

Software Architecture for Immersion

Alexandre R.J. François
Computer Science Department
alexandre.francois@usc.edu

© ARJF 2006



USC **Viterbi**
School of Engineering

Software Architecture



- Design, analysis and implementation of software systems
 - Improve the flexibility and comprehensibility of software systems (Parnas, 1972)
 - Address modularization as a design issue (Parnas)
- Explicit system structure
 - Technical basis for design
 - Provable properties
 - Blue-prints for implementation
 - Tools for analysis
- Project management
 - Separation of concerns
 - Planning: cost estimation, resource allocation

Immersipresence



- Vision of the Integrated Media System Center
 - NSF ERC in Multimedia, est. 1995-96
- “Combine real world with virtual world”
 - Experience immersion, presence
 - Interact naturally
 - Collaborate through shared virtual/augmented space
- Build systems capable of:
 - Handling video, sound, haptics, etc.
 - Real-time analysis/synthesis (immersion)
 - Low latency (interaction)

Requirements for Immersipresence



- **Interoperability**
 - Combine research from different fields/teams
- **Efficiency**
 - On-line, real-time, low latency
- **Scalability**
 - Performance evaluation and prediction
- **General model for *distributed asynchronous concurrent processing of data streams***

SAI Principles

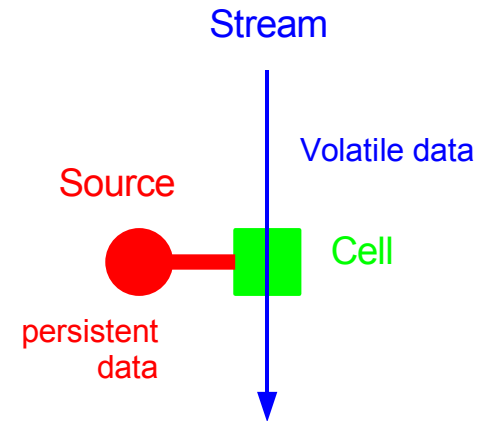


- Time
- Volatile vs. persistent data
- Asynchronous concurrent processing

- Architectural style [ICSE2004]
 - High level abstractions
 - Hybrid model or more general model?

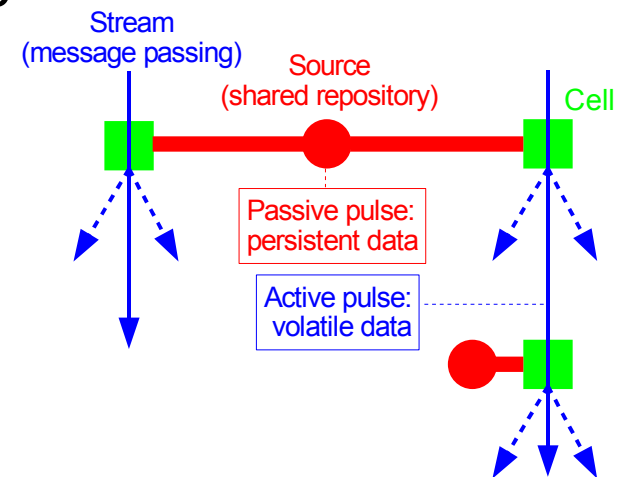
Architectural Primitives

- Stream
 - Volatile data
- Cell
 - Processing unit (no state)
 - Asynchronous parallel model
- Source
 - Shared repository of persistent data
- Pulse
 - Synchronization structure (time stamp, duration)
 - Active: volatile, flow down stream connections
 - Passive: persistent (dynamic), held in sources



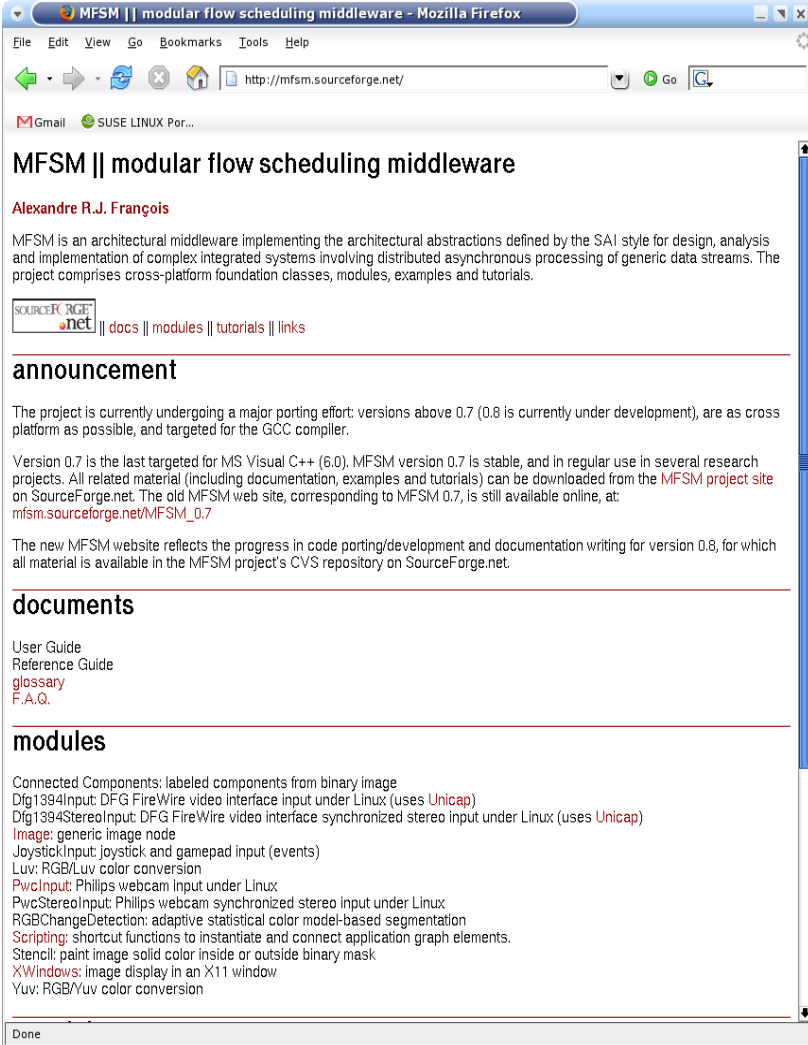
Constraints and Semantics

- Cell-cell: volatile data samples (stream)
 - At most one upstream cell
 - Any number of downstream cells
 - Process dependency
- Cell-source: persistent data access
 - Exactly one source to a given cell
 - Any number of cells to a given source
 - Concurrent access
- Processing model
 - Active pulse triggers cell process
 - Process may add to active pulse
 - Process may modify passive pulse



