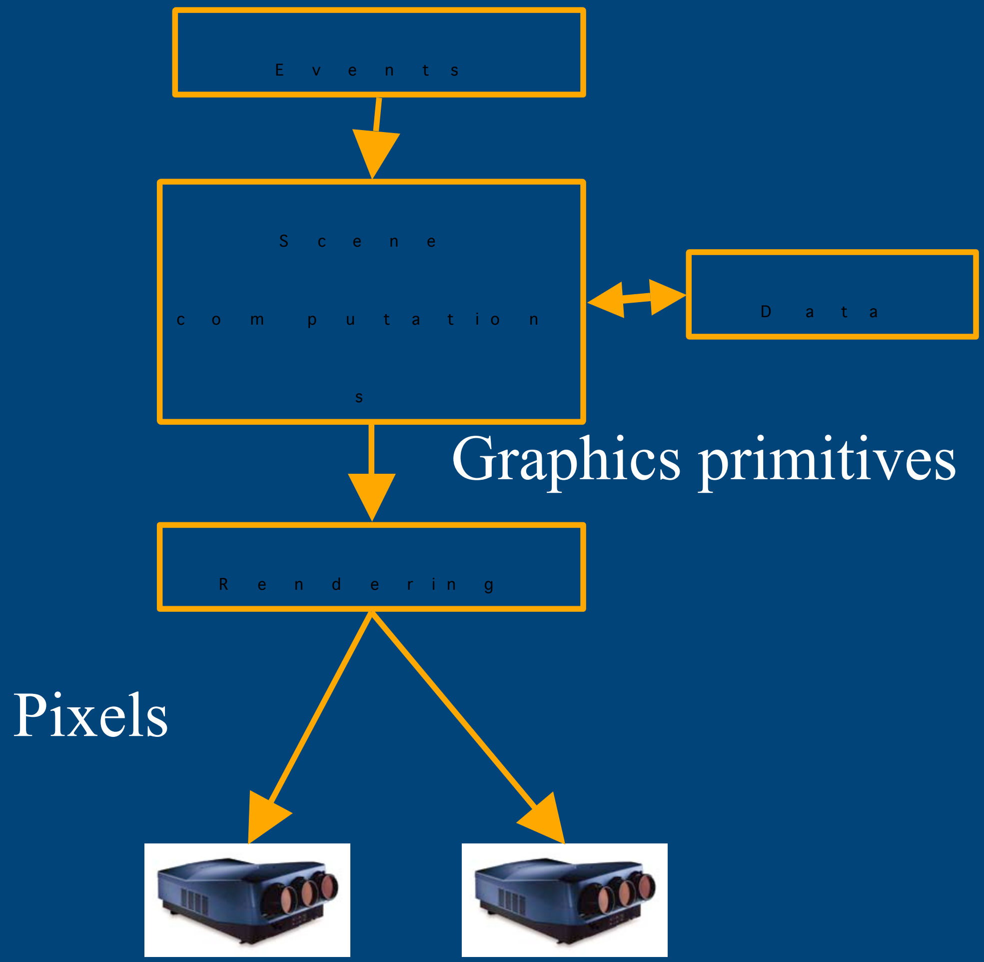
- **What is particular in VR:**
  - **Interactive 3D graphics**
    - Add 3D graphics cards

      Benefits from the boom of graphics cards for gamers (Nvidia geforce 6800 Ultra)
  - **Multiple displays**
    - Need for image distribution and several synchronization levels (application, swaplock, genlock) … but not available on a PC cluster

One PC – One display

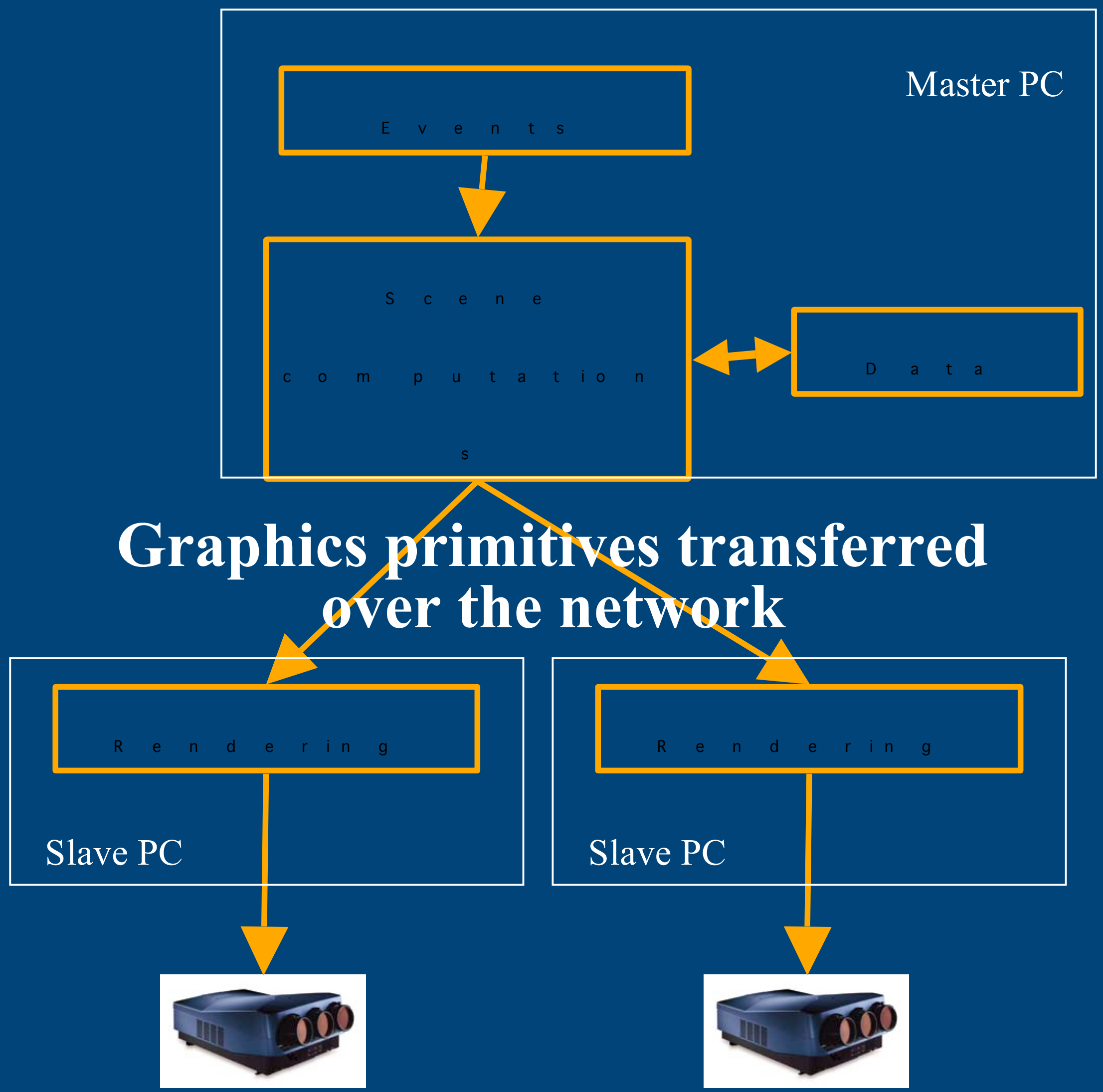


One PC – N displays



- **Very simple but not scalable**
  - Limited by the number of graphics boards supported by a PC
  - One machine carry all computations

