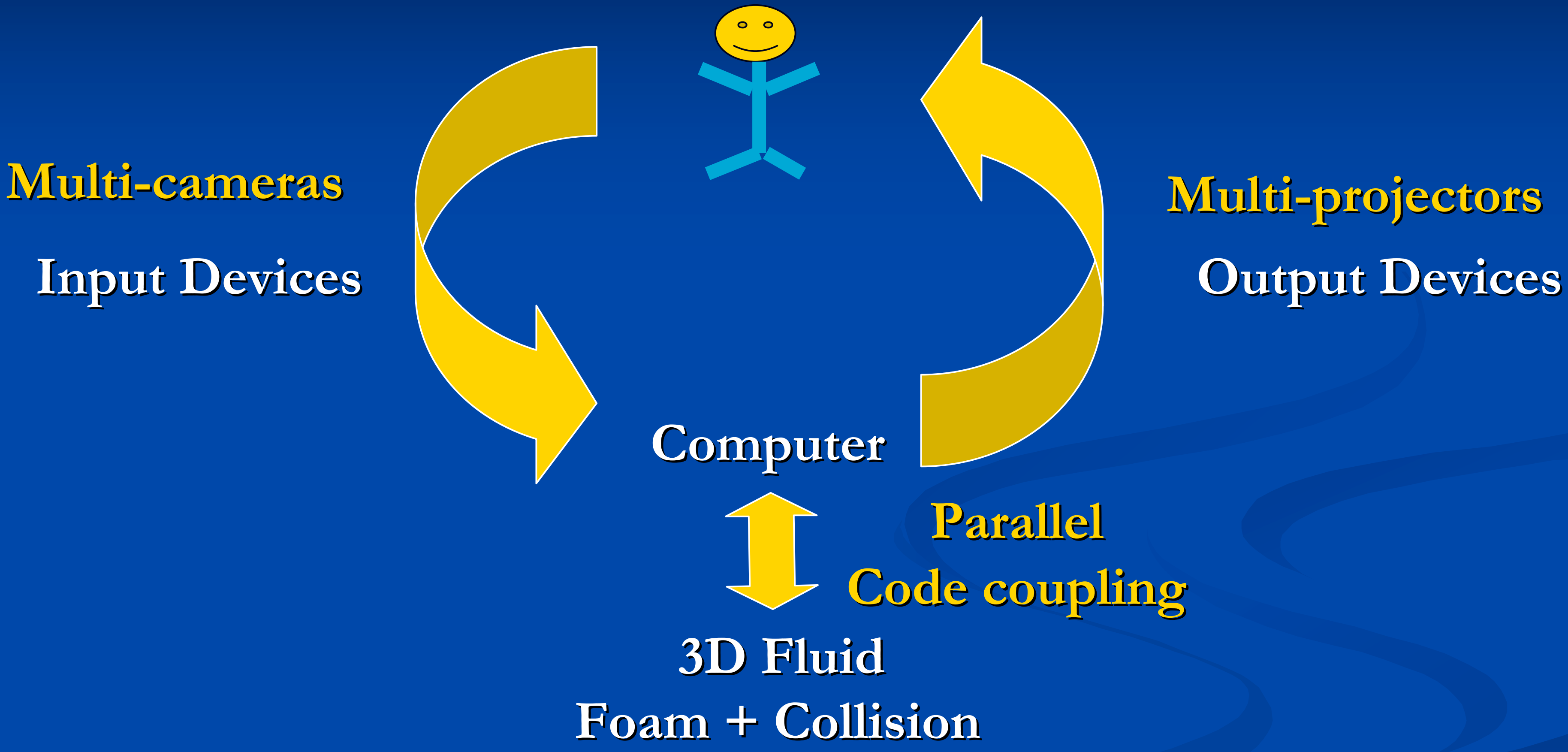
Running Large VR Applications on a PC Cluster: the FlowVR Experience

Clément Menier, Jérémie Allard,
Edmond Boyer, Bruno Raffin
INRIA Rhône-Alpes, France

Large VR Applications



A complex distributed application with real time constraints

Large VR Applications

- Critical Problems:
  - Code coupling, code re-use
  - Aggregatation of multiple ressources (performance vs complexity)

Use a middleware to alleviate both problems

Middleware Solution

- Requirements:
  - Modularity (hundreds of components)
  - Interactivity (10-1000Hz)
  - Parallel code coupling (efficient communications)

	VR	Parallelism	C. B.	**FlowVR**
Modularity	✓	✗	✓	✓
Interactivity	✓	✗	✗	✓
PCC	✗	✓	✗	✓

FlowVR

- Modularity: component based
  - Modules: minimal modification of available programs.
  - Clear separation between the modules and the application network.
- Performance:
  - Zero-copy shared memory.
  - Distribution schemes inspired from parallelism, adapted to VR
- Interactivity: low latency and high frame rate

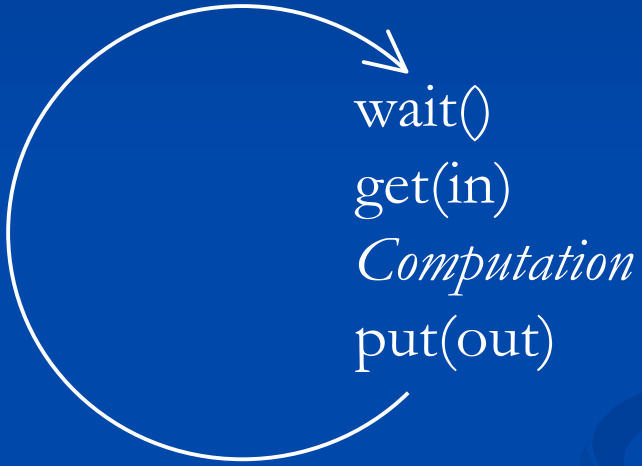
FlowVR

An application = Modules + Network

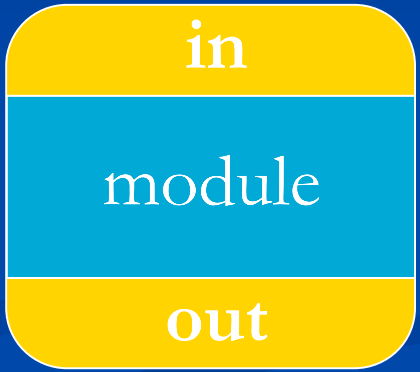
- Modules : a computation loop
  - Executed outside FlowVR (own process)
  - Not aware of the existence of other modules
- Network :
  - A dataflow graph
  - Connect modules and define how messages are processed

FlowVR Modules

- Module API:

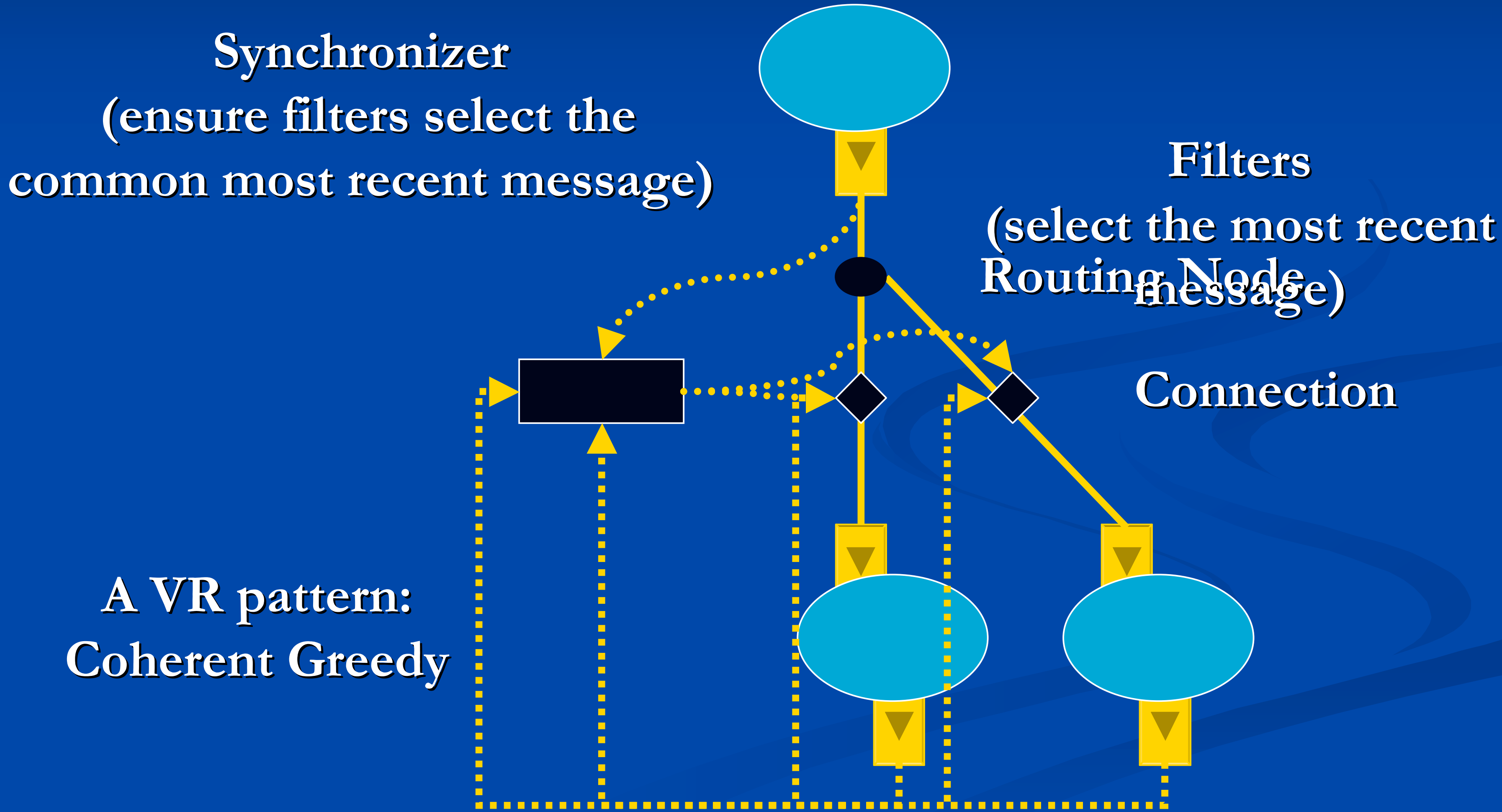


```
wait()
get(in)
Computation
put(out)
```



- Messages
  - Buffer: payload
  - Stamps: light-weight data (time stamp, bounding box)

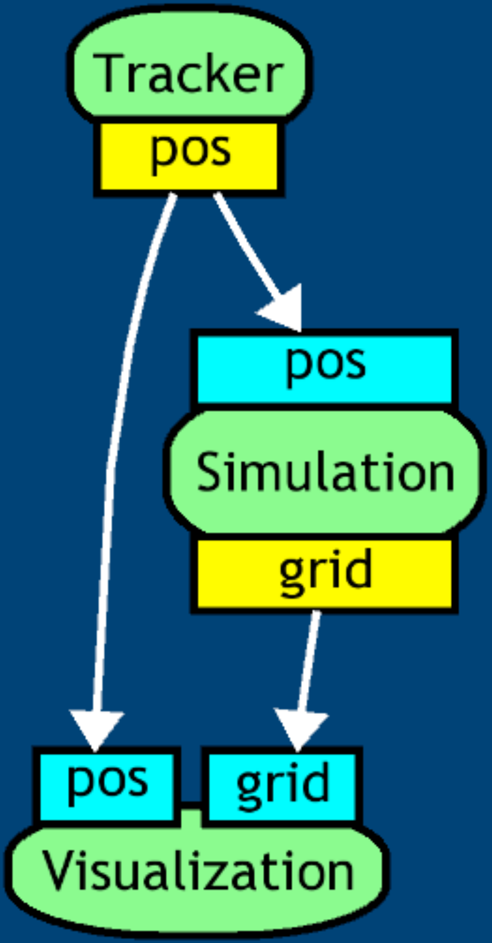
FlowVR Network


Synchronizer
(ensure filters select the common most recent message)
Filters
(select the most recent message)
Routing Node
Connection