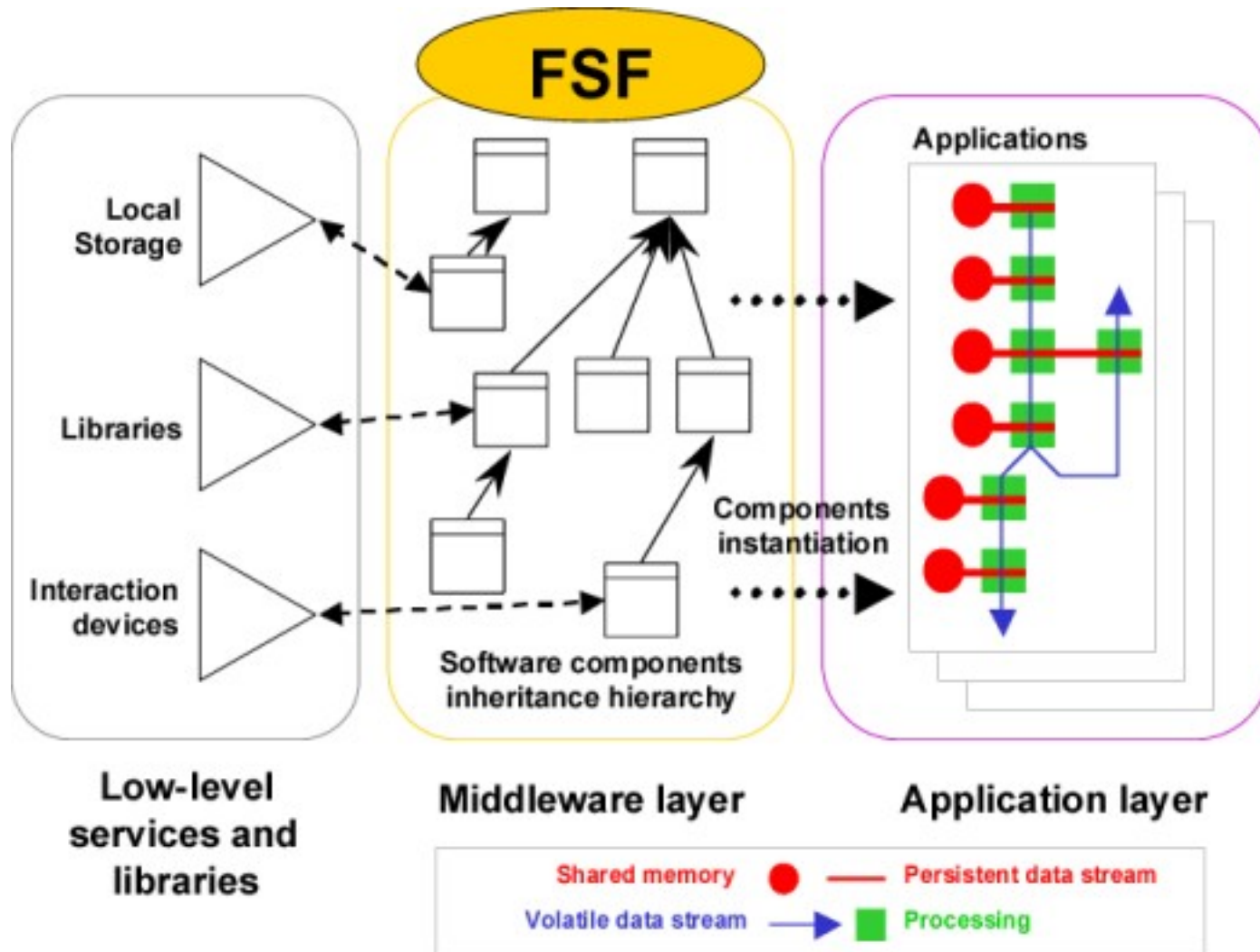
Architectural Middleware

- Support architectural abstractions
 - Pulse, source, cell, etc.
 - Direct mapping from logical specification to code!
- Modular Flow Scheduling Middleware (MFSM)
 - Open source project: mfsm.SourceForge.net
 - C++, cross-platform
 - (GNU compiler)
 - Base library, functional modules, documentation, tutorials