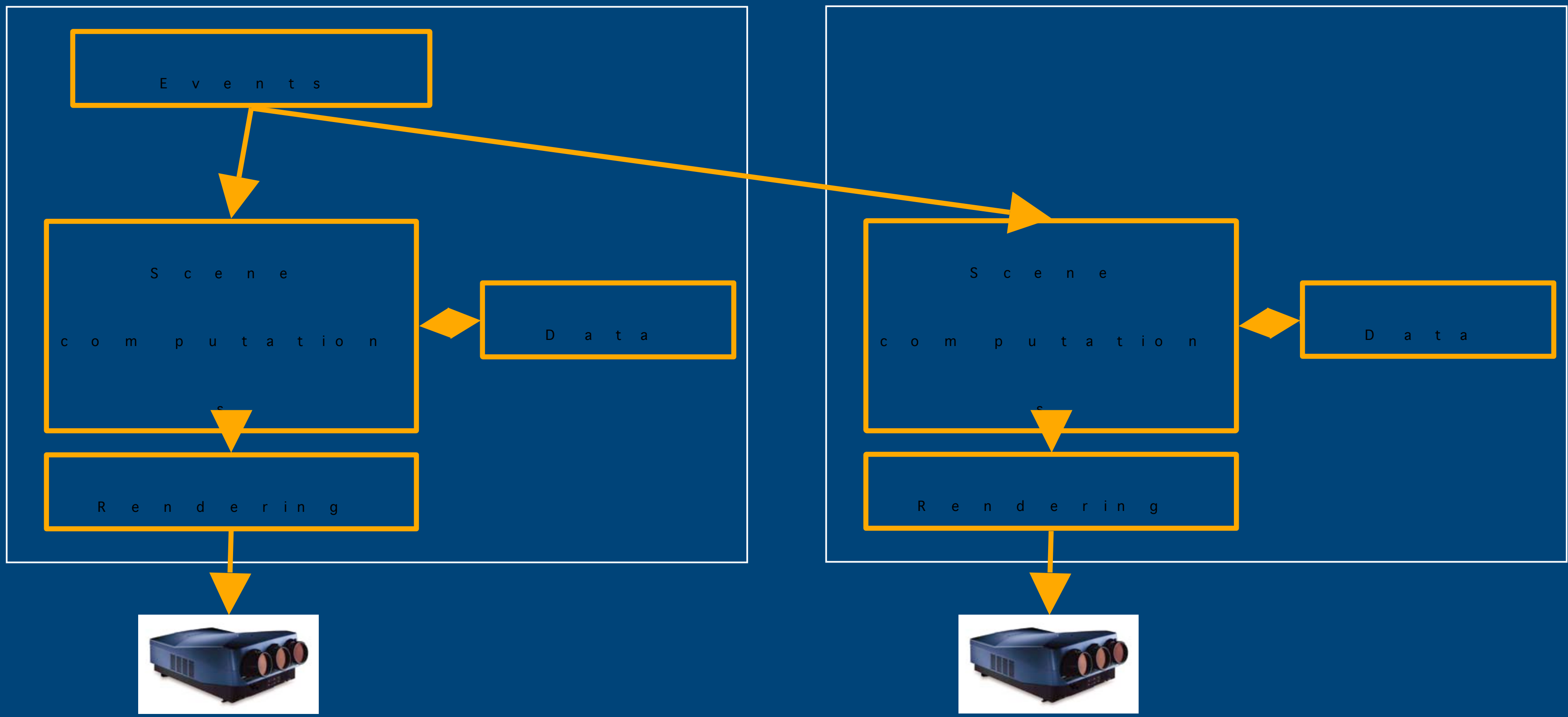
N+1 PCs – N displays



+ Portability
+ Distribute the rendering work
- Important network load
- One machine carry all scene computations

Chromium

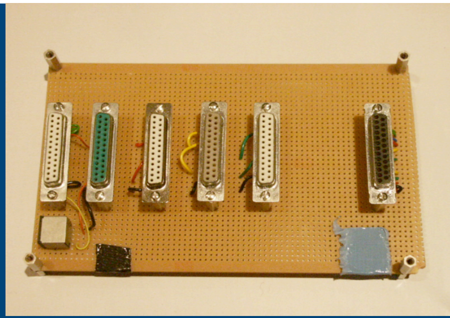
N PCs – N displays



+ Low network load
- Portability
- Data and some computations duplicated

Net Juggler, VR Juggler, Syzygy, Amira cluster,…

Swaplock and Genlock



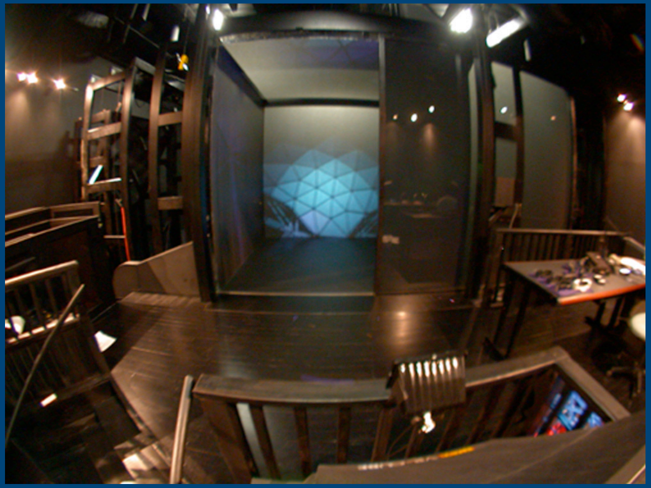
- Genlock: to synchronize the video signals
- Swaplock: to synchronize the frame buffers publication

- Available on some high-end graphics cards (NVIDIA 3800G)
- Commodity component approach:
  - Swaplock: synchronization barrier using the cluster network
  - Genlock: a parallel port based network and "genlock" linux drivers (SoftGenLock)

VR Cluster

- Software solutions for the missing features
- VR Clusters match (or even outperform) the performance of dedicated machines for a fraction of the cost