A VR pattern:
Coherent Greedy

Connections

- Connect each Input to one Output
- FIFO Communications

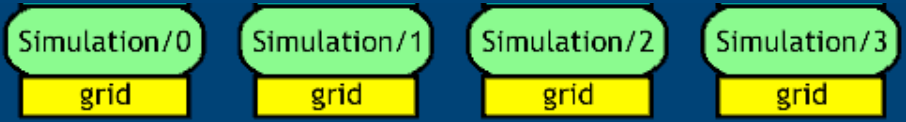


http://flowvr.sf.net/

Parallel Code Coupling

Simulation:

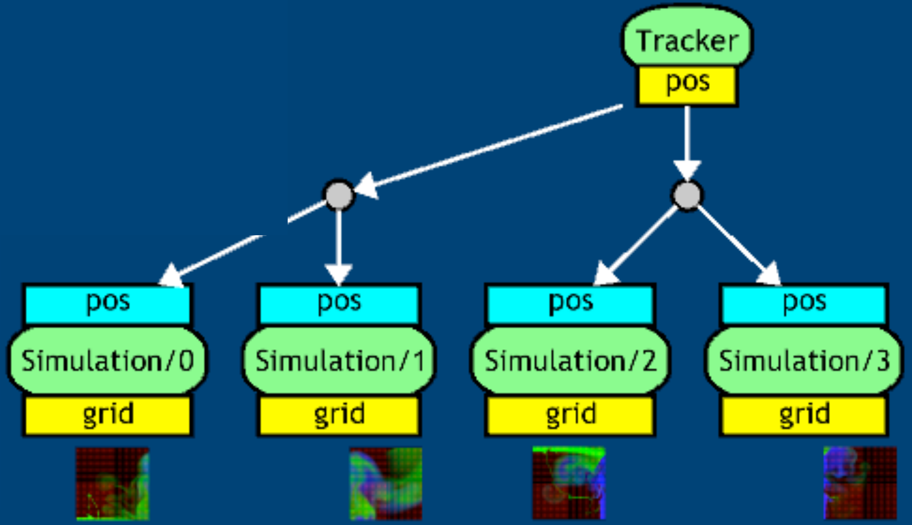
- **Parallel Navier-Stokes solver based on a 2D**
- **Programmed with MPI (or other)**



- **Each process**
  - **has one piece of the mesh**
  - **at each iteration**
    - **exchange values on the mesh borders**
    - **compute a new state**

Parallel Code Coupling

Native (MPI) communications are transparent for FlowVR



- FlowVR point of view:
  - 1 module per process
  - Inputs must be broadcasted or scattered
    - Broadcast tree specified using *Routing Nodes*
  - Each module may output only a part of the data
    - Gather may be required

http://flowvr.sf.net/

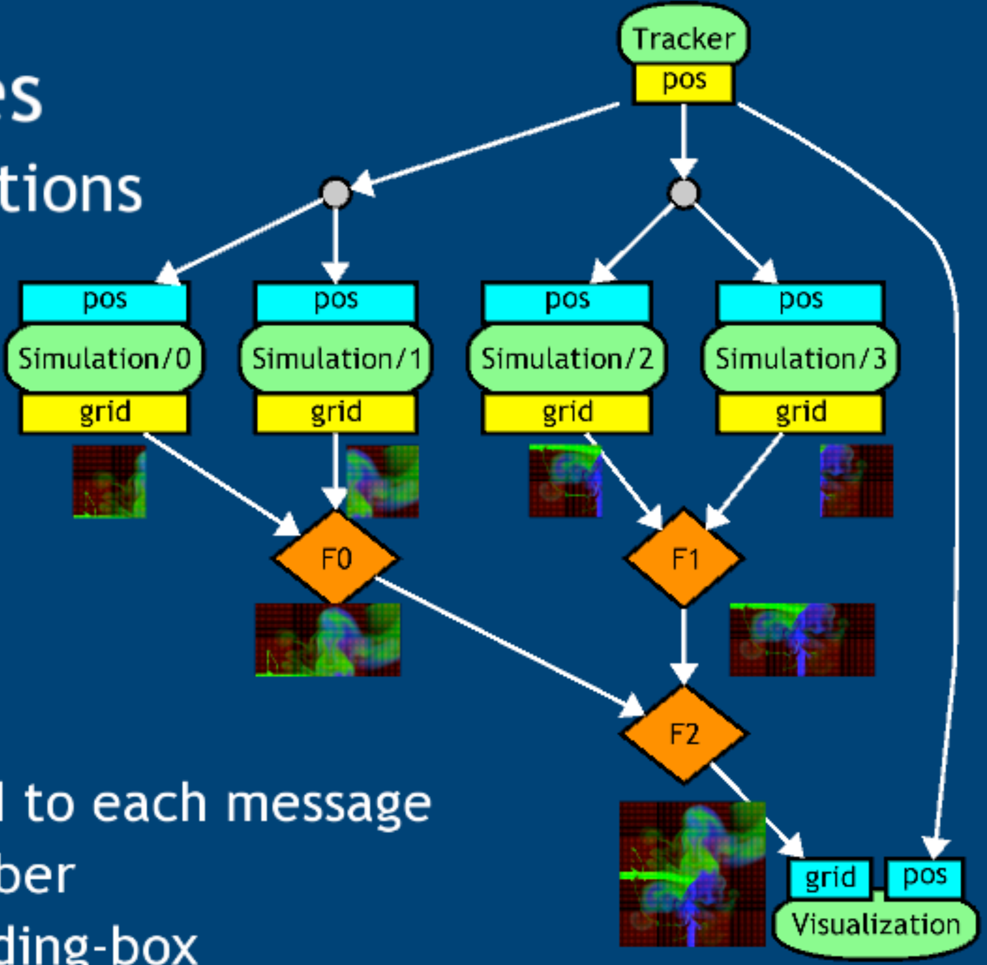
Filters

- Process messages

  Collective communications
  - Scatter
  - Gather
  - Filtering
  - Conversion
  - Compression

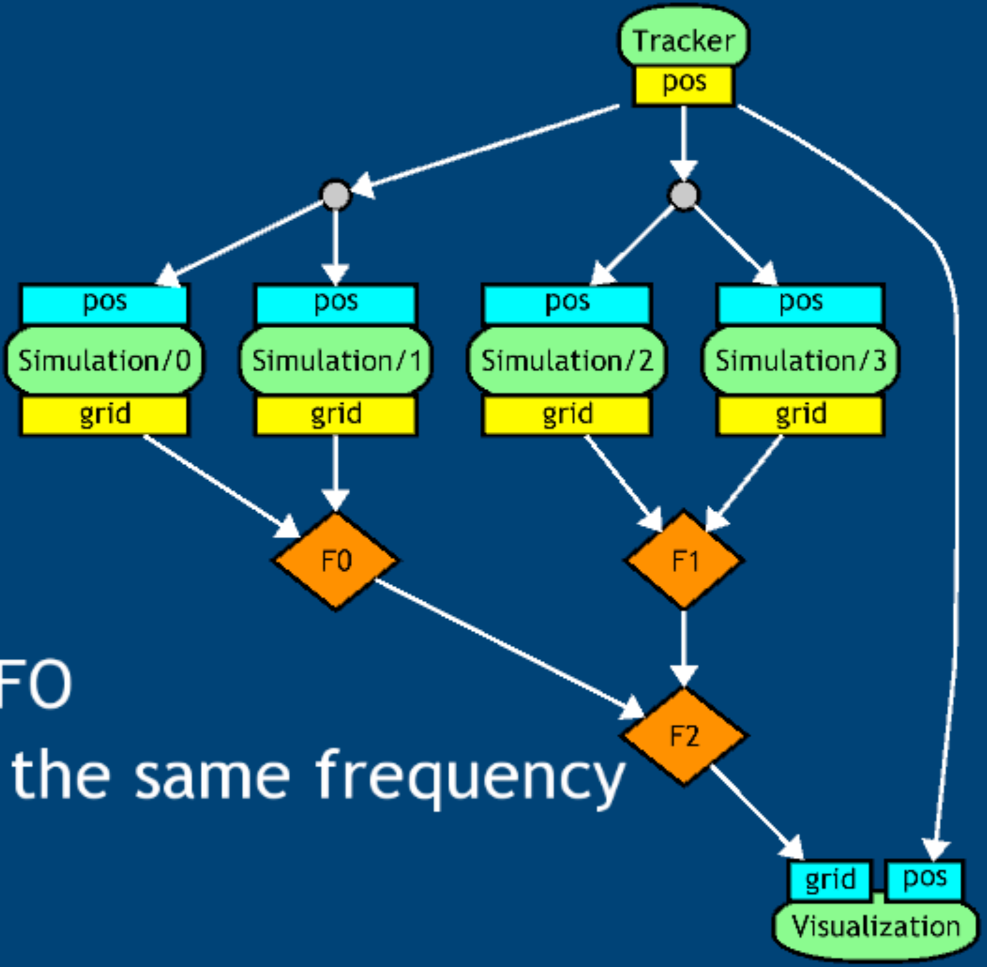
- Use *Stamps*

  Semantic data associated to each message
  - source, message number
  - coordinates, 3D bounding-box
  - user-defined