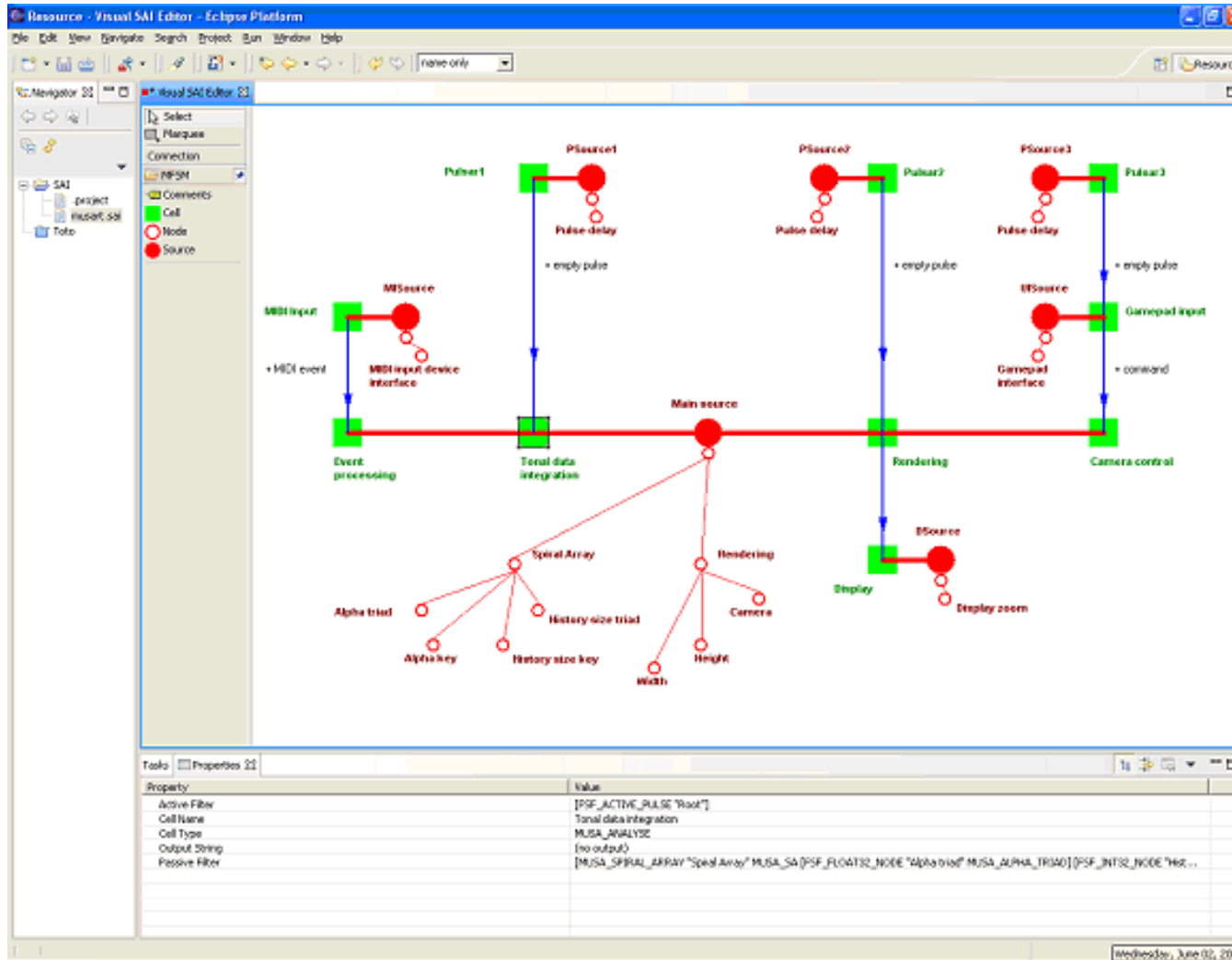


The screenshot shows a Mozilla Firefox browser window displaying the MFSM project page on SourceForge.net. The page title is "MFSM || modular flow scheduling middleware". The author is listed as Alexandre R.J. François. The page content includes a description of MFSM as an architectural middleware implementing the SAI style, a list of navigation links (docs, modules, tutorials, links), and sections for announcements, documents, and modules. The announcements section mentions a major porting effort to GCC and the release of version 0.7. The documents section lists a User Guide, Reference Guide, glossary, and F.A.Q. The modules section lists various input and processing modules like Dfg1394Input, Dfg1394StereoInput, Image, JoystickInput, Luv, PwcInput, PwcStereoInput, RGBChangeDetection, Scripting, Stencil, XWindows, and Yuv.