- Today most people will replace their SGI Onyx by a PC cluster



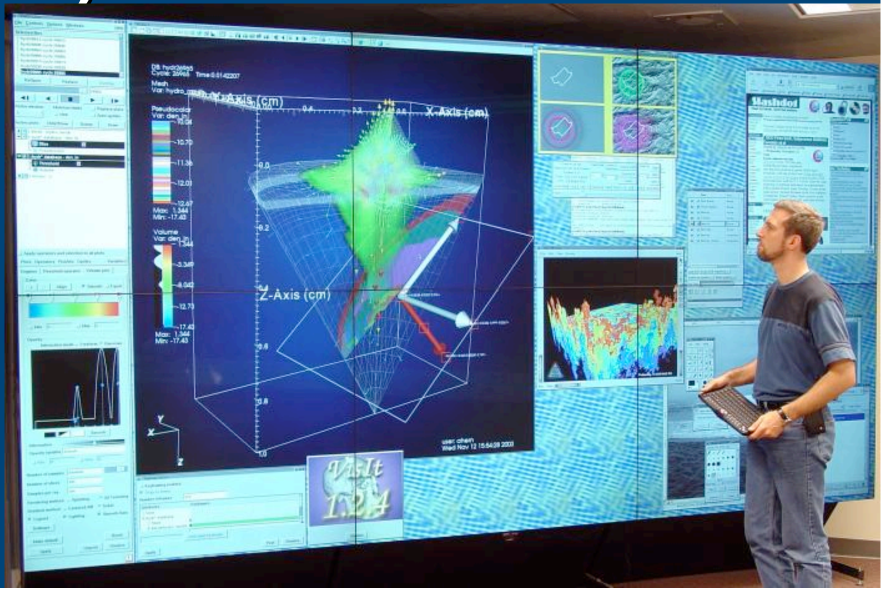
University of urbana-Champaign

University of Orléans

Legacy 2D Application

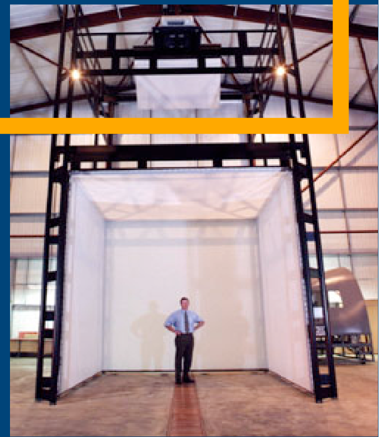
Goal: to execute 2D applications on a display wall without modification or recompilation.

- **A proxy for the X server to hide the display wall specificity from clients :**
  - **X Proxy (J. Verduzsco)**
  - **DMX**



Classical VR Components

- Output devices:
  - High-end CRT or DLP stereo projectors
  - Active stereo glasses (LCD shutters)



- Input devices:
  - 3D tracker
  - 3D mouse or Wand

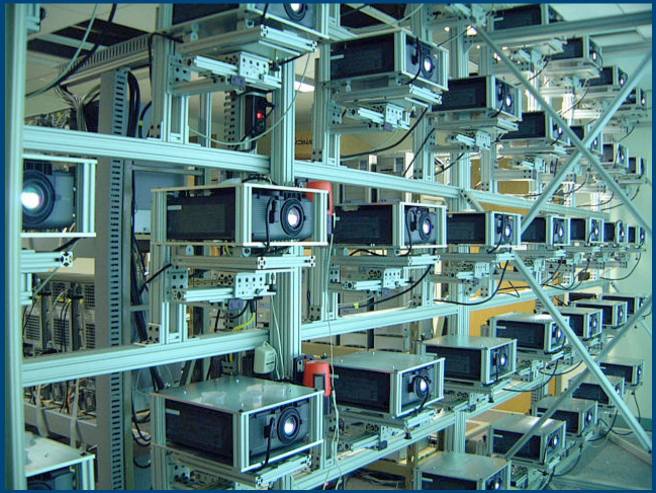
- Computer
  - Dedicated graphics supercomputers
  - Multi CPU and multi GPU
  - Classically an SGI Onyx

VR Projectors

- Immersive environments use few VR projectors that are
  - High resolution
  - High brightness
  - Good color and brightness uniformity

  But
  - Expensive
  - Heavy (> 60 kg) and large
  - Not that bright (cannot work in day light)

Commodity Projectors



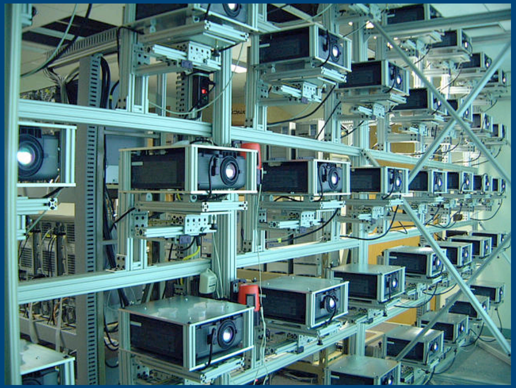
Today projectors of
- 1024x768 pixels
- 1500 – 3000 lumens
- 2 – 10 kg

are inexpensive (2000 – 6000 euros)

Idea: tile many of them to build a display of:
- very high resolution
- high brightness (work at day light)

NCSA: 40 commodity projectors – Resolution: 8192x3840 pixels

Calibration Issues



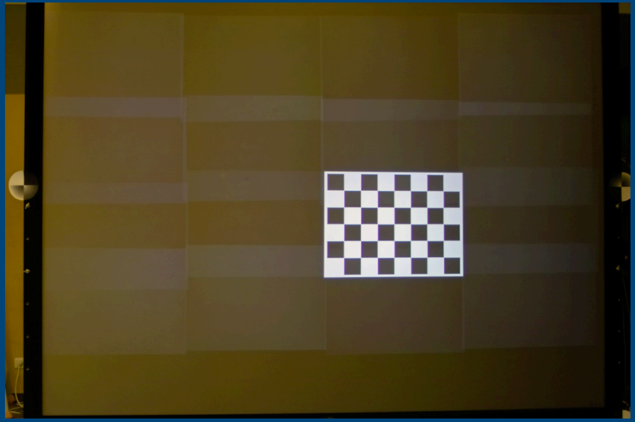
- Geometry: projector alignment
- Color: color uniformity
- Brightness: brightness uniformity