http://flowvr.sf.net/

Filters

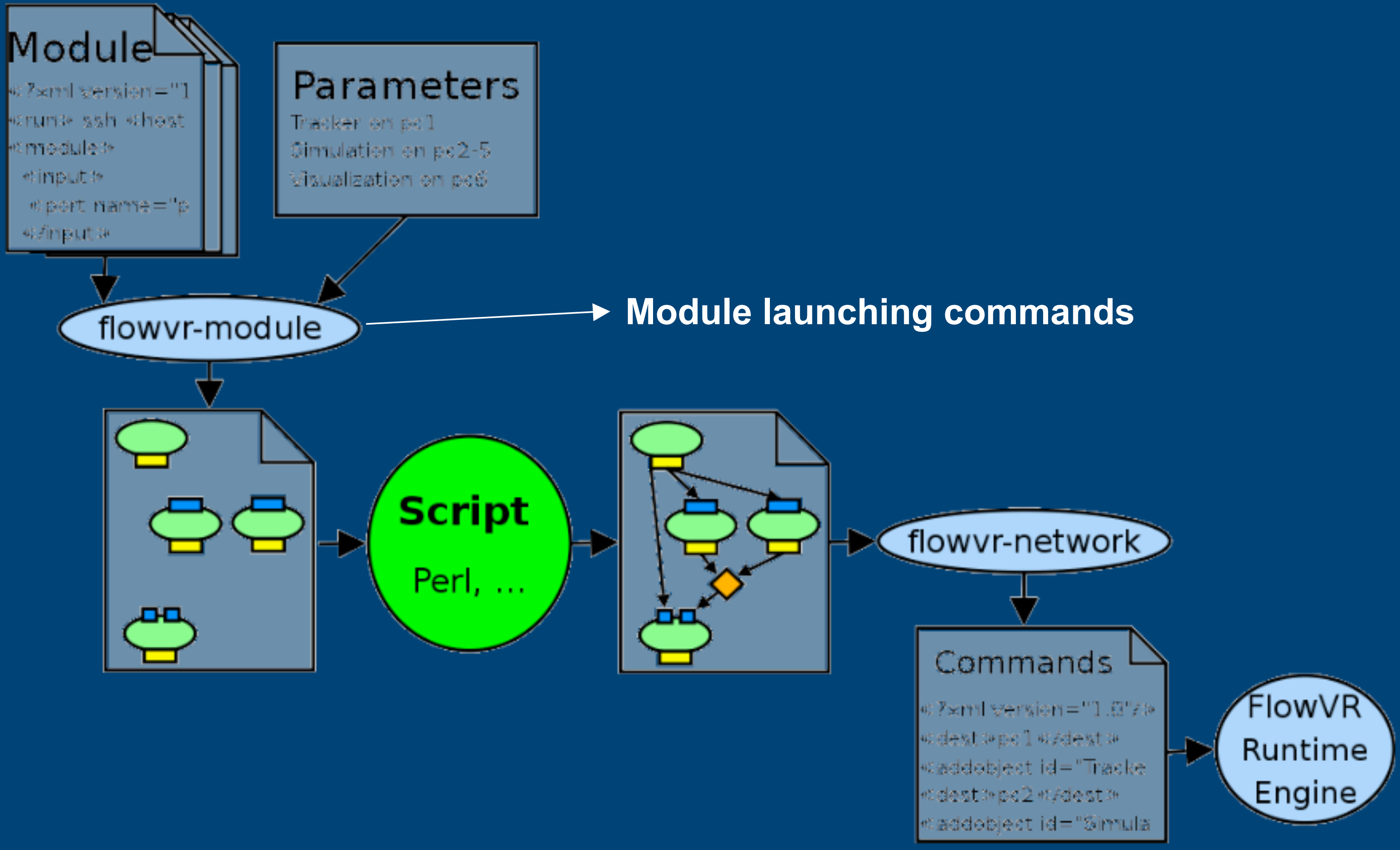


All connections are FIFO

➔ All modules run at the same frequency

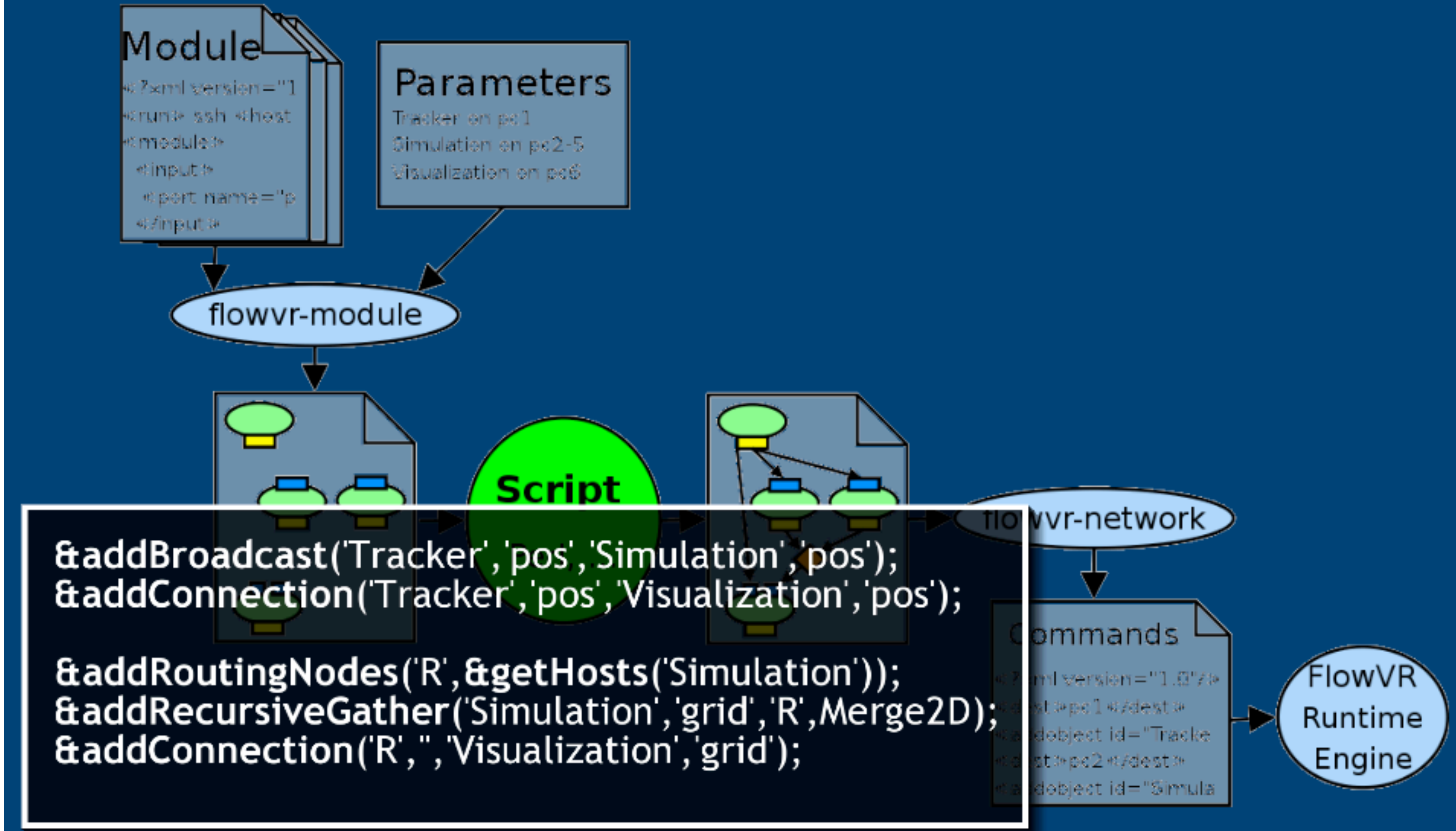
http://flowvr.sf.net/

Development Environment



http://flowvr.sf.net/

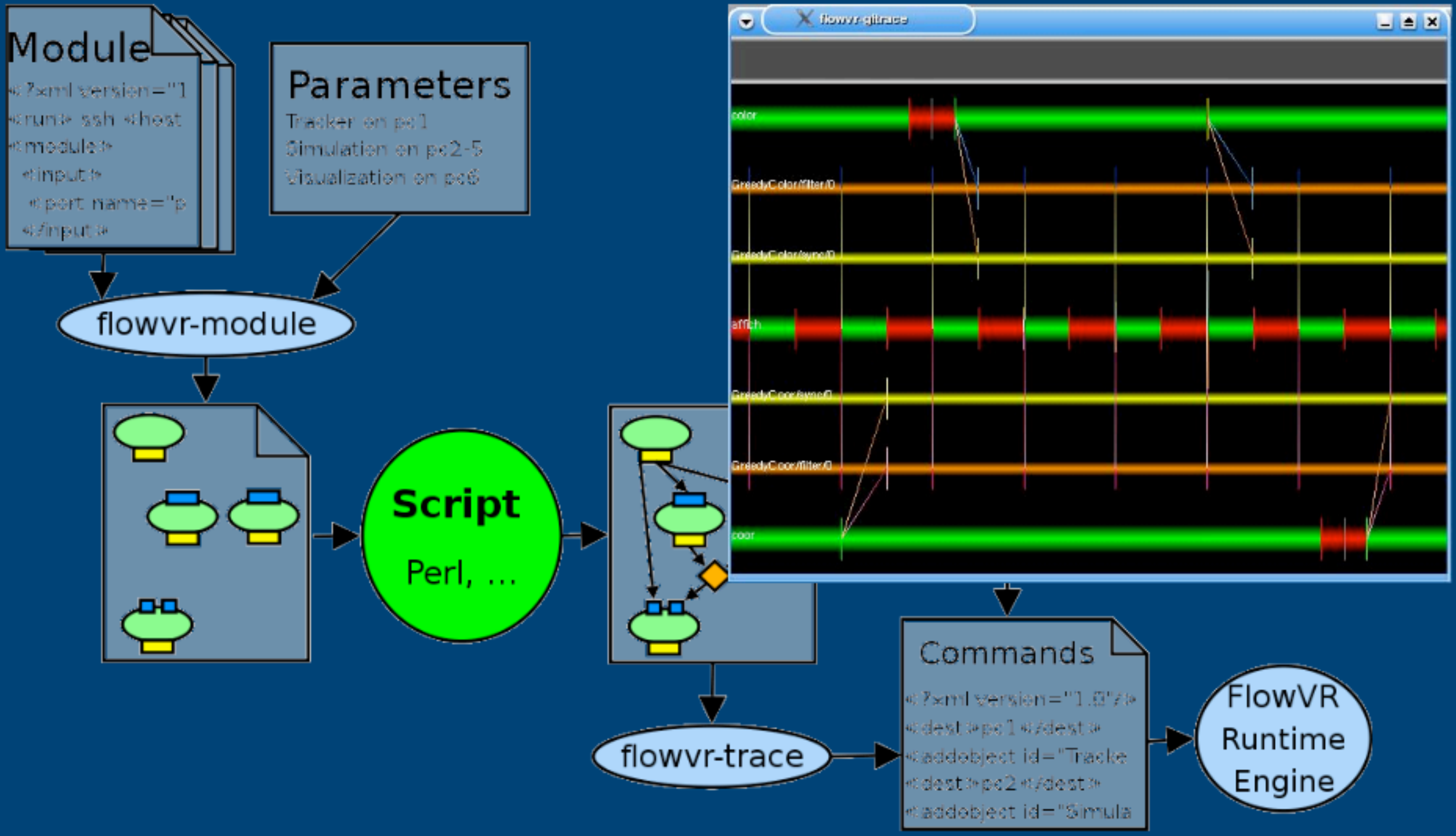
Development Environment



```
&addBroadcast('Tracker','pos','Simulation','pos');
&addConnection('Tracker','pos','Visualization','pos');

&addRoutingNodes('R',&getHosts('Simulation'));
&addRecursiveGather('Simulation','grid','R',Merge2D);
&addConnection('R','','Visualization','grid');
```

http://flowvr.sf.net/

Development Environment

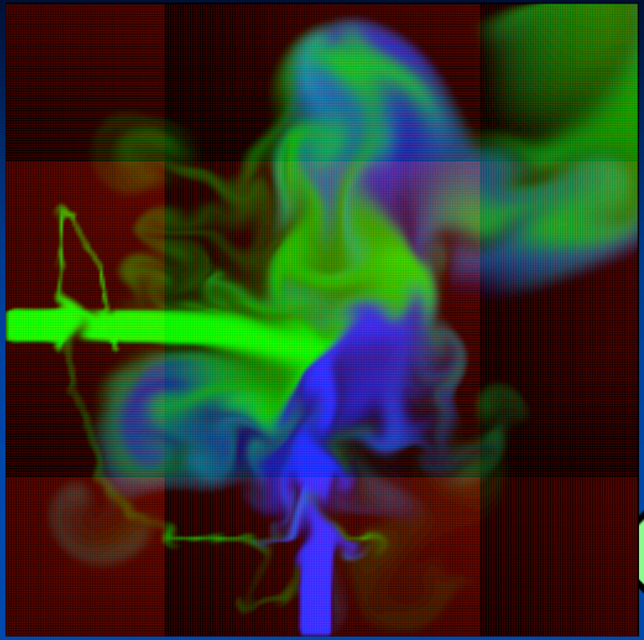


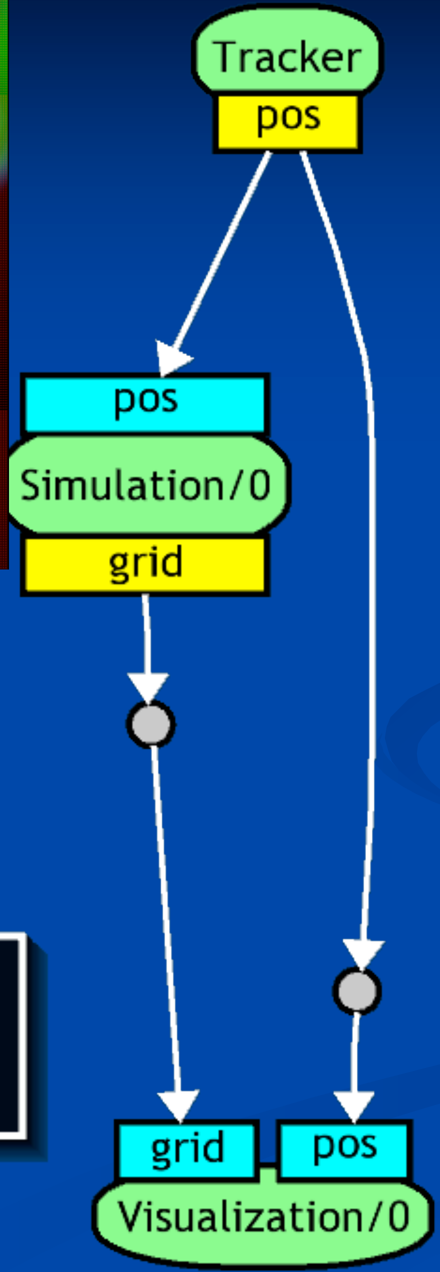
```
<?xml version="1
<run> ssh <host
<module>
  <input>
   <port name="p
  </input>
```

```
Tracker on pc1
Simulation on pc2-5
Visualization on pc6
```

```
<?xml version="1.0"/>
<dest>pc1</dest>
<addobject id="Tracke
<dest>pc2</dest>
<addobject id="Simula
```