MFSM Diagram



VisualSAI

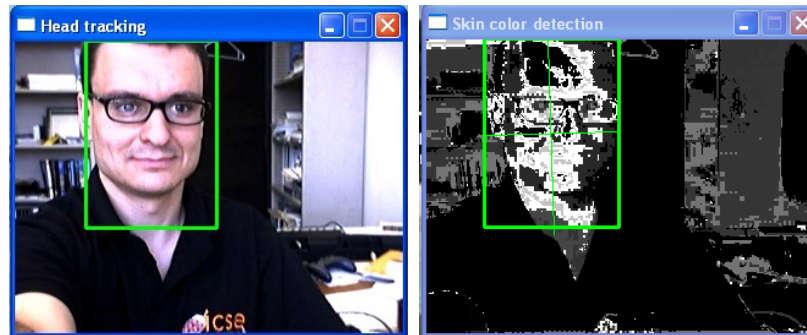


Wednesday, June 03, 2004

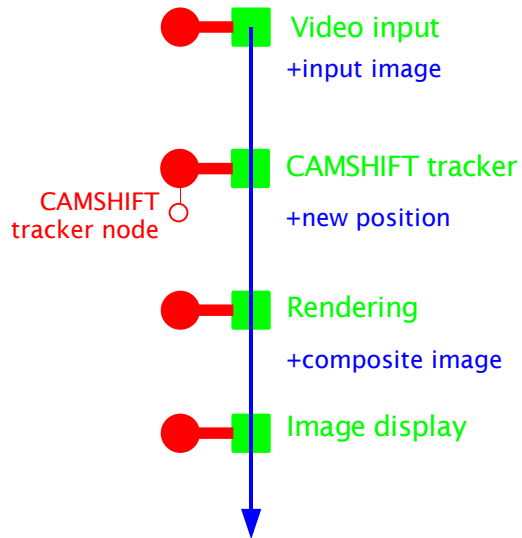
CAMSHIFT Tracking



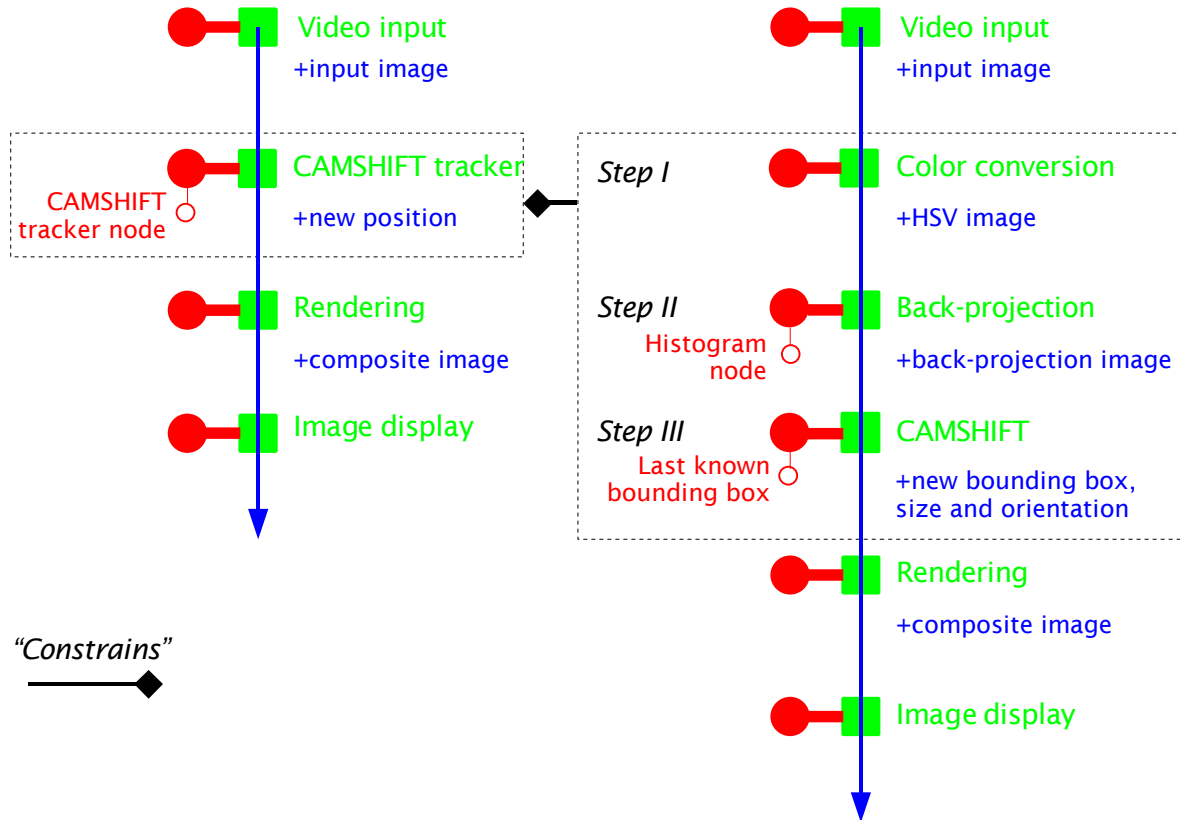
- Tracking algorithm: CAMSHIFT
 - Continuous Adaptive Mean SHIFT (Bradski, 1998)
 - Mean shift: iteratively find the mode in a probability density distribution (Comaniciu & Meer, 1997)
- Perceptual User Interface
 - Real-time video processing