Difficult, almost impossible to calibrate by hand

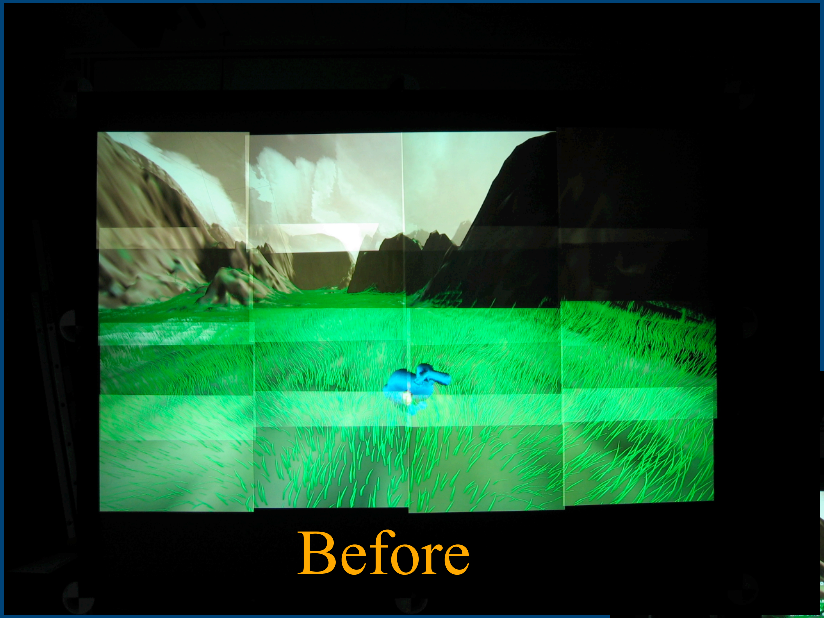
Idea: use an artificial eye (camera) and mathematics (in a software) for automatic calibration

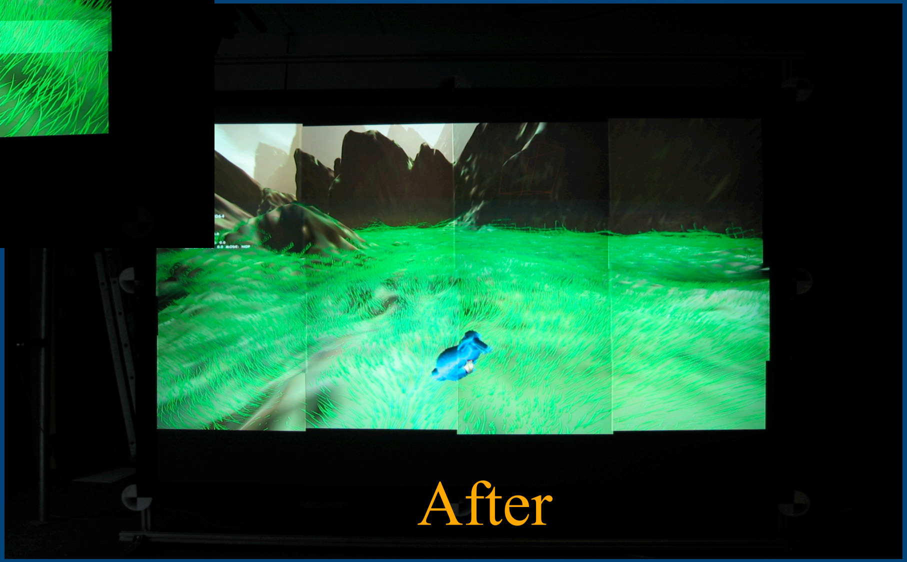
Software Calibration

- Re-use techniques from artificial vision
- Compute image transformations from picture analysis (transformation matrices and attenuation masks).
- These matrices and masks are applied by the graphics cards