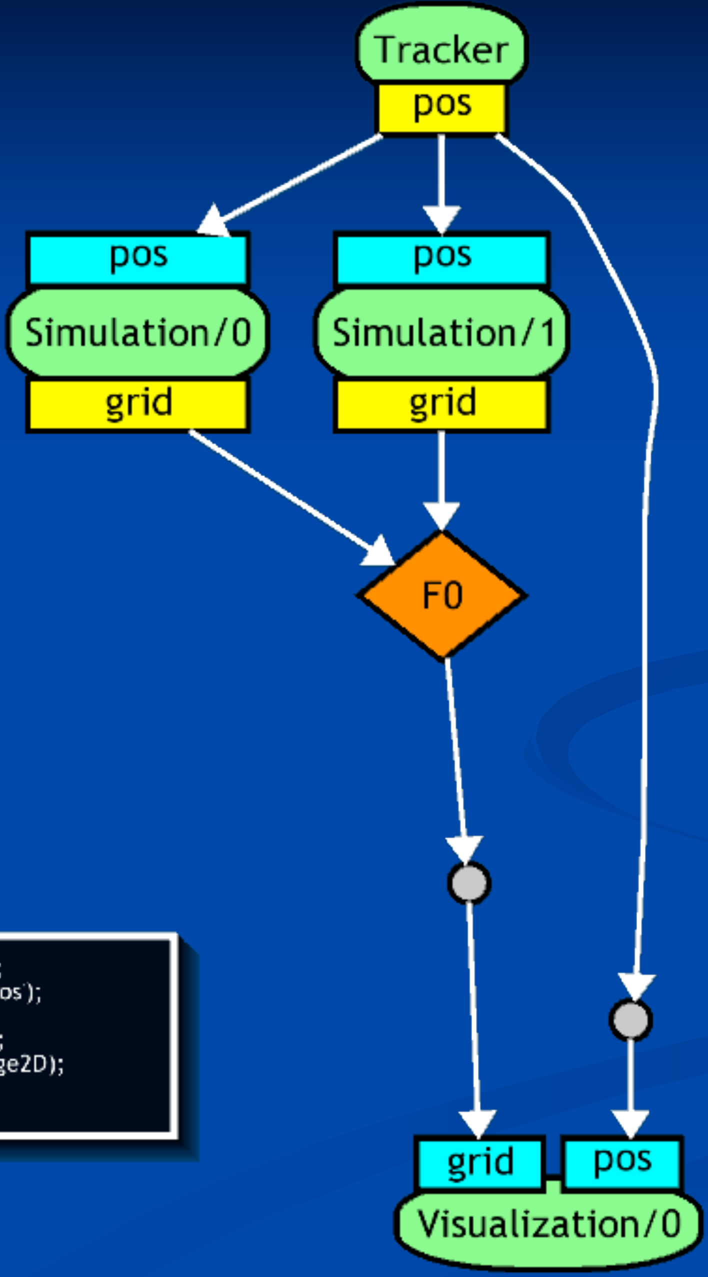
http://flowvr.sf.net/





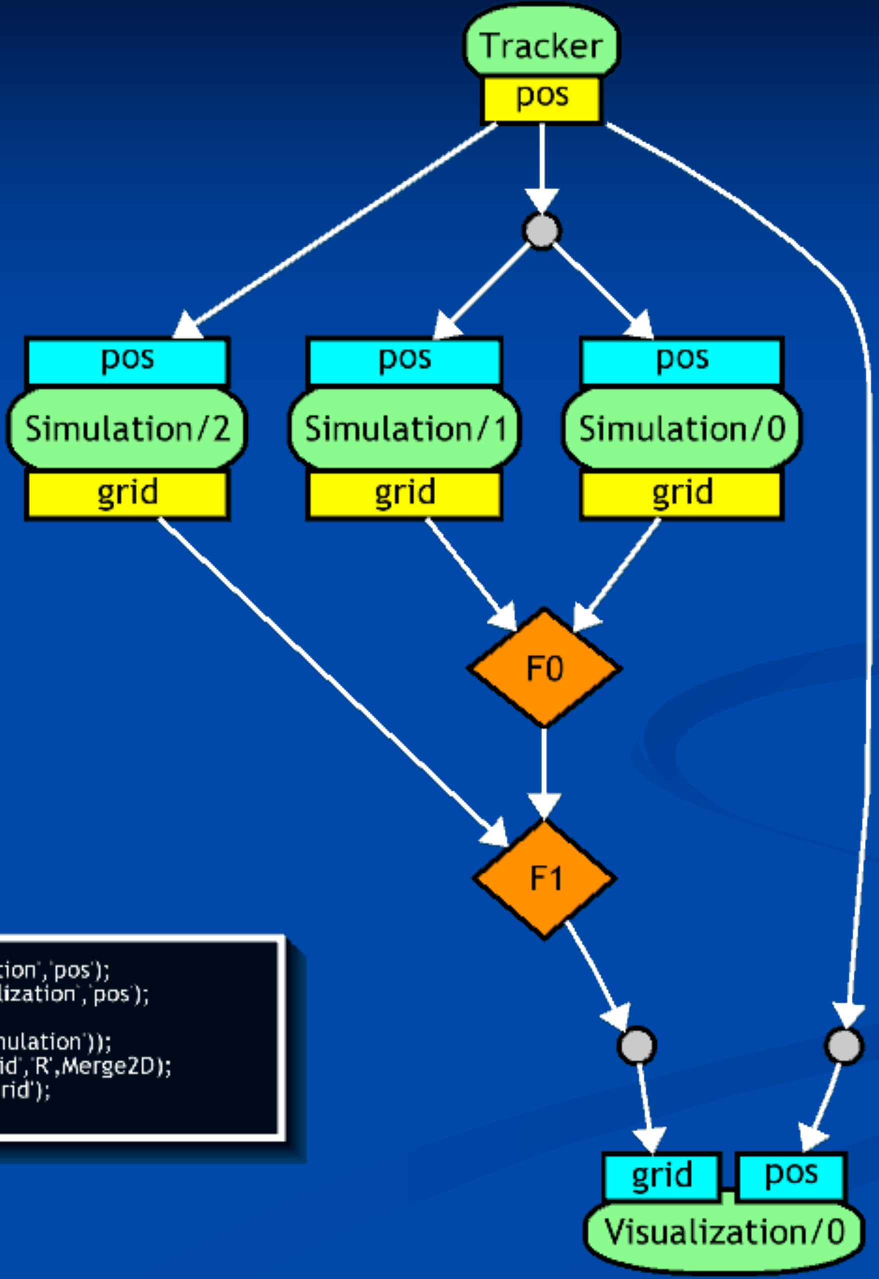
Tracker: 1
Simulation: 1
Visualization: 1

```
&addBroadcast('Tracker','pos','Simulation','pos');
&addConnection('Tracker','pos','Visualization','pos');

&addRoutingNodes('R',&getHosts('Simulation'));
&addRecursiveGather('Simulation','grid','R',Merge2D);
&addConnection('R','','Visualization','grid');
```



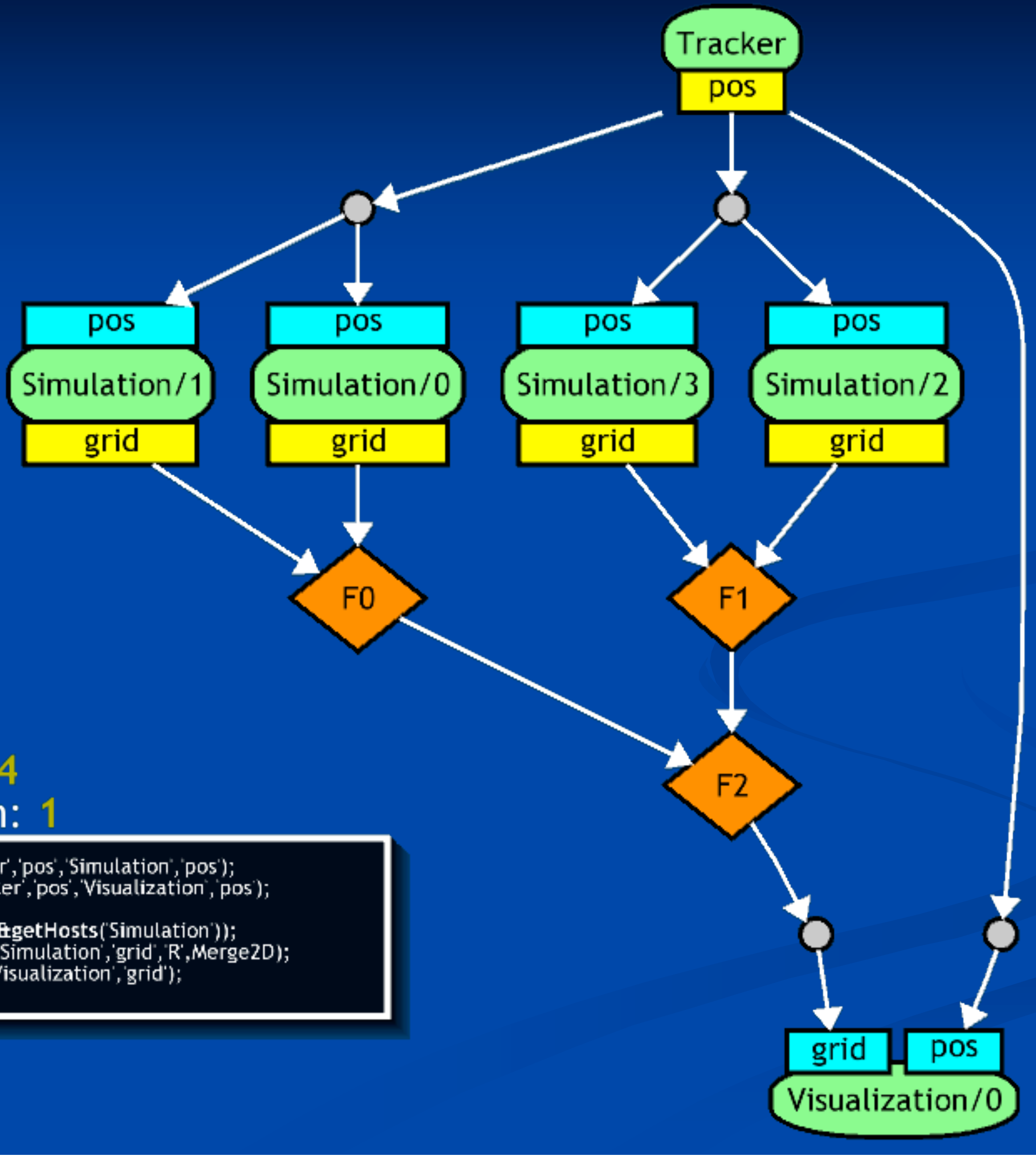
Tracker: 1
Simulation: 2
Visualization: 1

```
&addBroadcast('Tracker','pos','Simulation','pos');
&addConnection('Tracker','pos','Visualization','pos');

&addRoutingNodes('R',&getHosts('Simulation'));
&addRecursiveGather('Simulation','grid','R',Merge2D);
&addConnection('R','','Visualization','grid');
```