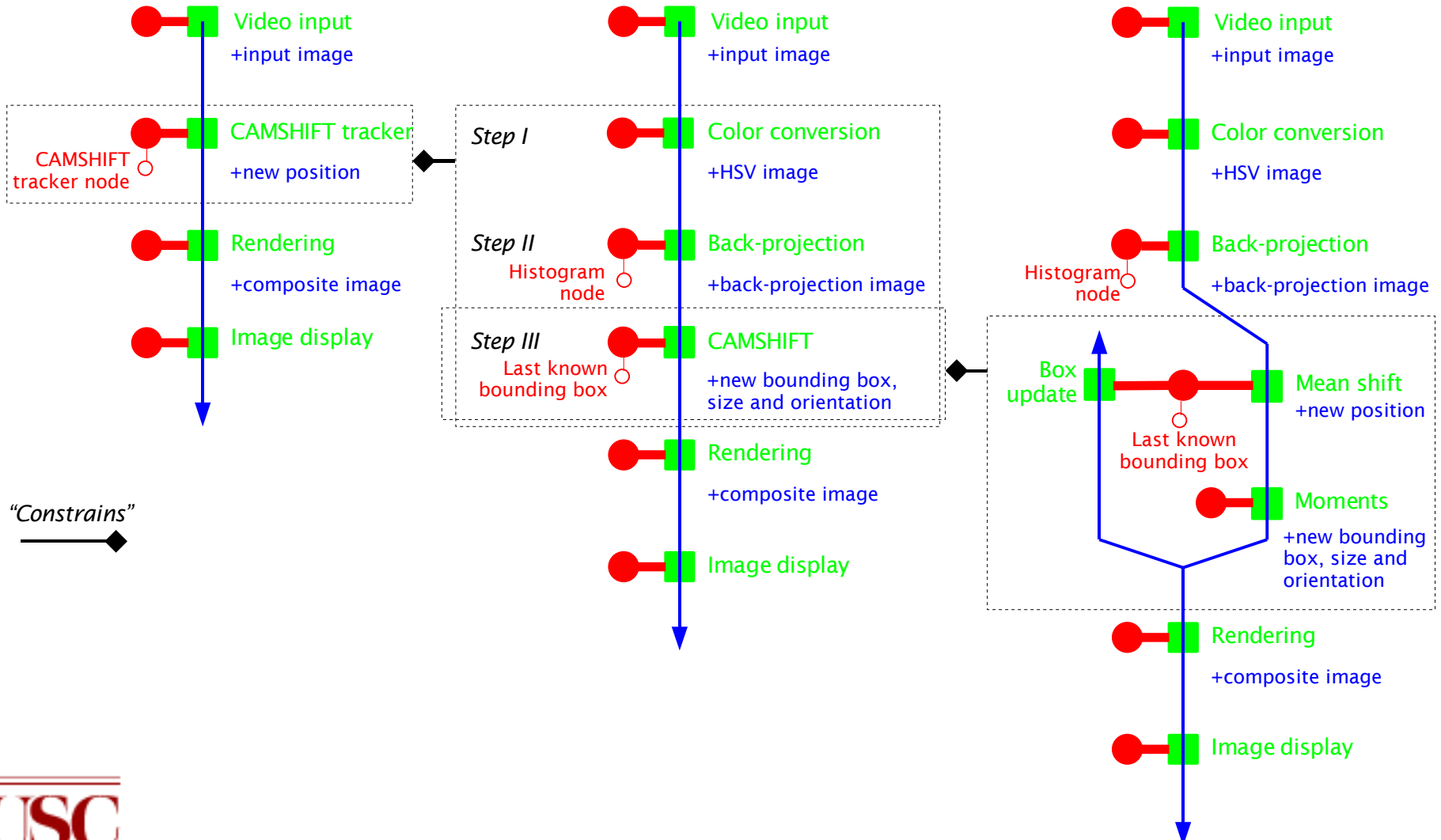
Example System



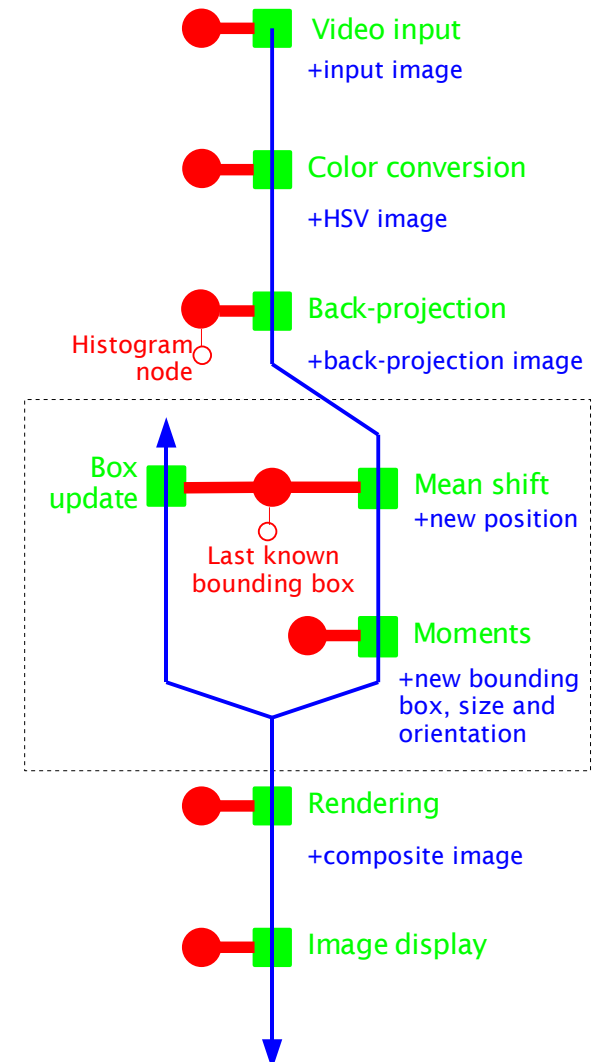
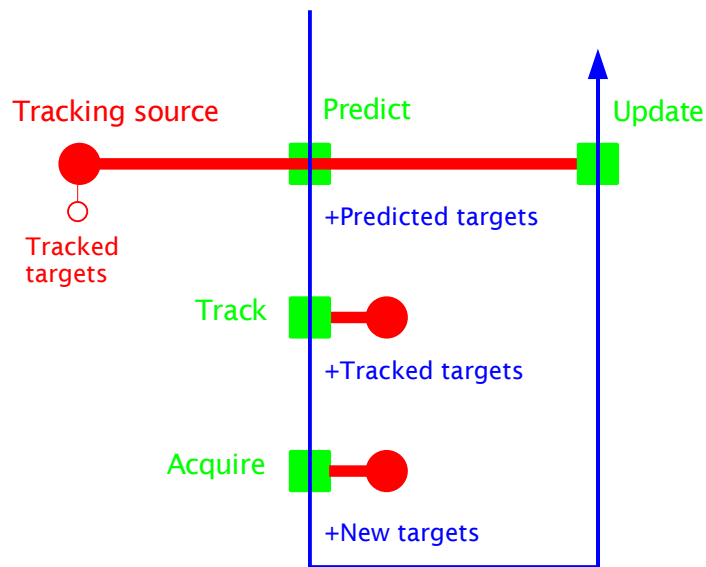
Refinement 1