Software Calibration

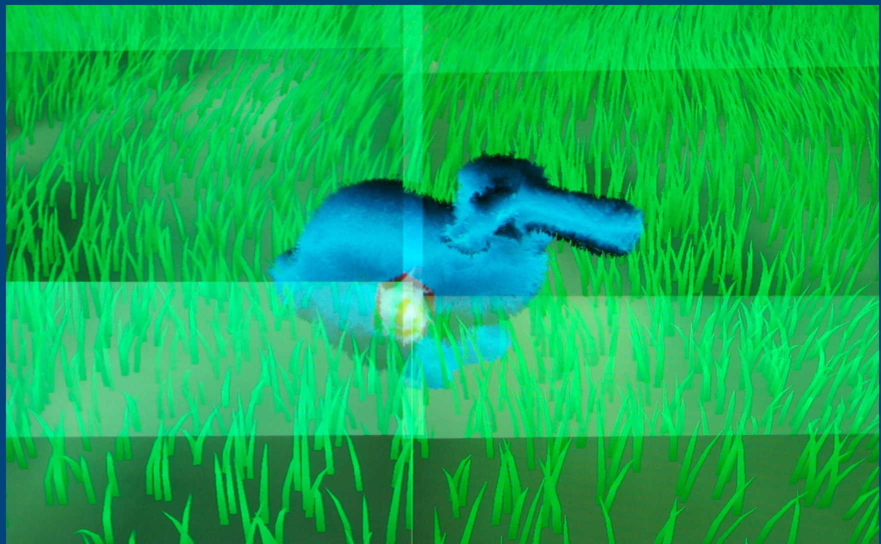




Geometry and brightness

Software Calibration

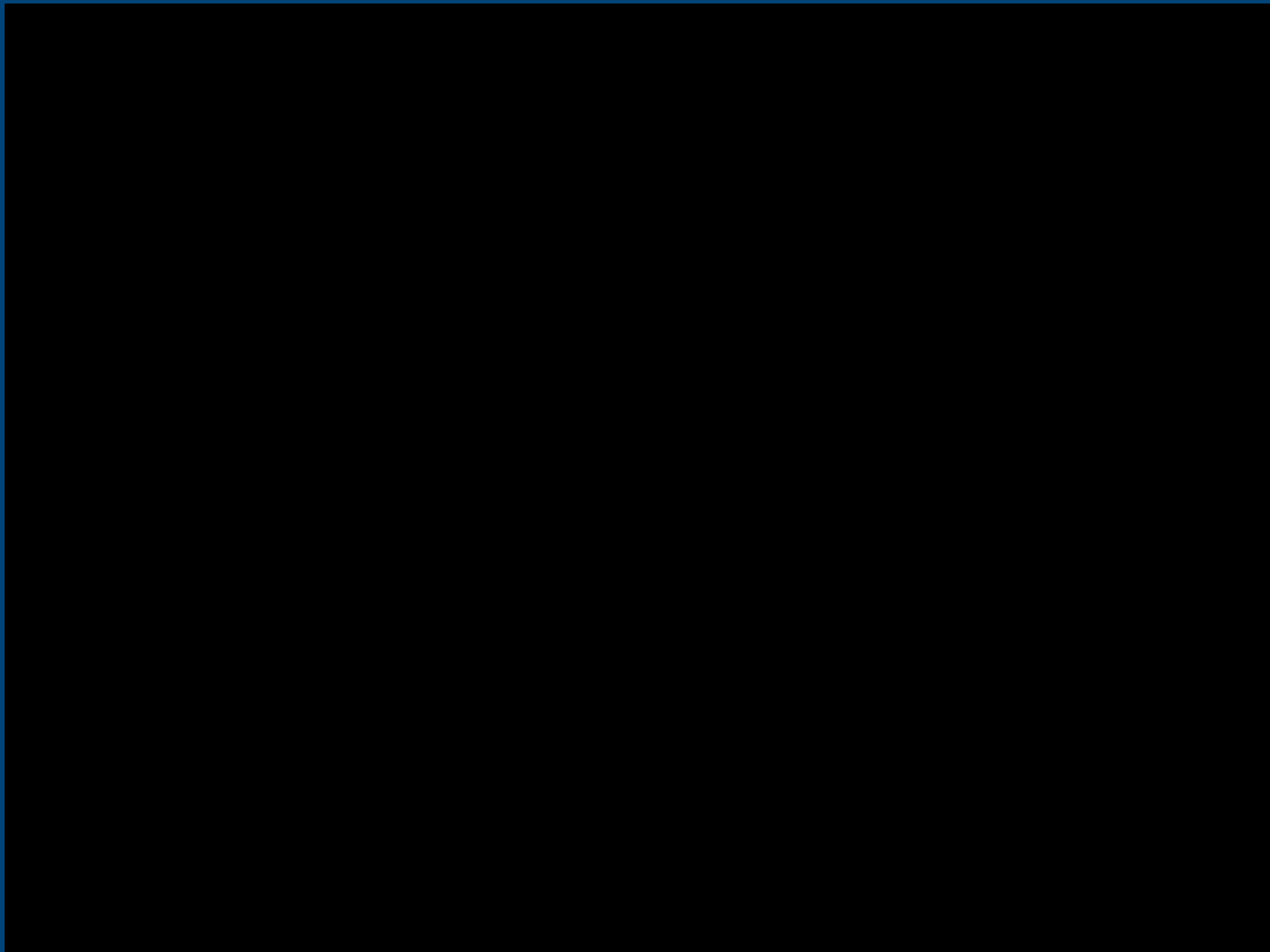
Before



After

Geometry and brightness

Software Calibration



Calibration Related Open Issues

- Color calibration:
  difficult to measure with camera

- Calibration is valid from one point of view (due to screen materials) :
  - Dynamics calibration ?
  - Average calibration from several point of view ???

Classical VR Components

- Output devices:
  - High-end CRT or DLP stereo projectors
  - Active stereo glasses (LCD shutters)

- Input devices:
  - 3D tracker
  - 3D mouse or Wand

- Computer
  - Dedicated graphics supercomputers
  - Multi CPU and multi GPU
  - Classically an SGI Onyx

Input Devices

3D Tracker: returns the 3D position of a point

- Electromagnetic waves, IR, ultrasounds,…
- Very precise when well tuned
- Low latency
- Just a few points
- The user need to be equipped with markers

Ascension System (electromagnetic)

Vicon System (IR cameras)

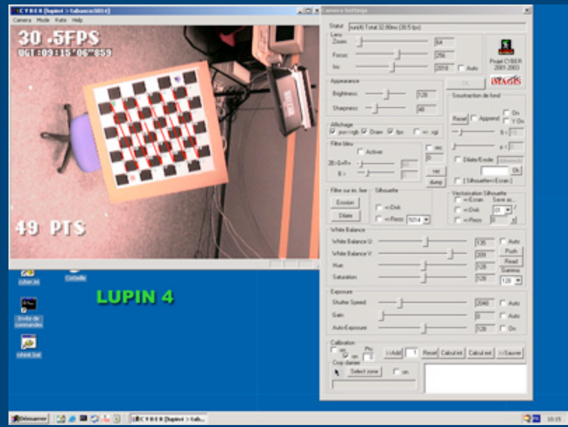
Commodity Components

Idea: Use several commodity cameras shooting the user to retrieve **3D and color data**

- With markers: get only marker related data (position)
- No marker: lot more data available but more difficult

  Visual hull, texture, identification, tracking
- Lot of computations + real time constraints -> parallelization required
- Calibration issues : intra and inter camera parameters

Calibration



- **Compute:**
  - **Internal camera parameters (geometrical distortions, color, ...)**
  - **Inter camera parameters (position in a global coordinate system,...)**

- **Today software enable to compute optical distortions and global positions by shooting a reference object (OpenCV)**