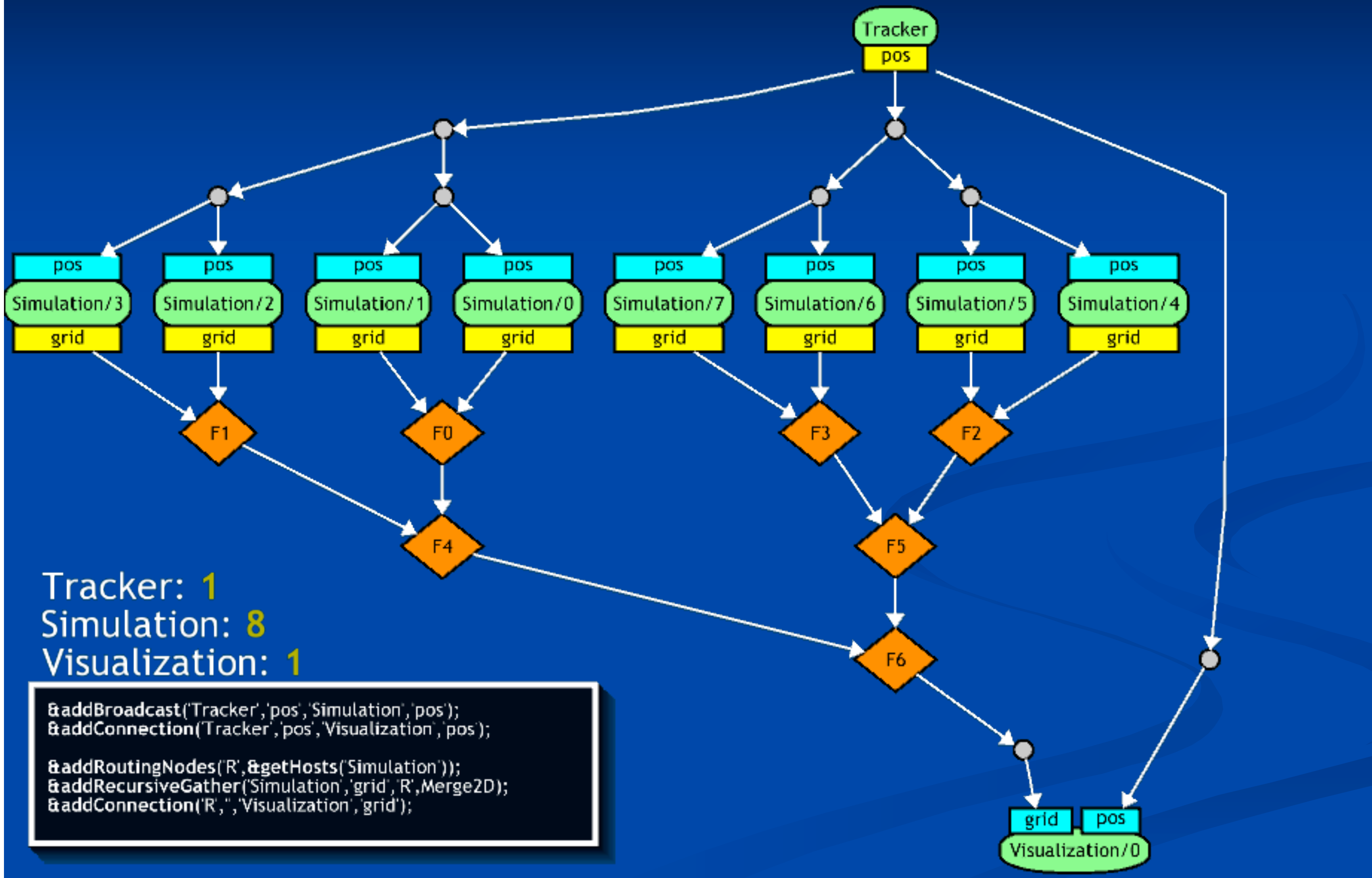
Tracker: 1
Simulation: 3
Visualization: 1

```
&addBroadcast('Tracker','pos','Simulation','pos');
&addConnection('Tracker','pos','Visualization','pos');

&addRoutingNodes('R',&getHosts('Simulation'));
&addRecursiveGather('Simulation','grid','R',Merge2D);
&addConnection('R','','Visualization','grid');
```



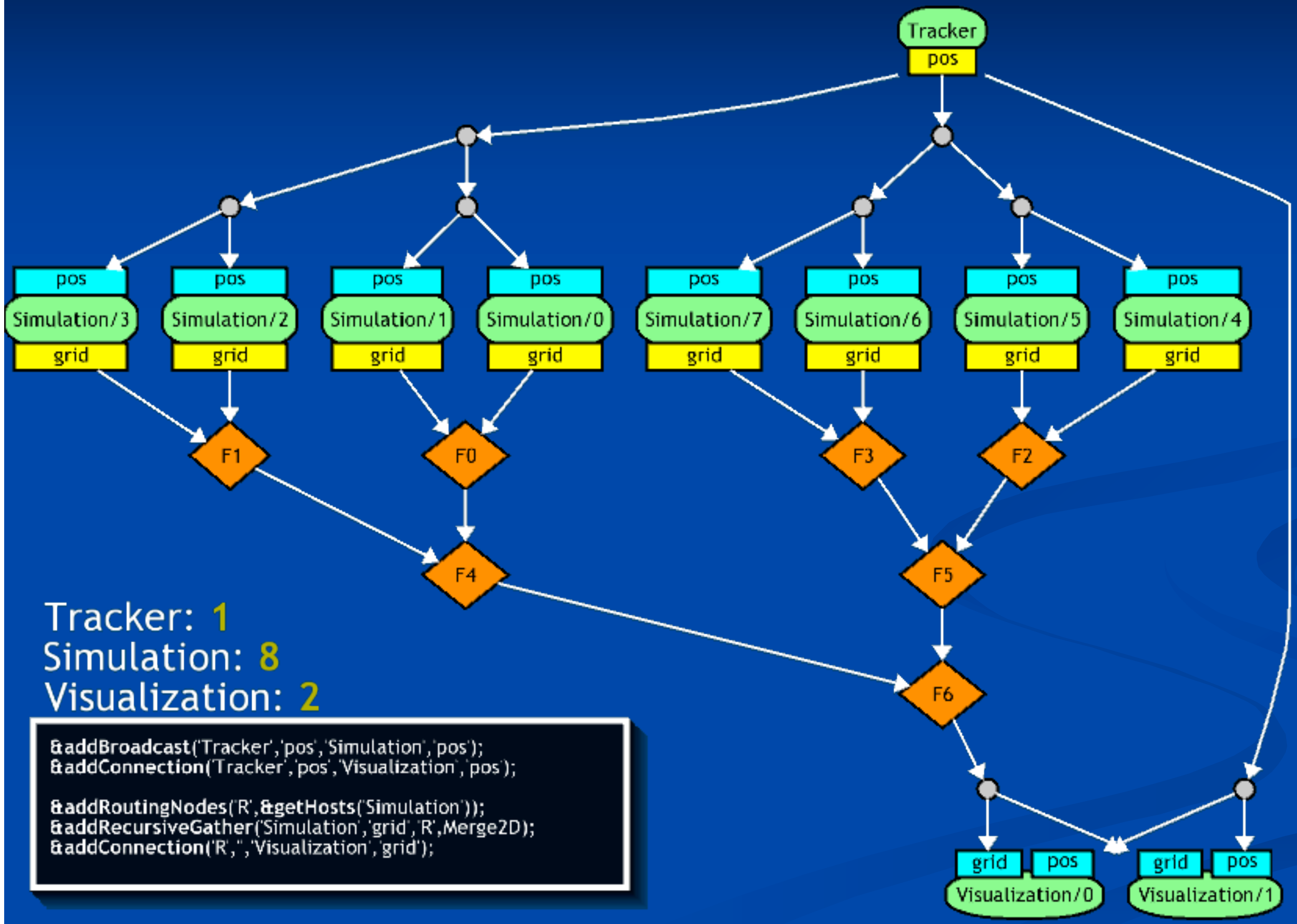
Tracker: 1
Simulation: 4
Visualization: 1

```
&addBroadcast('Tracker','pos','Simulation','pos');
&addConnection('Tracker','pos','Visualization','pos');

&addRoutingNodes('R',&getHosts('Simulation'));
&addRecursiveGather('Simulation','grid','R',Merge2D);
&addConnection('R','','Visualization','grid');
```



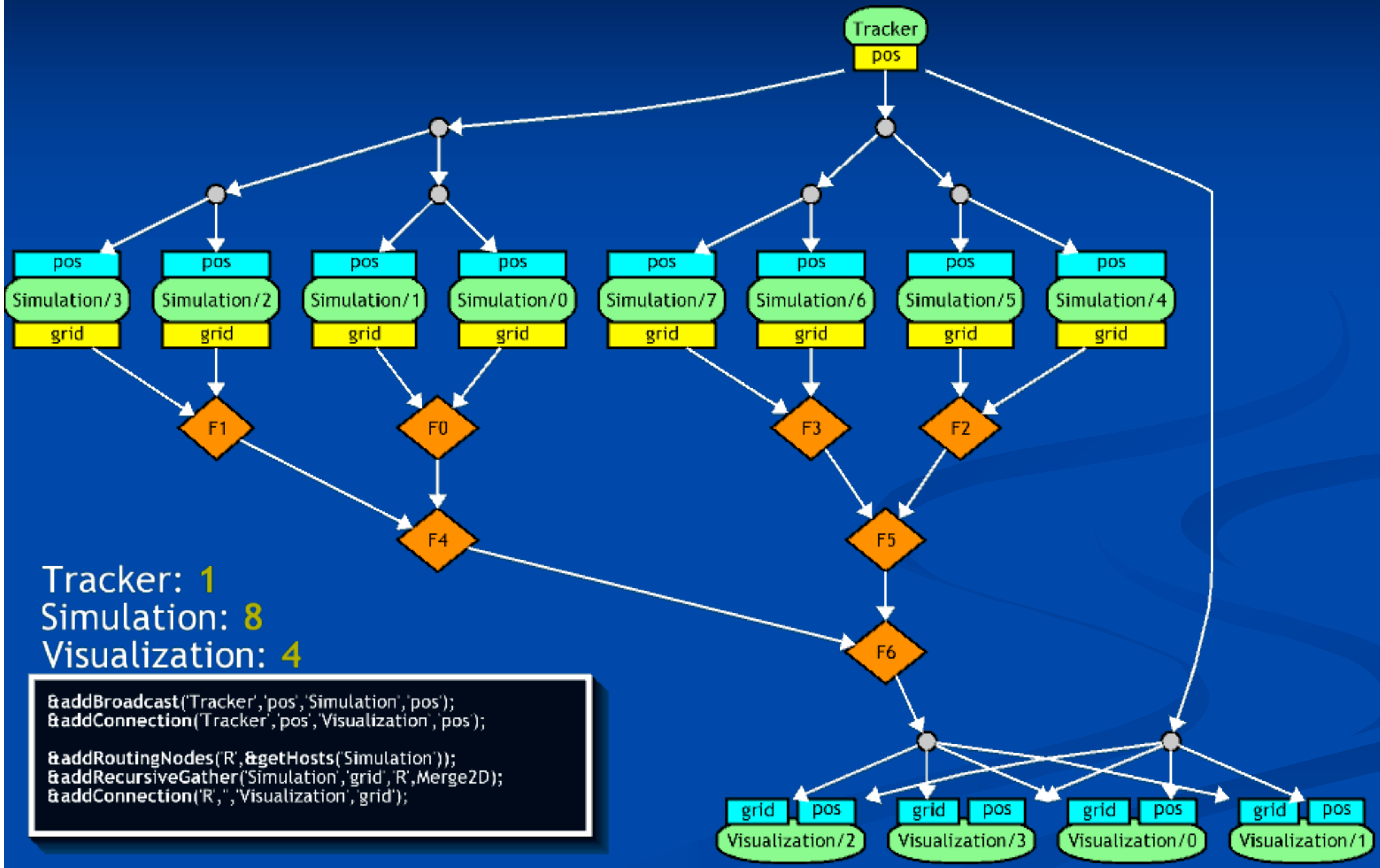
Tracker: 1
Simulation: 8
Visualization: 1

```
&addBroadcast('Tracker','pos','Simulation','pos');
&addConnection('Tracker','pos','Visualization','pos');

&addRoutingNodes('R',&getHosts('Simulation'));
&addRecursiveGather('Simulation','grid','R',Merge2D);
&addConnection('R','','Visualization','grid');
```



Tracker: 1
Simulation: 8
Visualization: 2

```
&addBroadcast('Tracker','pos','Simulation','pos');
&addConnection('Tracker','pos','Visualization','pos');

&addRoutingNodes('R',&getHosts('Simulation'));
&addRecursiveGather('Simulation','grid','R',Merge2D);
&addConnection('R','','Visualization','grid');
```