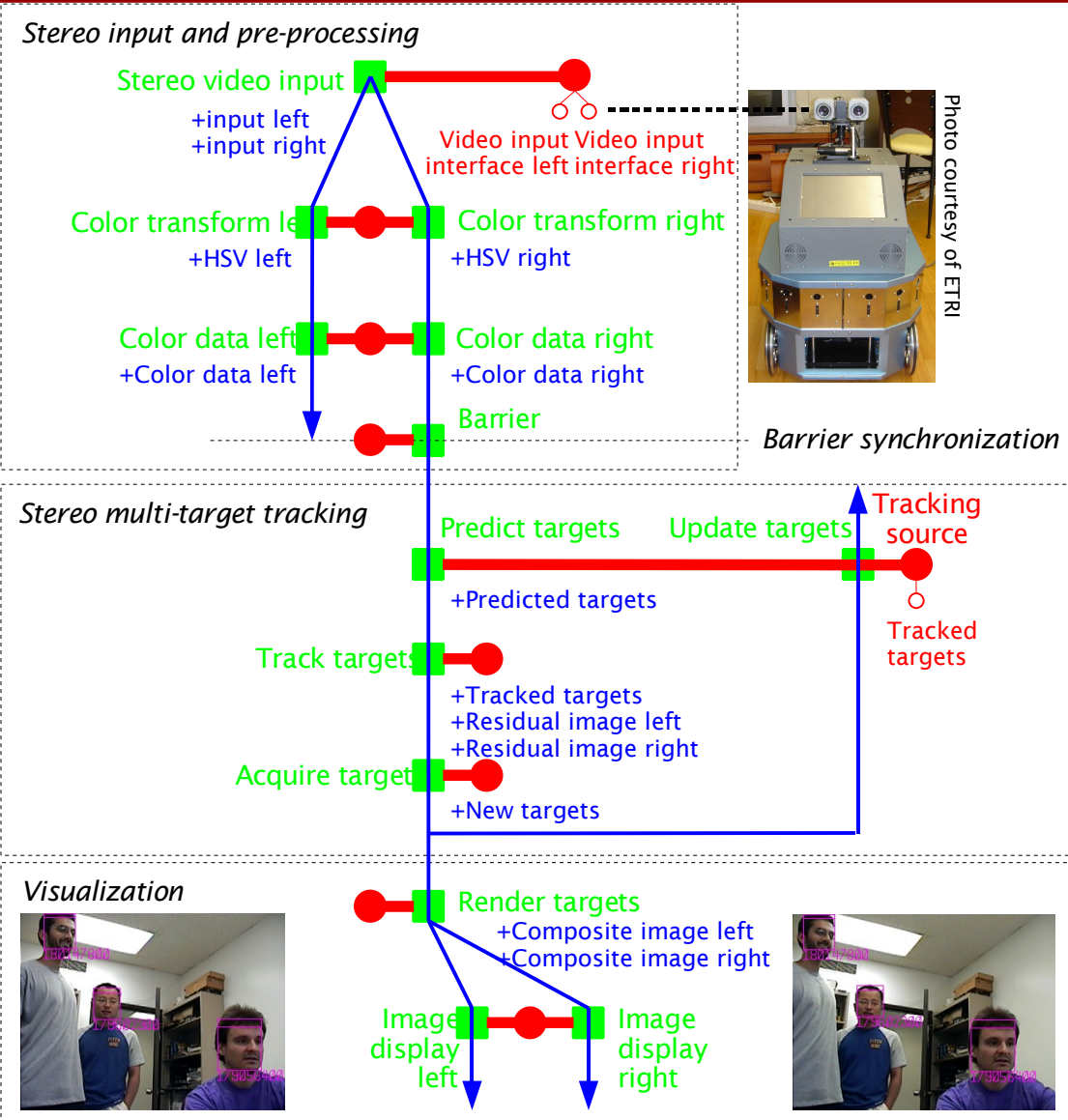
Refinement 2



General Tracking Pattern



Vision for Robot: Stevi 1

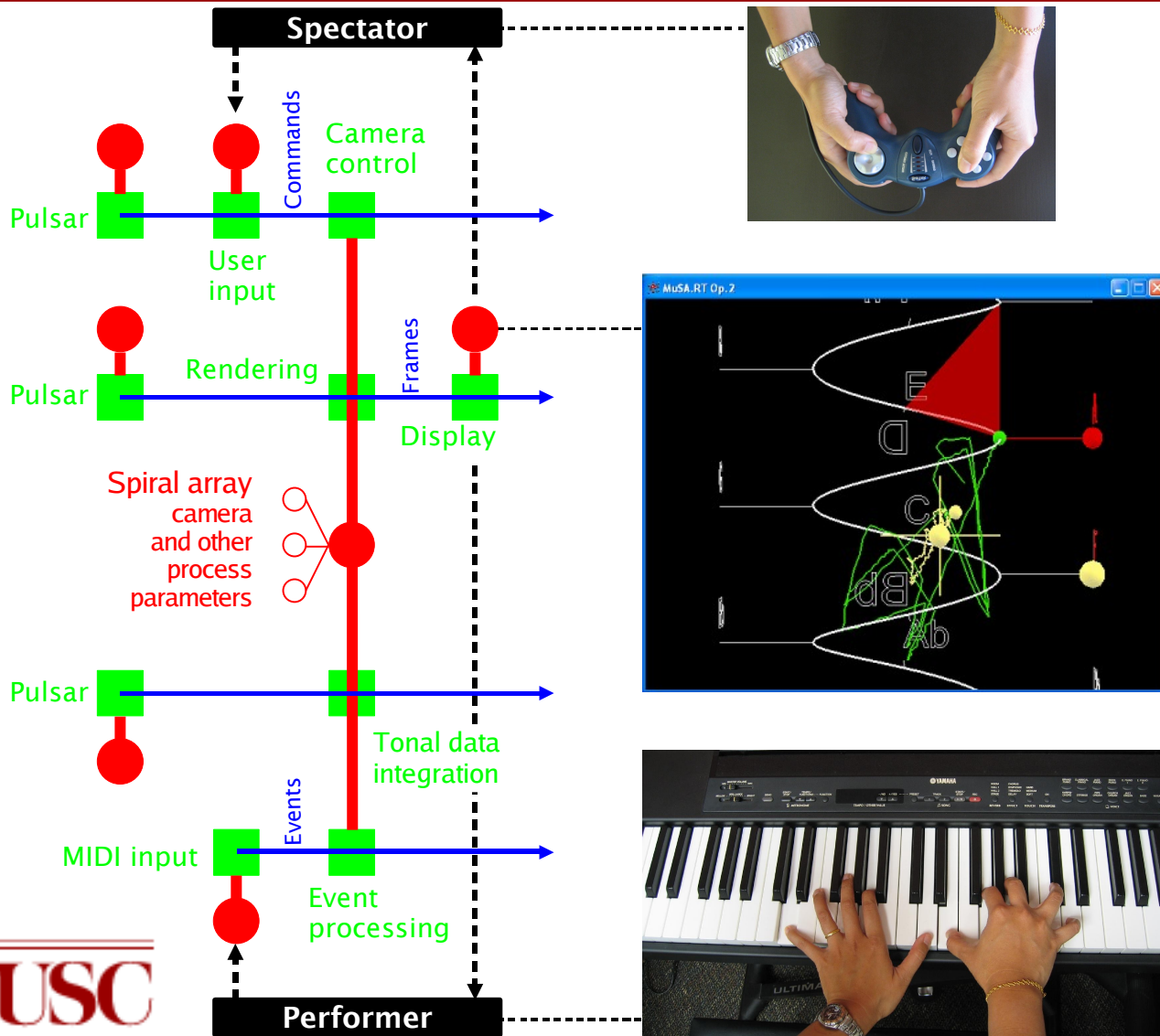


