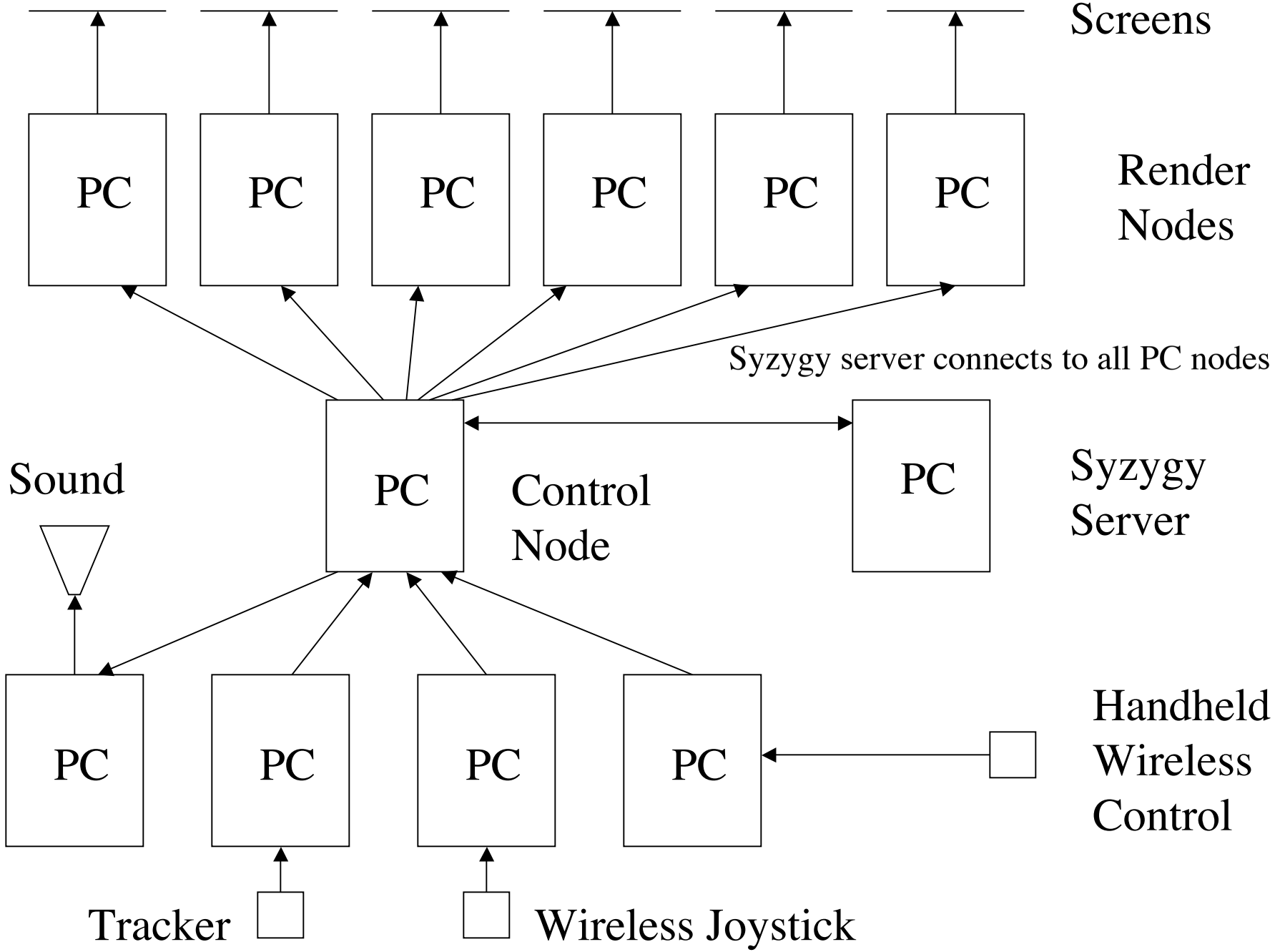