Visual Hull Modeling

Multi Camera Acquisition

↓

Background Subtraction

↓

Parallel 3D Modeling

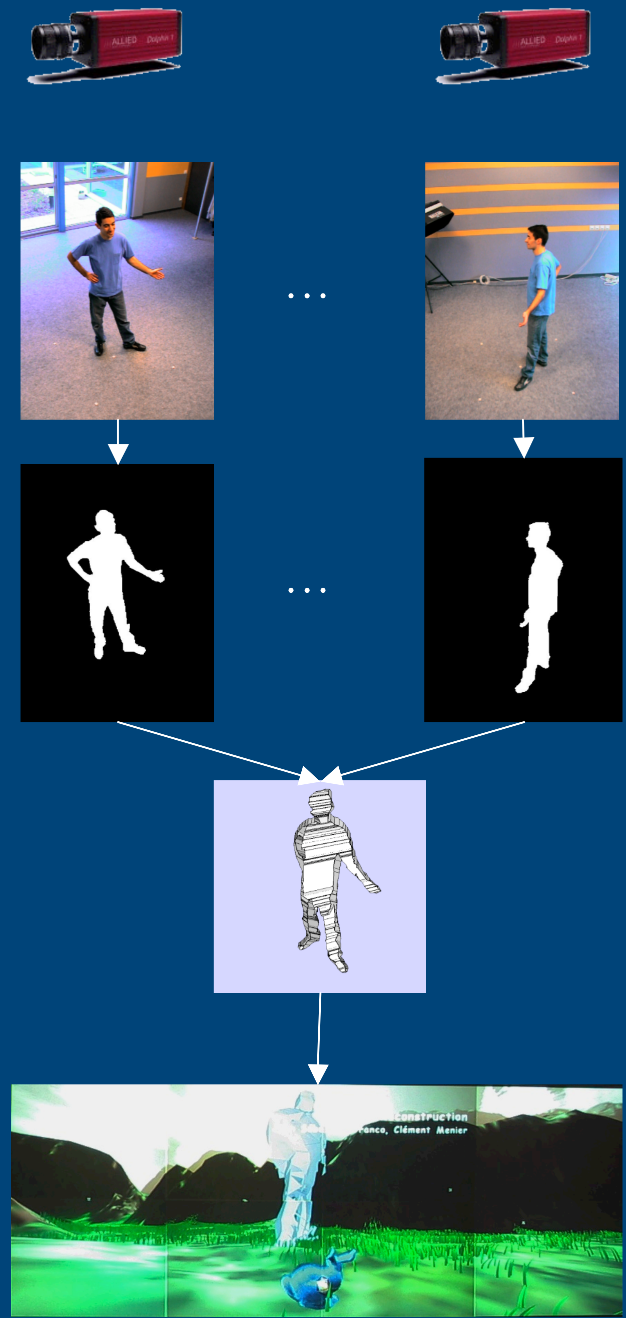
↓

VR Application

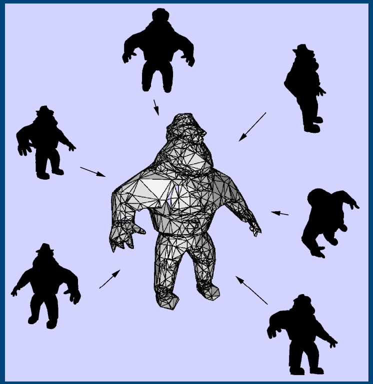
↓

Multi-projector Visualization

PC Cluster



Results 1/3



The quality depends on the number of cameras

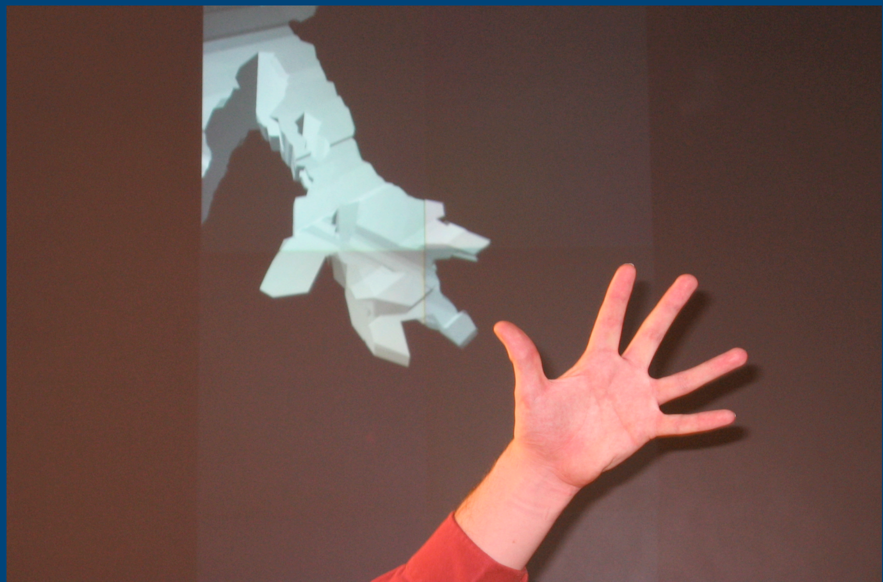
6 cameras

12 cameras

25 cameras

Results 2/3

- High precision: 0.5cm

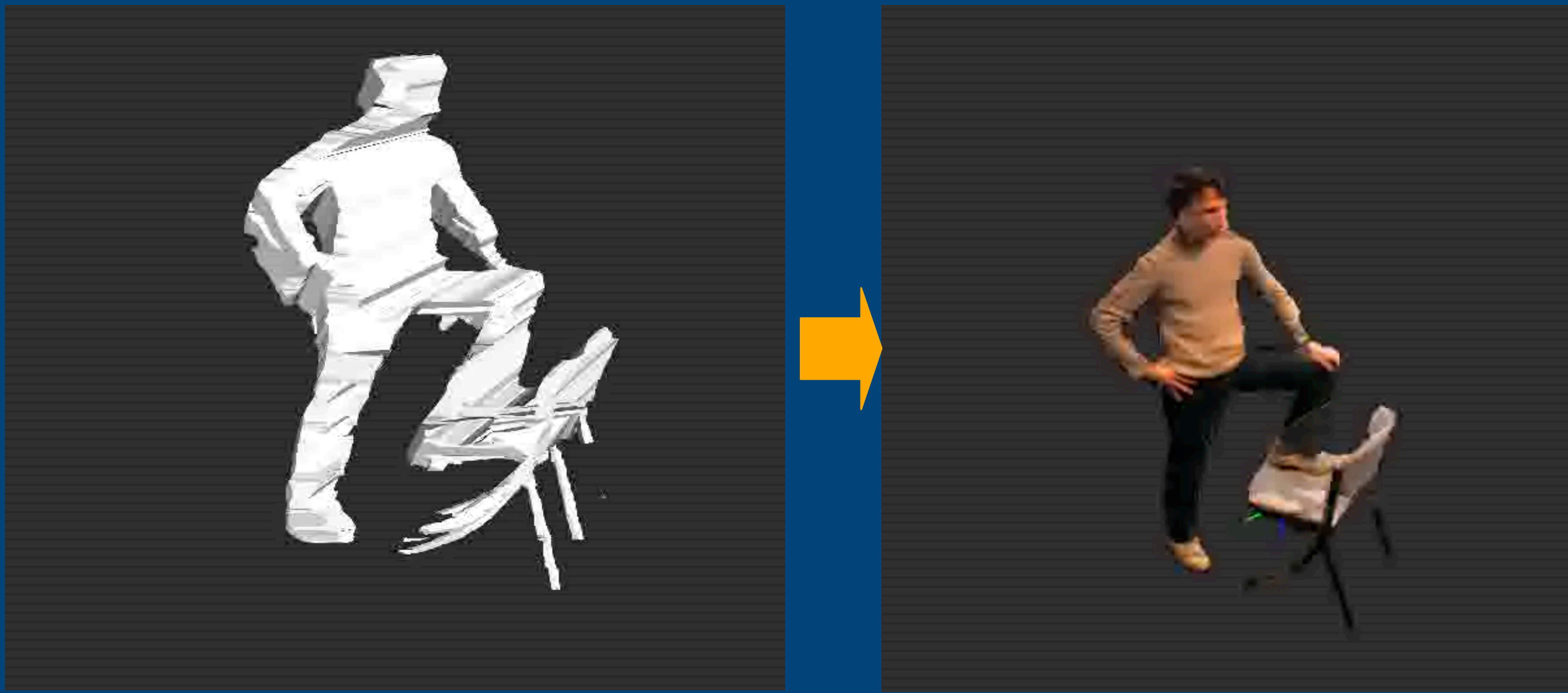


- Real time: 20-30 fps, 100-150 ms latency, with 4 cameras and 11 dual Xeon, gigabit network