MuSA.RT

Music on the Spiral Array . Real Time



Co-PI: Elaine Chew

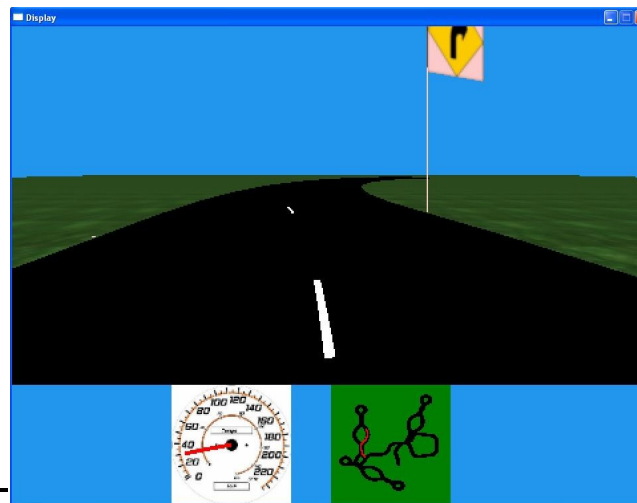
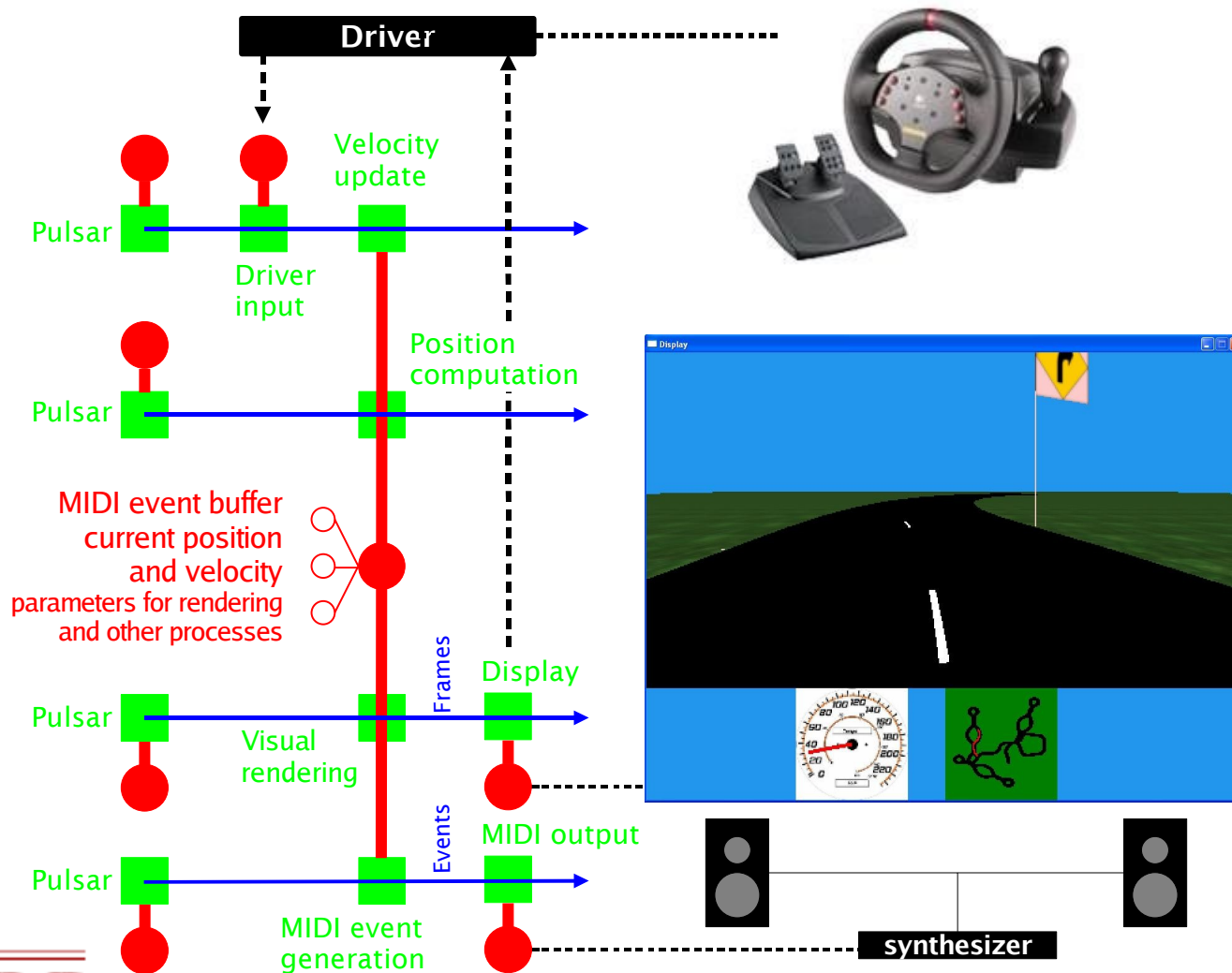