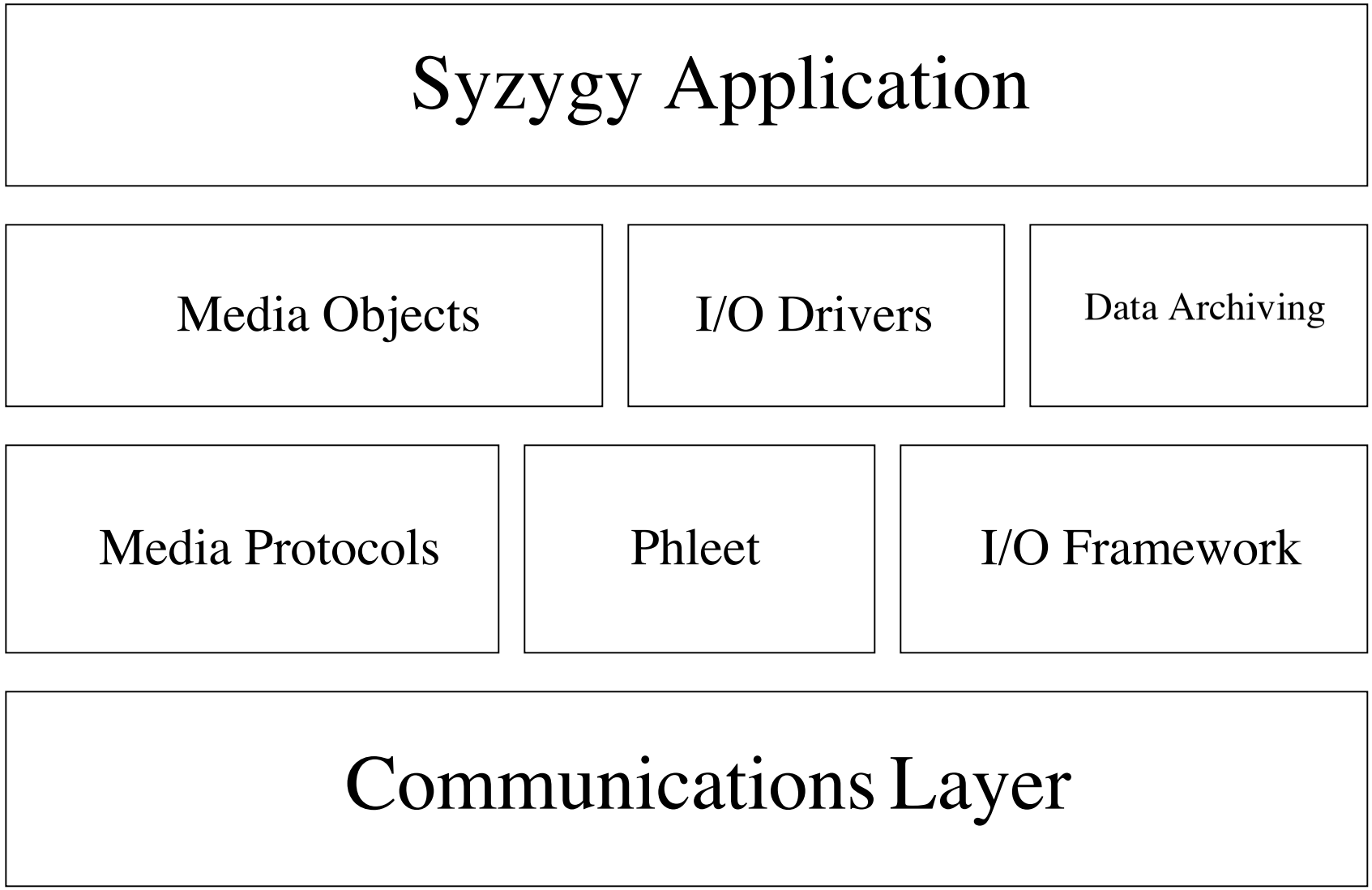
Syzygy