Tracker: 1
Simulation: 8
Visualization: 4

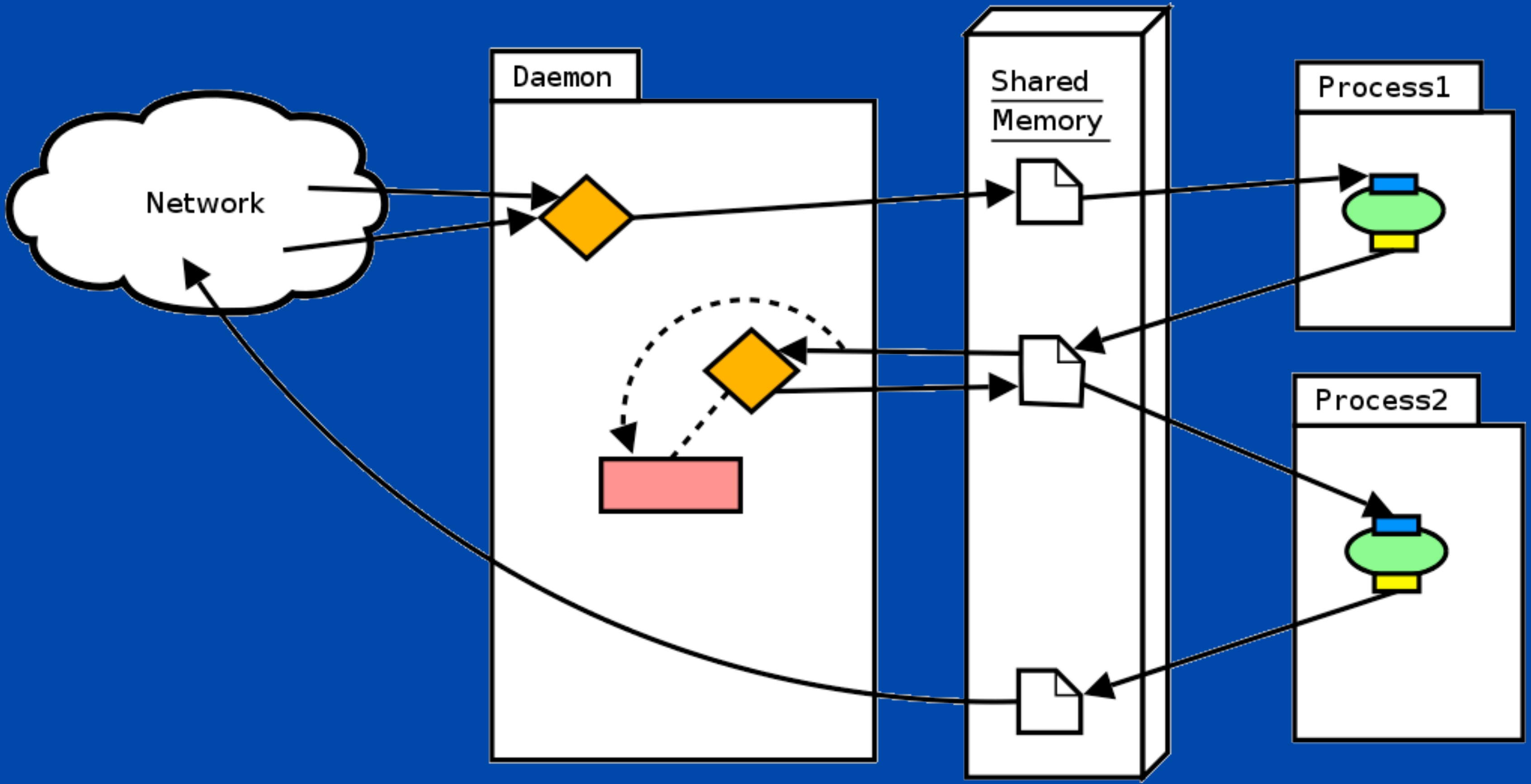
```
&addBroadcast('Tracker','pos','Simulation','pos');
&addConnection('Tracker','pos','Visualization','pos');

&addRoutingNodes('R',&getHosts('Simulation'));
&addRecursiveGather('Simulation','grid','R',Merge2D);
&addConnection('R','','Visualization','grid');
```

FlowVR Runtime Engine

- Each module runs in its own process
- A *daemon* on each node
  - Implements communications
  - Filters and Synchronizers are loaded as plugins
- A *Shared Memory Area* is used to store messages
  - No copy for local communications

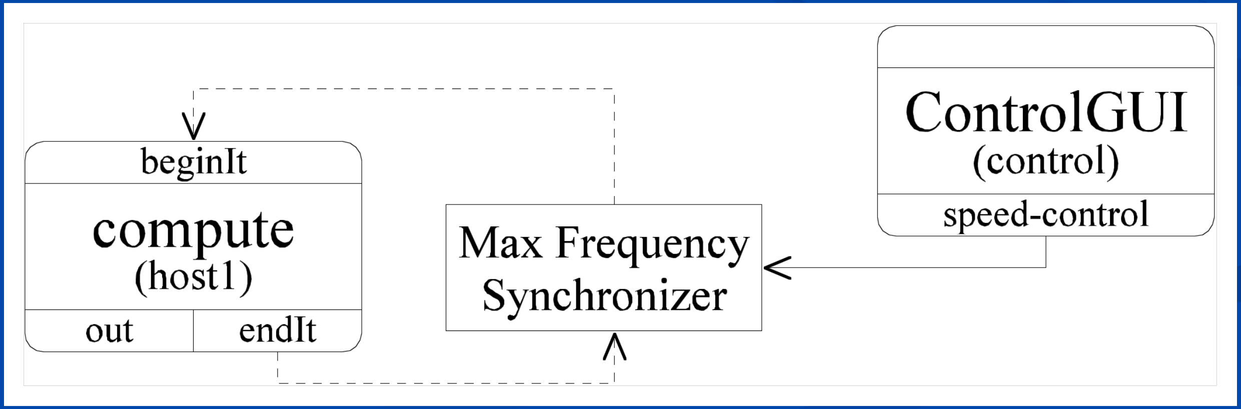
FlowVR : Deamon based

- In charge of the network
- Modules <-> Deamon : Shared memory
- Local communication: pointer exchange



VR Patterns – Control

- Stop/Start Control
  - Activate or disactivate part of the application
- Frequency Control
  - Regulate a module frame rate