ESP

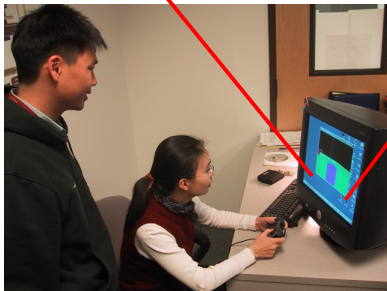
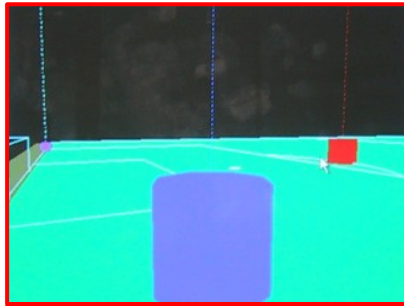
Expression Synthesis Project



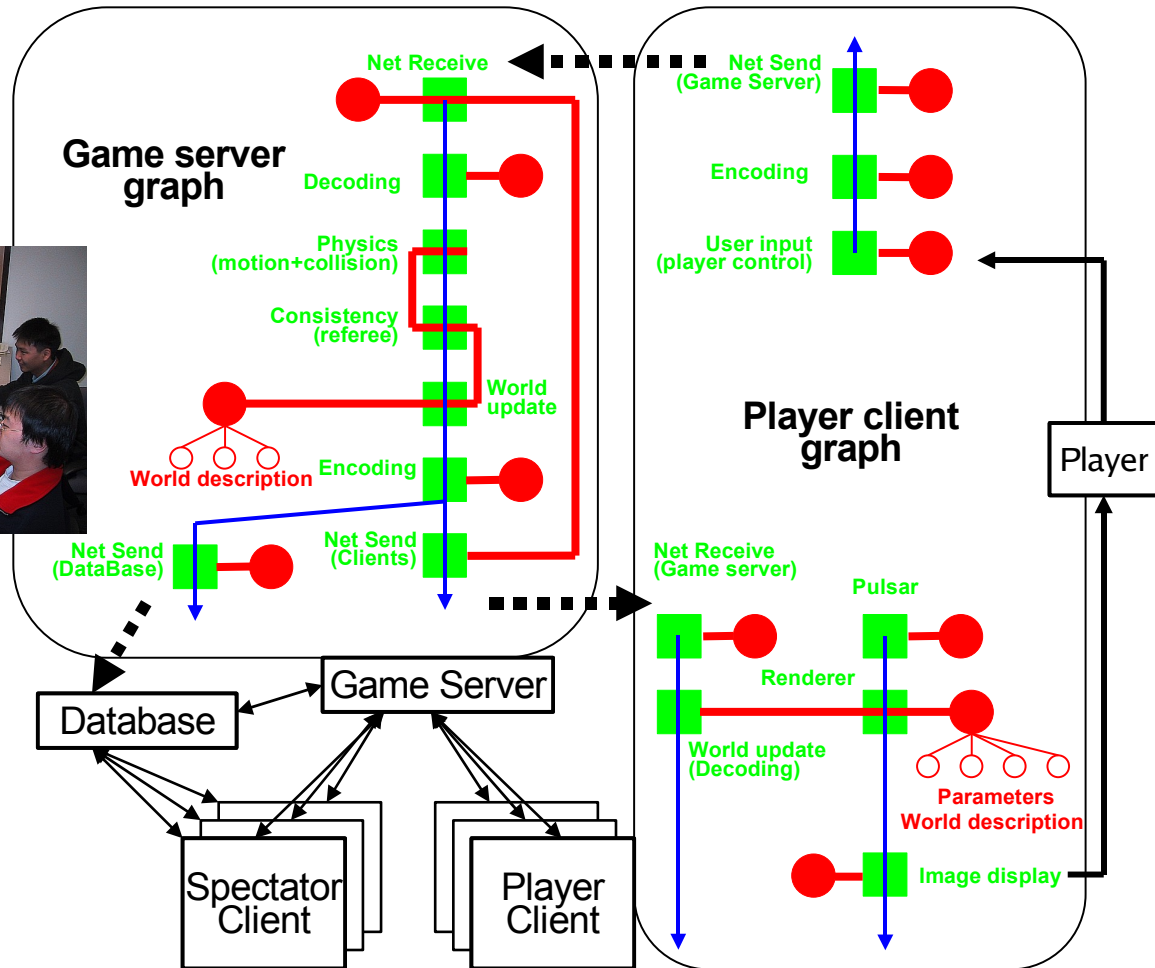
Co-PI: Elaine Chew



Distributed Game Project



25 students, 2 months
 Distributed development
 Real-time multiplayer gaming
 with database
 recording/replay



Summary



- **SAI: Software Architecture for Immersipresence**
 - Design and analysis of complex software systems
 - MFSM: Architectural middleware
 - Patterns for synchronization
- **Applications:**
 - Interactive music systems, Computer vision and graphics systems, Distributed Interactive games, etc.
- **For more information:**
 - <http://iris.usc.edu/~afrancoi>
 - <http://mfsm.sourceforge.net>

SAI Properties (1)



- Model time explicitly in data and processing
- Model modularity
 - Separation of concerns
 - Scalability
- Model concurrent execution (asynchronous)
 - Separate throughput and latency
- Model distributed computing

SAI Properties (2)



- Facilitate system design
 - Intuitive architectural style, based on data streams
 - Unified processing model and unified data model
 - Design patterns
- Facilitate system analysis
 - Safety, liveness, etc.
- Facilitate distributed development
 - Fast integration
 - Code reusability
- Facilitate system maintenance, modification and evolution
 - Change in algorithm and in function

Acknowledgments and Disclaimer



- The research presented here was funded in part by:
 - The Integrated Media Systems Center, an NSF Engineering Research Center, Cooperative agreement No. EEC-9529152.
 - The Advanced Research and Development Activity of the U.S. Government under contract No. MDA-908-00-C-0036
- Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect those of the funding agencies.