- Design overview
- Distributed Scene Graph
- Master/slave application framework
- I/O Device Integration using Syzygy
- Scaling down: simulators and other development methods using Syzygy

The Problem

- Cluster-based VR leads to a bewildering variety of computers and devices!
- Management: How to quickly move from demo to demo… if each requires an entirely different set of software resources?
- Connections: How to integrate the different devices? The system must be robust, self-assembling, and self-healing!
- Options: Severe communications bottlenecks mean many approaches to cluster-based VR are appropriate.


Screens
PC
PC
PC
PC
PC
PC
Render
Nodes
Syzygy server connects to all PC nodes
Sound
PC
Control
Node
PC
Syzygy
Server
PC
PC
PC
PC
Handheld
Wireless
Control
Tracker
Wireless Joystick


Syzygy Application
Media Objects
I/O Drivers
Data Archiving
Media Protocols
Phleet
I/O Framework
Communications Layer

Communications Layer

- Message based. Binary format.
- Automatic translation between machine architectures (endianess and byte alignment)
- Server, client objects with extensive connection management functionality.
- Distributed barrier objects.
- By default, connections can occur in any order between objects, be broken, reformed, etc. Robust.
- Includes simple abstraction layer to ease writing single source Unix/Win32 applications.

Phleet

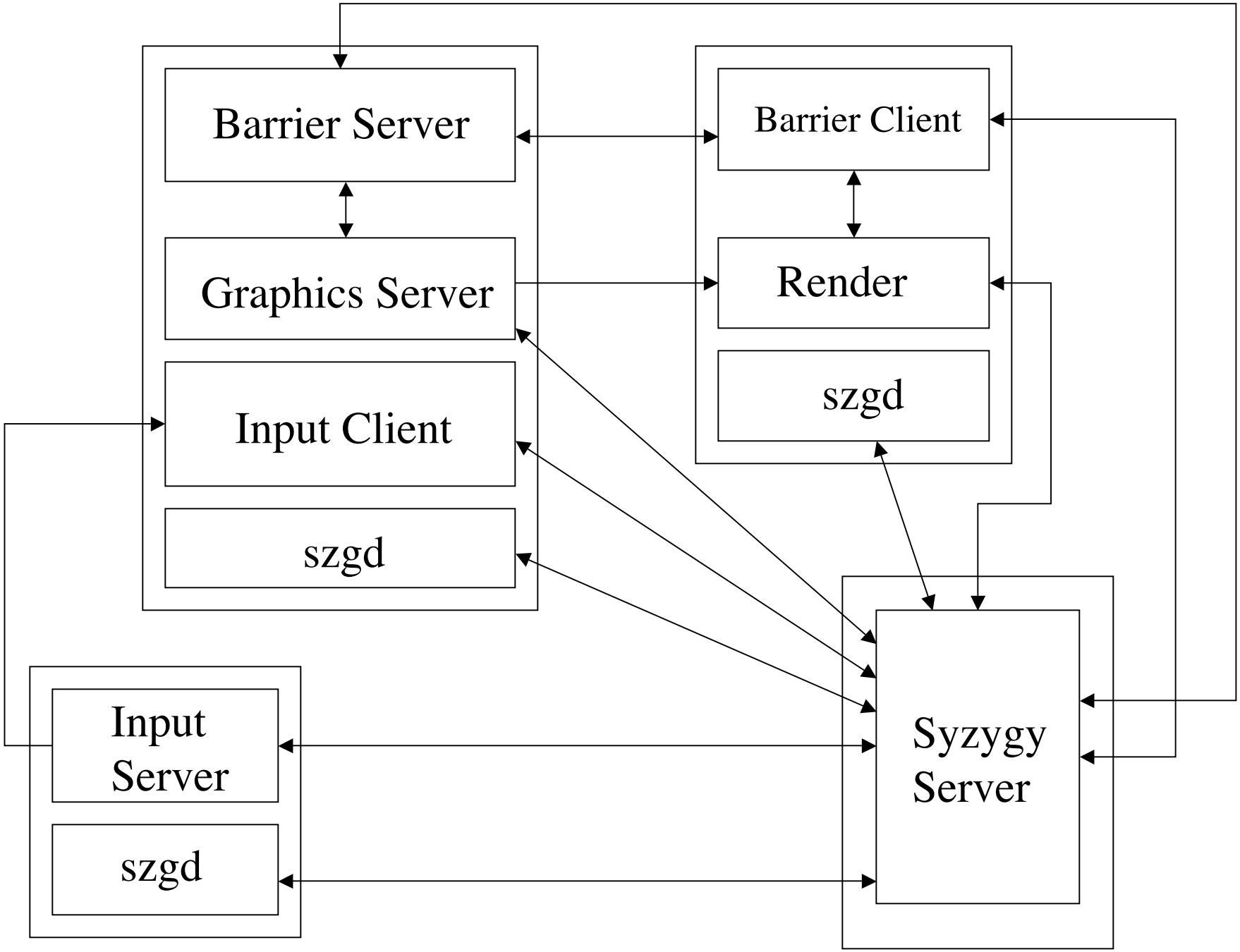
- **Distributed operating system (minimal)**
- **szgServer controls.**
- **szgd provides remote execution services.**
- **Parameter database stored in szgServer. Meta-config file that can be managed from the command line of any computer.**
- **Global locks**
- **Simple message API routes through szgServer to all managed programs (like Unix signals). Can tell a program to reload its parameters or exit.**
- **Otherwise szgServer is merely a connection broker. Most communication occurs directly between programs.**