Results 3/3

Texturing the Model

Use camera images to add color information



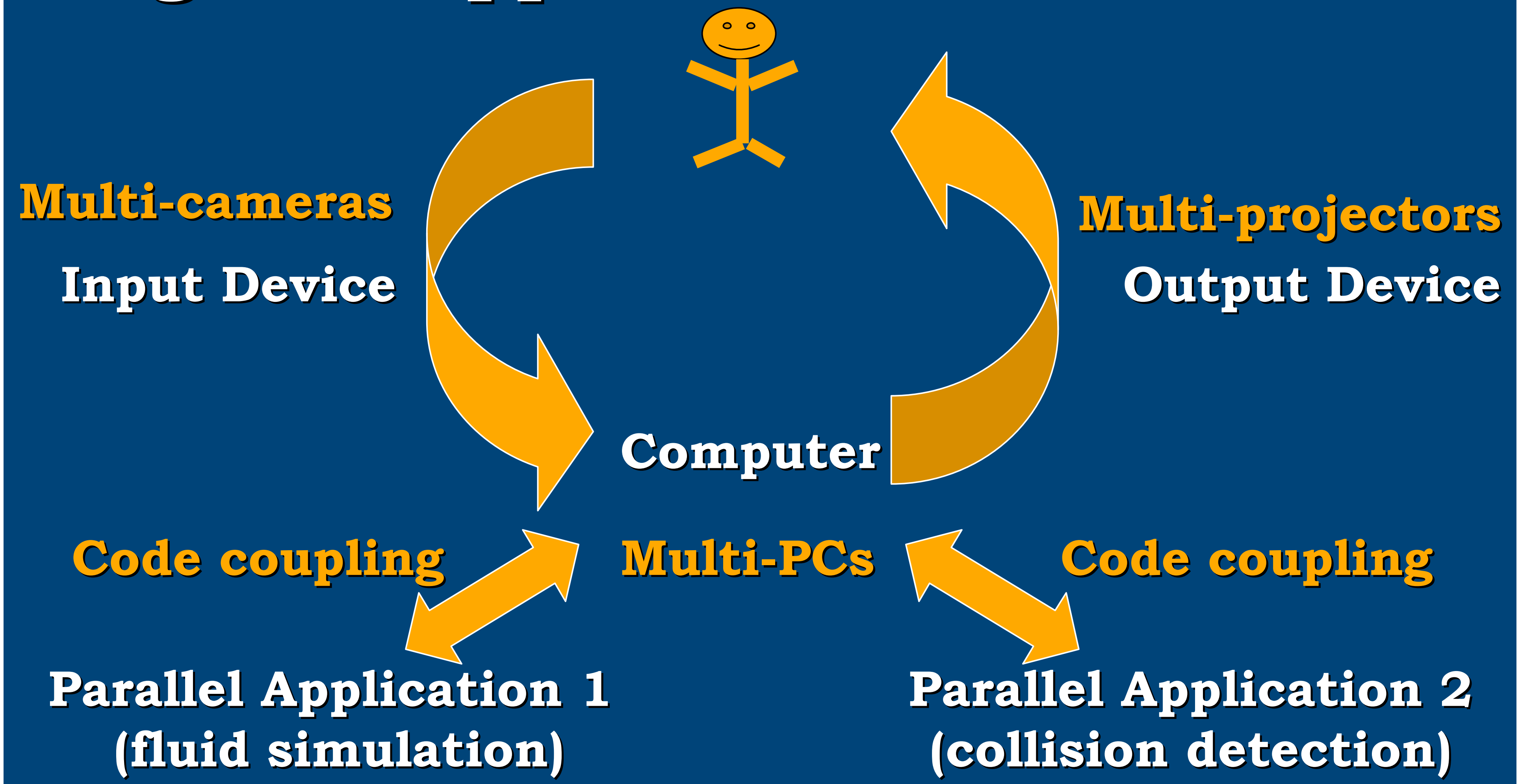
What's Next ?

- Real time texturing
- Identification
- Tracking

A lot of difficult issues

(artificial vision)

Large VR Applications



A complex distributed application: difficult to program and execute

FlowVR: Middleware for Large VR Applications

Existing tools like Corba, MPI, PadicoTM are not well adapted for large VR applications

FlowVR goals:

- Parallel Code coupling
- “Component” programming
  - Ease code composition, multi-device support
- Advanced collective communications

The FlowVR Model

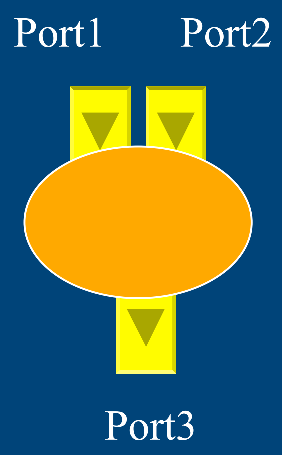
- An application: **Modules + Network**
- **Module:** a computation loop (usually a process)
- **Network:** connect modules and define how messages are processed

Module and network programming are separated.

The FlowVR Modules

- An iterative process
- Several Input and output ports
- Simple API to ease porting existing code:

```
Init()
While (! Stop)
a= Port1.Get()
b= Port2.Get ()
E= Compute(a,b)
Port3.Put(e)
End loop
```

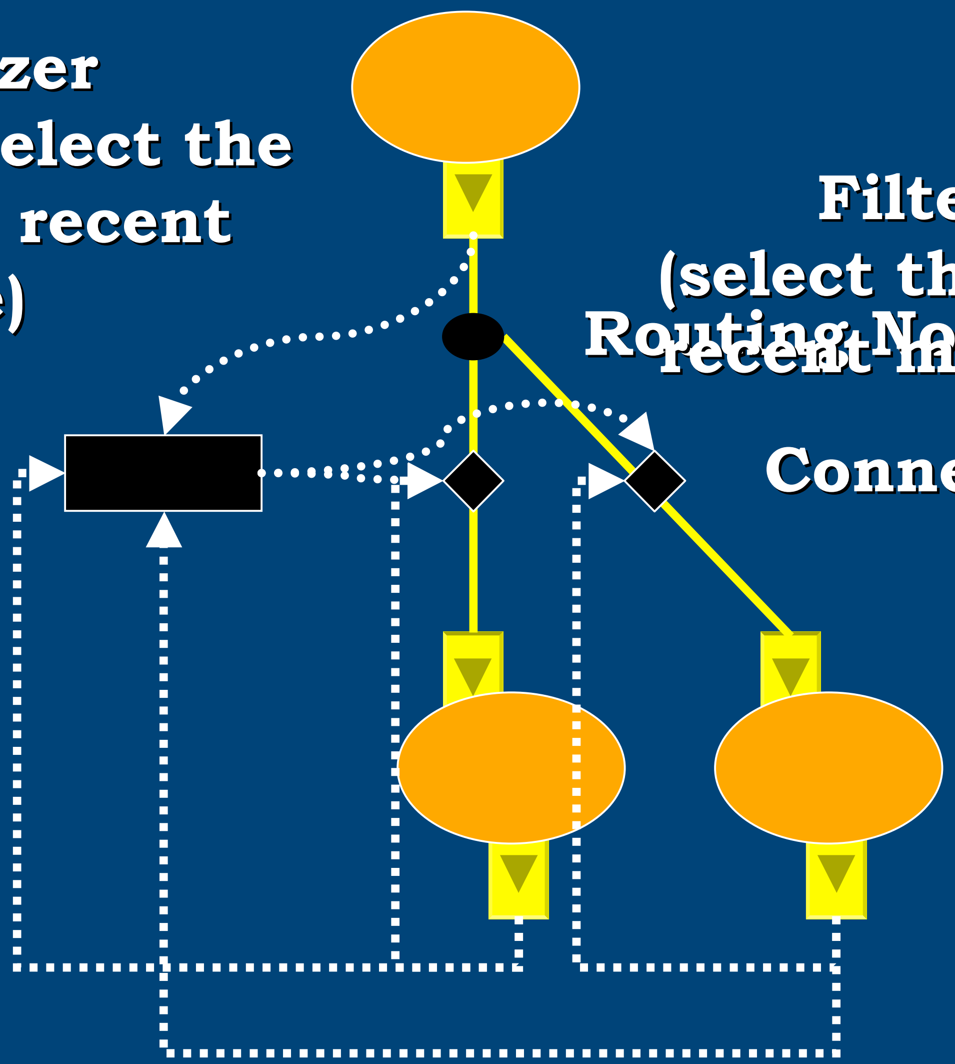


- An XML description used when assembling modules

The FlowVR Network

- The network assembles modules using:
  - Connections (FIFO channels)
  - Routing nodes
  - Filters (process messages)
  - Synchronizers (implement synchronization policies)