Example of Application

Multi Camera Acquisition

↓

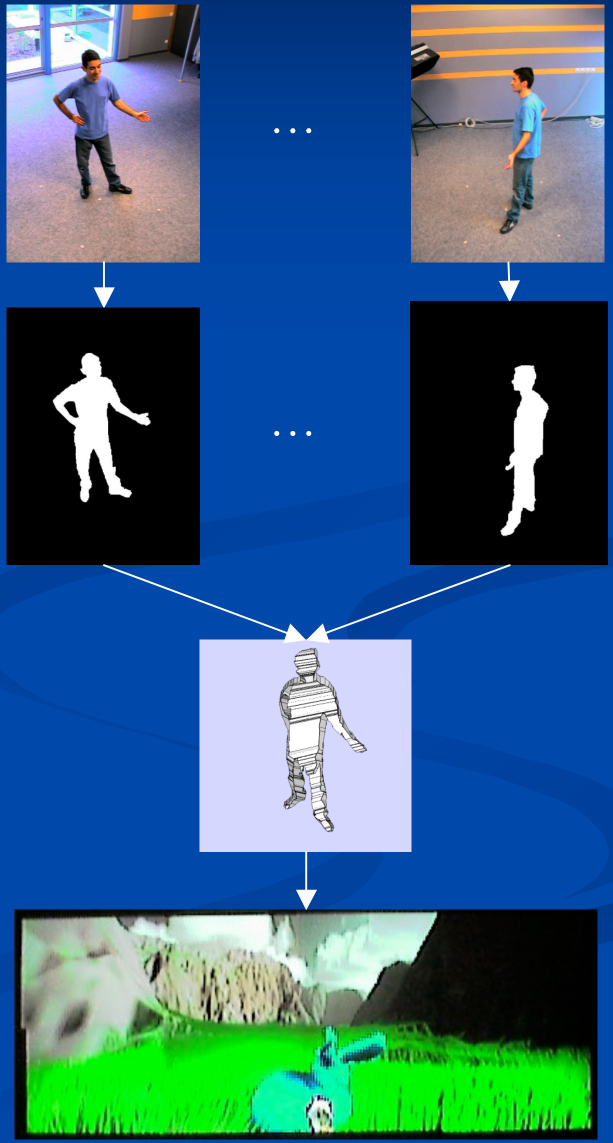
Background Subtraction

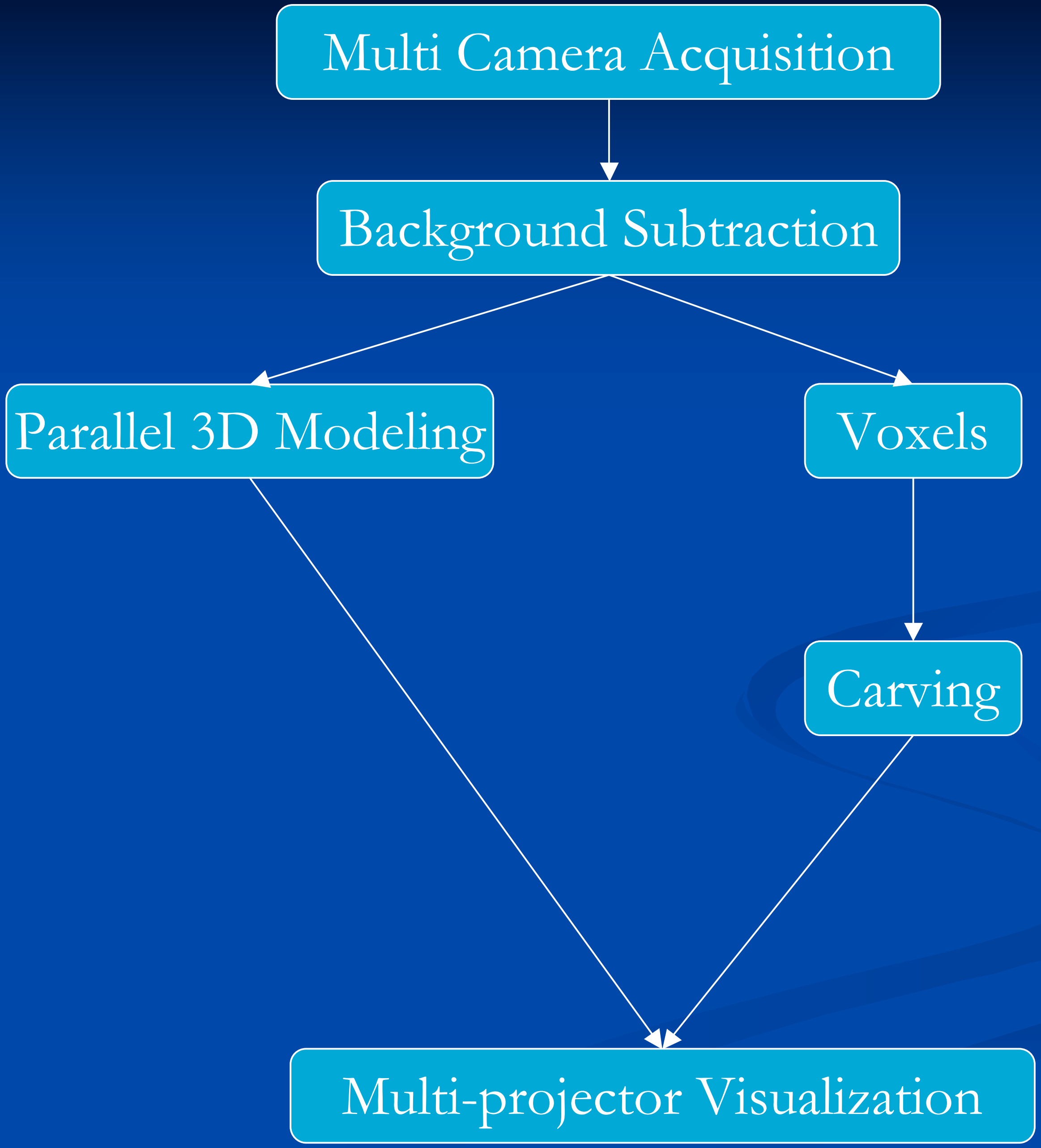
↓

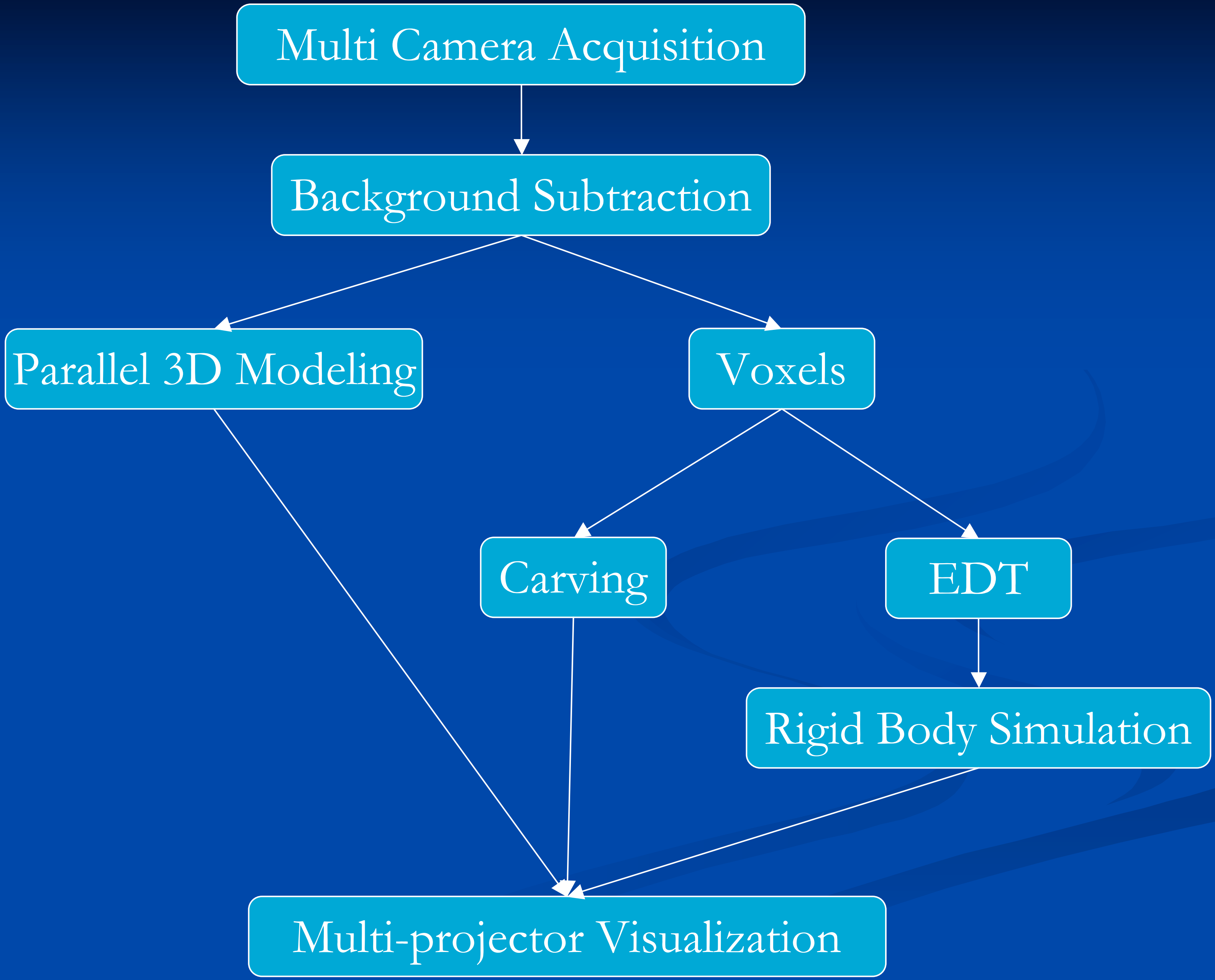
Parallel 3D Modeling

↓

Multi-projector Visualization

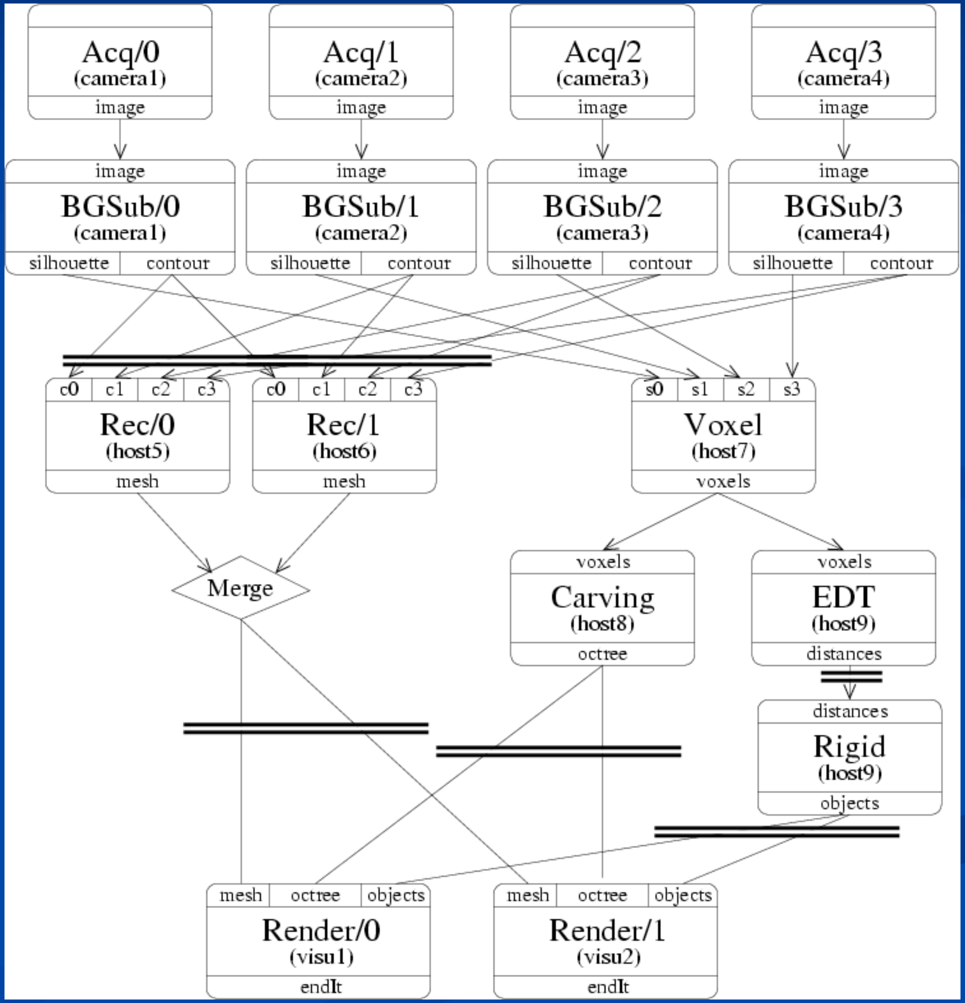



$$\begin{tikzcd}
 & \text{Multi Camera Acquisition} \arrow[d] & \\
 & \text{Background Subtraction} \arrow[dl] \arrow[dr] & \\
\text{Parallel 3D Modeling} \arrow[ddr] & & \text{Voxels} \arrow[d] \\
 & & \text{Carving} \arrow[dl] \\
 & \text{Multi-projector Visualization} &
\end{tikzcd}$$


Multi Camera Acquisition
Background Subtraction
Parallel 3D Modeling
Voxels
Carving
EDT
Rigid Body Simulation
Multi-projector Visualization

Application Network

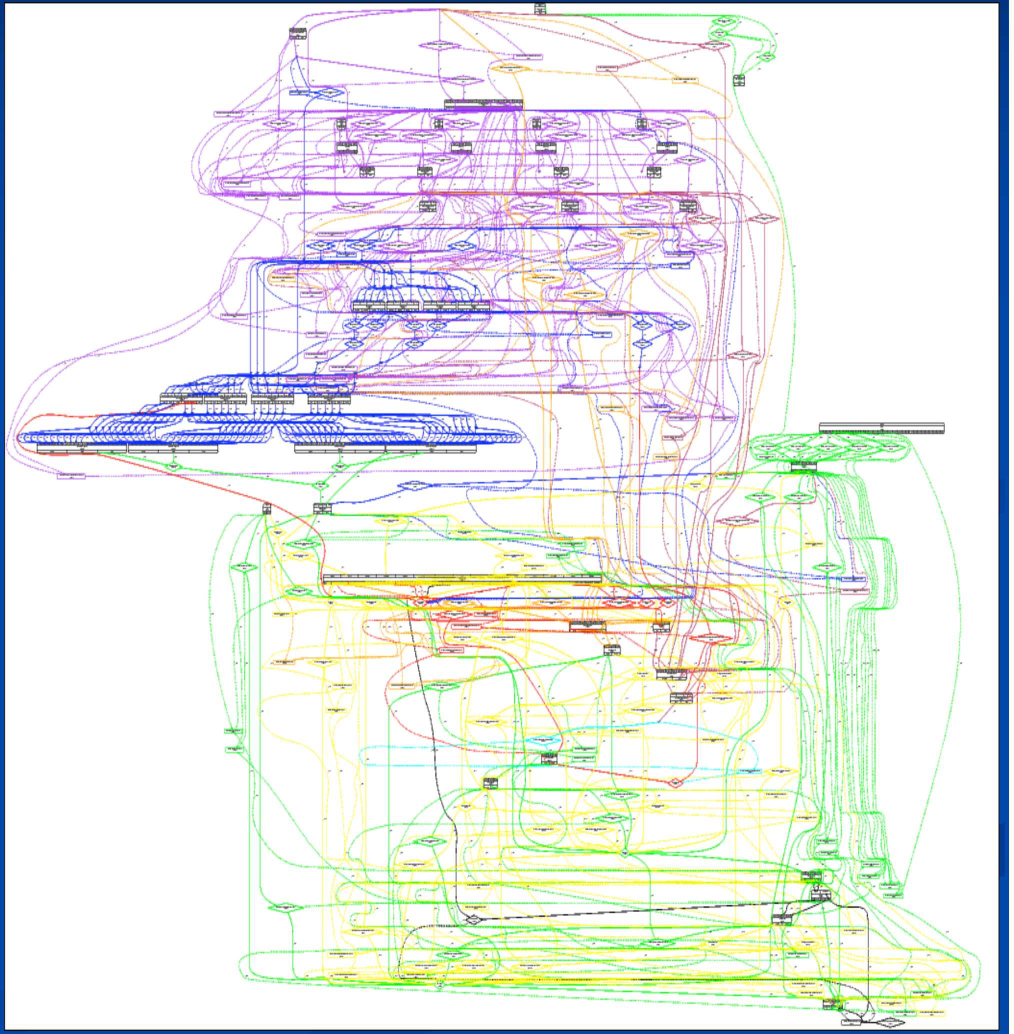
- Module Replication
  - Acquisition
  - Background Subtraction
- Module Parallelisation
  - Reconstruction
- Coherent Greedy
  - Rendering


Acq/0 (camera1) image
Acq/1 (camera2) image
Acq/2 (camera3) image
Acq/3 (camera4) image
image BGSub/0 (camera1) silhouette contour
image BGSub/1 (camera2) silhouette contour
image BGSub/2 (camera3) silhouette contour
image BGSub/3 (camera4) silhouette contour
c0 c1 c2 c3 Rec/0 (host5) mesh
c0 c1 c2 c3 Rec/1 (host6) mesh
s0 s1 s2 s3 Voxel (host7) voxels
Merge
voxels Carving (host8) octree
voxels EDT (host9) distances
distances Rigid (host9) objects
mesh octree objects Render/0 (visu1) endIt
mesh octree objects Render/1 (visu2) endIt

Application Network

- Module Replication
  - Acquisition
  - Background Subtraction
- Module Parallelisation
  - Reconstruction
- Coherent Greedy
  - Rendering

Network script: 100 lines



Large Scale Application

- Module pool: 20 modules (based on existing codes)
- Network script: 1000 lines
- Code re-use: 4 persons during 6 months

- Execution on Grimage:
  - 200 processes automatically launched
  - 4000 connections
  - 500 filters (from various VR patterns)

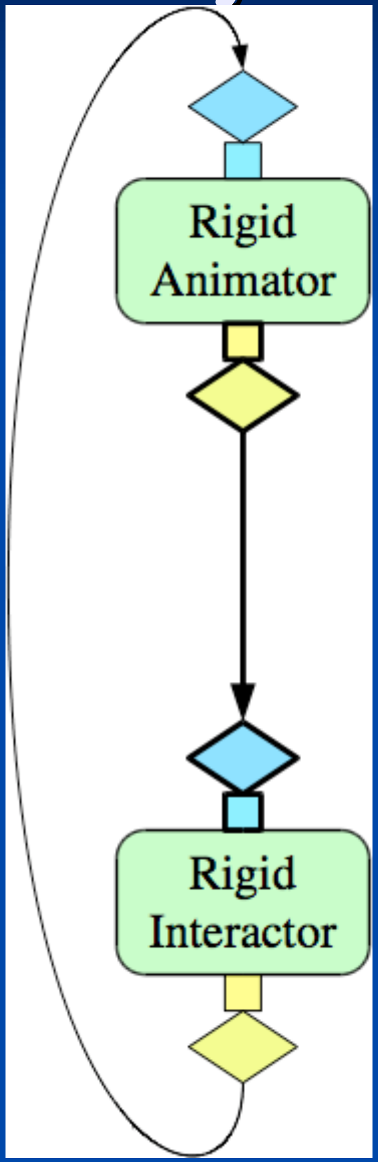
Flowvr-render

- A layer built on top of Flowvr to transport graphics primitives.