Media

- **Protocols not dependent on Phleet. Can implement glue code yourself (use only what you want).**
- **Built using the tools in the communications layer. Based on synchronized message buffer transfers.**
- **Distributed scene graph**
- **Master/slave application framework**
- **Sound**
- **VR-specific functionality (like calculation of view frusta)**
- **When integrated using phleet into media objects, one has a powerful self-configuring, fault-tolerant system.**

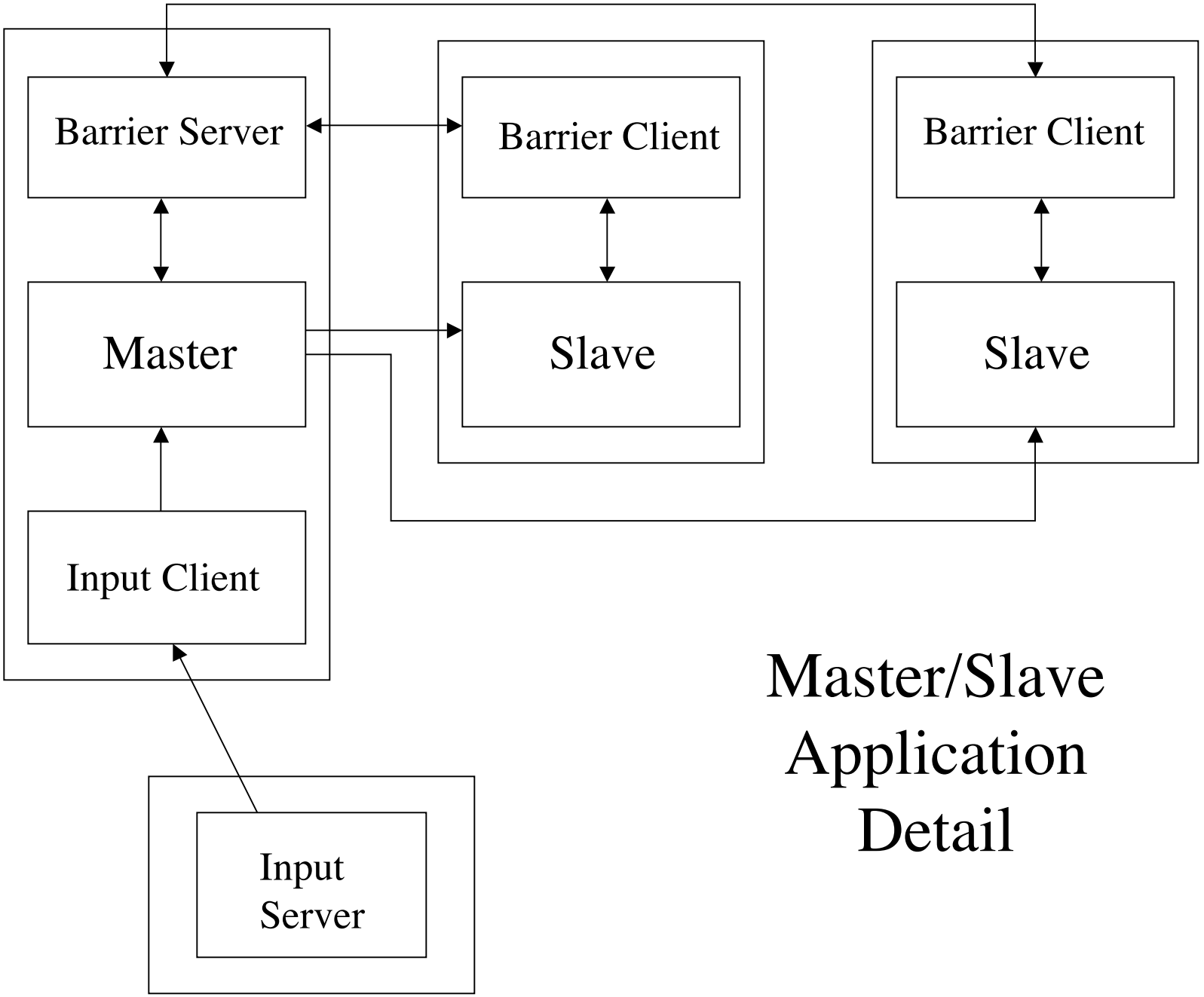
Distributed Scene Graph

- Client/server. One node serves geometry to the render nodes.
- Bandwidth is automatically conserved. Only scene changes are sent from one frame to the next.
- Semantic nature of scene graph helps eases programming.
- Robust. Render clients can disconnect and reconnect while the application is running. Dynamically change your display. Similarly, stop your application and start a new one without bringing down the render clients.


Barrier Server
Graphics Server
Input Client
szgd
Barrier Client
Render
szgd
Input
Server
szgd
Syzygy
Server

Master/Slave Application Framework

- Master/slave = same application runs synchronized on render nodes.
- One node is the master and distributes I/O or other control info.
- Framework makes it easy to write a synchronized application, handle input devices, and manage the whole thing using phleet.
- A good way to write a custom distributed graphics protocol!



Master/Slave Application Detail

I/O Device Integration

- Built on the same communications layer as everything else.
- Robust client, server objects make it easy to get device data onto the network and into your program. Put that Windows joystick on the network!
- Phleet allows configuration of the multitude of I/O devices and helps them make connections.

Simulators

- Need a reasonable replacement for the VR device in order to do debugging, etc.
- Random PCs around the office or lab provide multiple screens. A “wand sim” program lets you generate tracker data from the desktop. Really, we only simulate the input device. The rest of the system is unchanged.
- Hard guarantees ensure that if your program runs correctly on one graphics pipe, it will run correctly on 6 or more.

Free Software

- Syzygy is licensed under the GNU LGPL
- It comes with everything you need to start showing demos right away!
- Download the latest release at www.isl.uiuc.edu (follow the software link)