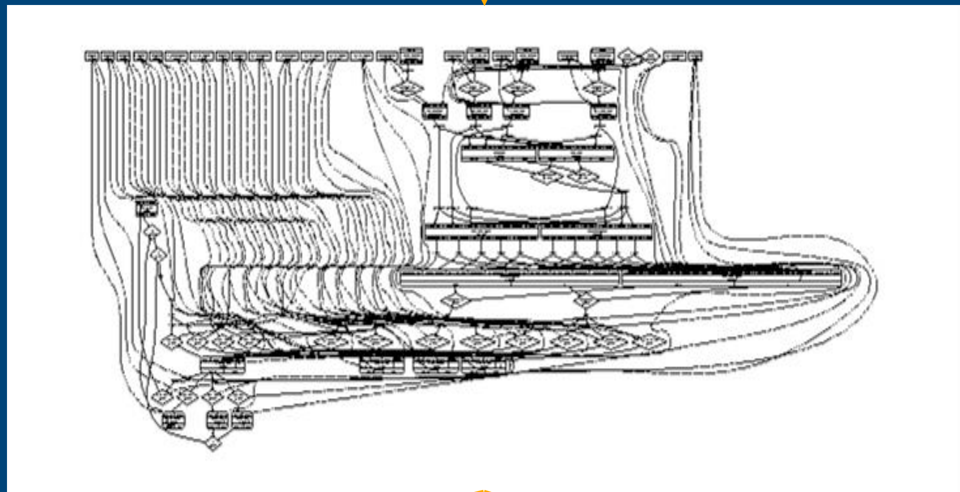
The FlowVR Network



The FlowVR Environment

XML files + Parameters

Script (Perl)



Launch modules

Launch the network

Application running !!

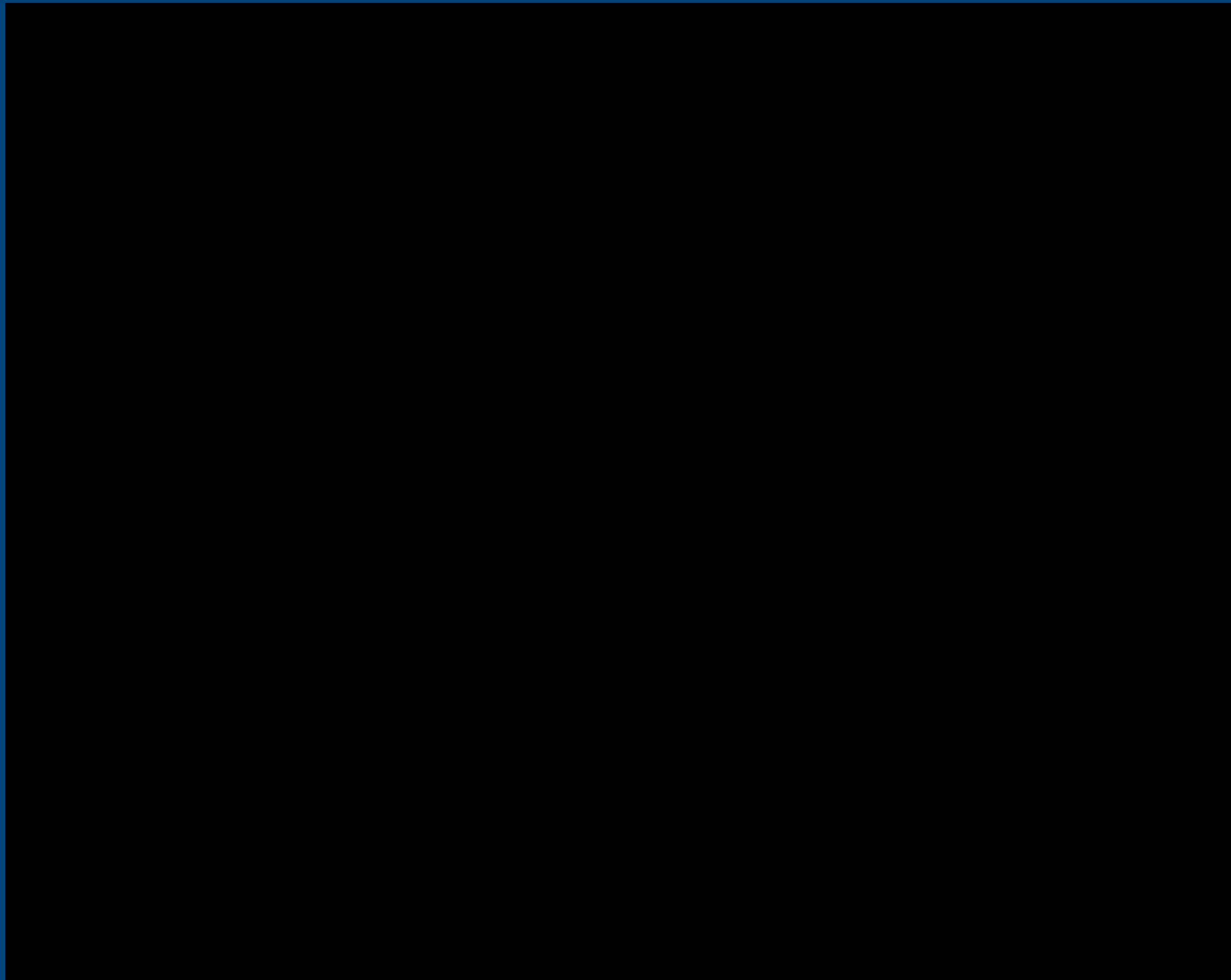
Results

- Performance matches existing parallel programming tools (MPI)
- Very positive feedback from users:
  - Programming is really simpler
  - Plug & Play modules
  - Change network without recompilation

Well adapted to large VR applications

http://flowvr.sf.net

FlowVR



Conclusion

- Aggregating multiple commodity components: YES !!
  - Low cost, modular, scalable, standard
  - Good softwares are essentials:
    - to compensate for missing features
    - to assist users for programming and tuning the multiple components.

Still room for new ideas