Physical-Based Animations

- Issue: how to build a large animated scene in a mdoular and efficient way ?
- Flowvr-VR Interact: A framework for distributed physical-based simulation based on 2 main components:
  - Animators : store and update object properties
  - Interactors : compute forces that apply to an object

Physical-Based Animations



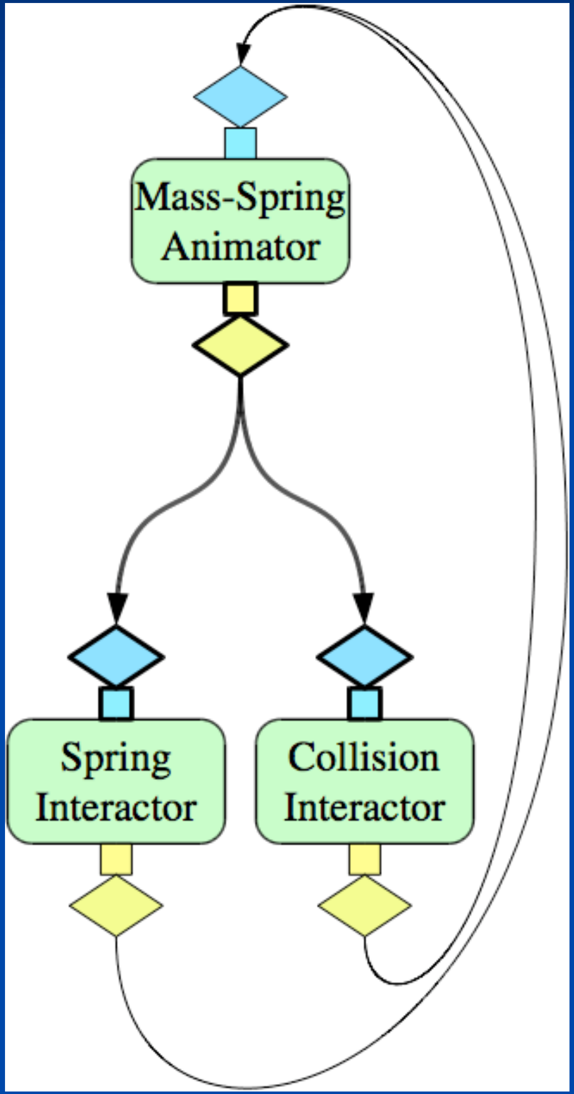
Update object properties

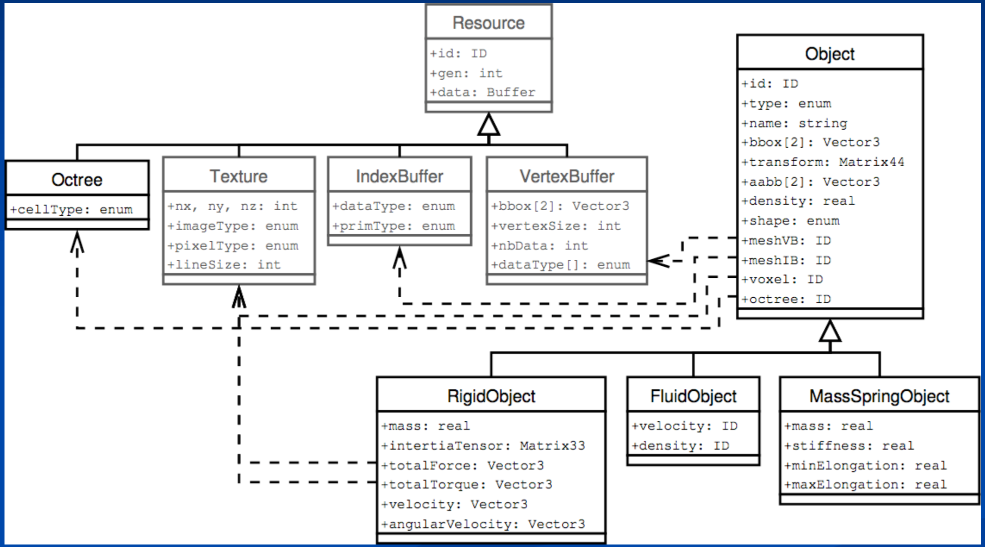
Object description (position, speed,masse, etc.)

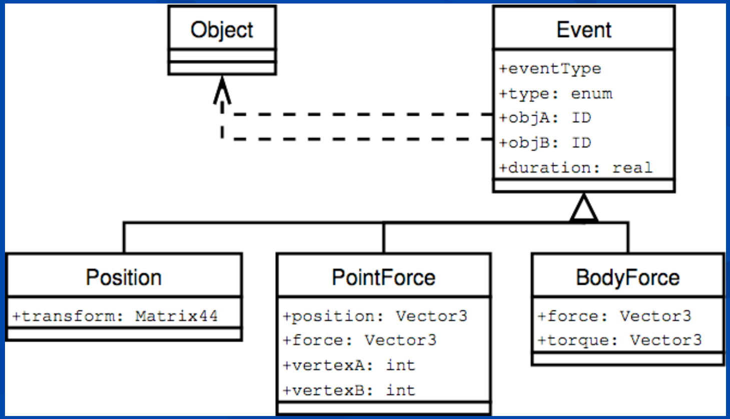
Compute forces

List of forces to apply

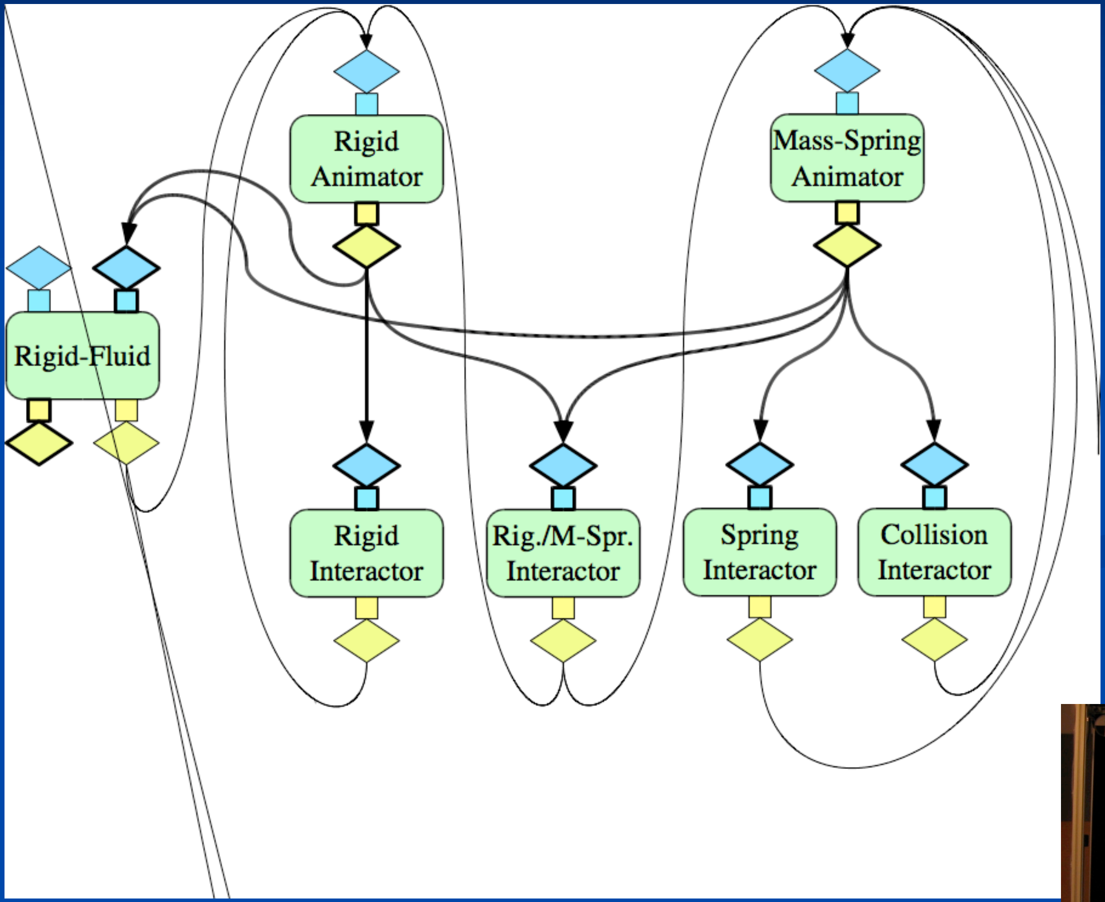
Physical-Based Animations

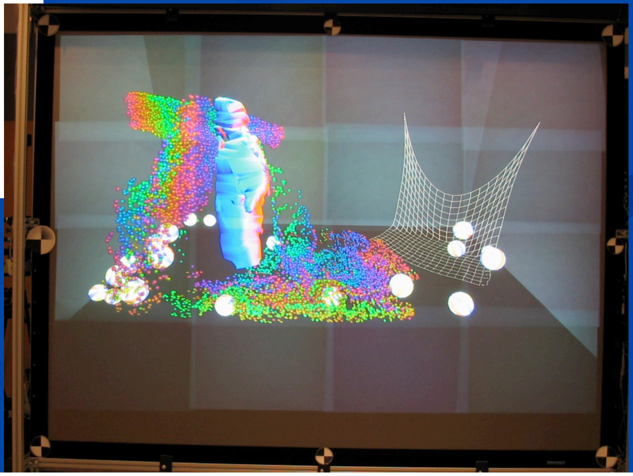






Physical-Based Animations





Conclusion

- FlowVR:
  - An empty shell : MPI, Jiggle, VTK, VR Juggler, QT, VRPN
  - Learning curve: 2 weeks
- Modularity:
  - Favor code re-use
- Efficiency:
  - Shared memory and Zero copy protocol
  - Advanced distributed network schemes
- Scalability:
  - Up to 54 processors (going for 200 hundreds)

Grimage Platform

- GrImage:
  - Display Wall:
    - 16 Video-projectors
  - PC Cluster (54 processors)
  - 15 Cameras

  - Markerless 3D Modeling [IPT04]
  - Interactions between the virtual world and the actor

Information

- FlowVR – 1.2
  - http://flowvr.sf.net
- GrImage
  - http://www.inrialpes.fr/sed/grimage

To come soon:

- Flowvr-Render: shader based distributed rendering (IEEE Vis 2005)
- Flowvr-mplayer: parallel video player