

Interaction and Computation

Alexandre R.J. François

Computer Science Department
alexandre.francois@usc.edu

© ARJF 2006



USC **Viterbi**
School of Engineering

Definitions



- **Interaction**
 - Mutual or reciprocal action, effect, or influence
- **Interact**
 - Act together or towards others or with others
- **Computation**
 - An act, process, or method of computing; calculation
 - A result of computing
 - The amount computed
 - The act of operating a computer
- **Compute**
 - To determine by mathematics, especially by numerical methods; calculate
 - To determine by the use of a computer

Computation



- Algorithms
 - Computable functions
 - Turing machine
 - Input- \rightarrow (p)- \rightarrow output
- Calculations
 - Exact
 - Repeatable
- Time irrelevant
 - Synchrony
 - Sequentiality
 - Lock-step

Mathematics

Accounting

Calculation

Interaction



- **Communication**
 - Experience
 - Influence
 - Constant feed-back
- **Perception**
 - Robust, non-exact
 - Multi-modal
- **Dynamic**
 - Causality
 - Concurrency
 - Time metric

Dynamic system

Design

Integration
Scalability

Low latency

Interaction vs. Computation



- **Communication**
 - Experience
 - Influence
 - Constant feed-back
- **Perception**
 - Robust, non-exact
 - Multi-modal
- **Dynamic**
 - Causality
 - Concurrency
 - Metric
- **Algorithms**
 - Computable functions
 - Turing machine
 - Input- \rightarrow (p)- \rightarrow output
- **Calculations**
 - Exact
 - Repeatable
- **Time abstracted**
 - Synchrony
 - Sequentiality
 - Lock-step

Sales contract
vs. marriage contract
(Wegner)

Outline



- Introduction
- **Challenges**
 - Computer Music
 - Computer Graphics and Vision
 - Solutions?
- Case study
 - CAMSHIFT tracking
 - Stevi
- More examples
 - MuSA.RT
 - ESP
 - Improv
 - Soccer game
- SAI
 - Architectural style
 - Tools
- Summary and Perspectives

Computer Music



- On-line
 - Sound synthesis: DSP, hard real-time (scheduling)
 - Music N, Csound, Max, Pd, etc.
 - Interactive control: events
 - Aura
- Off-line
 - Composition
 - Process-centered: Max/MSP
 - Data-centered: OpenMusic
 - Interaction: GUIs (patch metaphor)
- Music analysis: DSP, ML, etc.
 - Same techniques on-line and off-line
 - Time vs. structure

Computer Graphics and Vision



- CG: Synthesis, CV: Analysis
- Off-line
 - Data-centered: modeling, animation, LISP
 - Process-centered: procedural rendering, DirectShow
 - Interaction: GUI
 - Maya, Photoshop, Cantata
- On-line
 - Historical context
 - Off-line analysis in “real-time” (time vs. structure)
 - Intuitive interfaces for animation
- Synthesis and analysis are very different problems

Challenges



- Difficulty of combining signal processing and event processing (Dannenberg, 2002)
- Divide between processing and representation paradigms (Puckette, 2004)
- Reconcile DSP (time), control and data-centered representations?
- Implicit => explicit representation of time

Time and Computation



- Time is central and necessary concept in:
 - Perception
 - Expression
 - Performance (music, sports, games, etc.)
 - Learning
 - Robustness
 - Intelligence (human-style)
- Dynamic systems evolve in time
 - Differentiate instance (individual) from class
 - Termination is irrelevant (and inevitable...)
 - Nothing is immutable
 - “(Useful) Mathematics of complex systems” (Brooks) ?

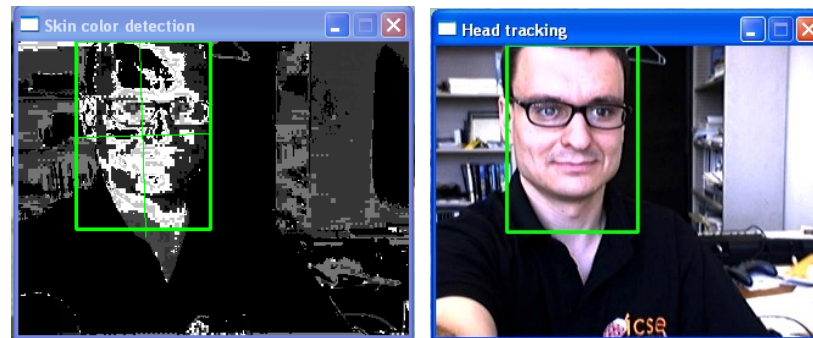
Outline



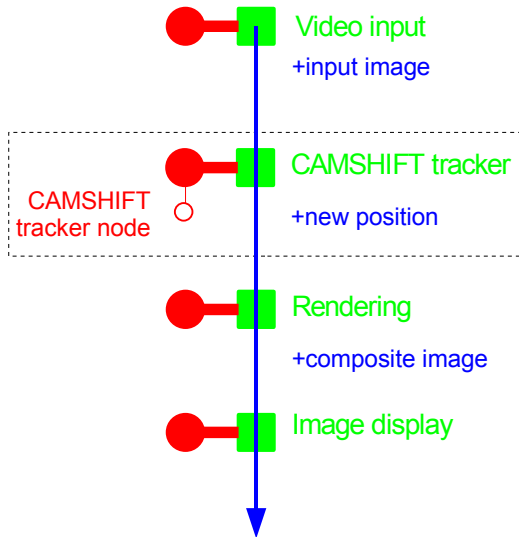
- Introduction
- Challenges
- **Case study**
 - CAMSHIFT tracking
 - Stevi
- More examples
- SAI
- Summary and Perspectives

CAMSHIFT Tracking

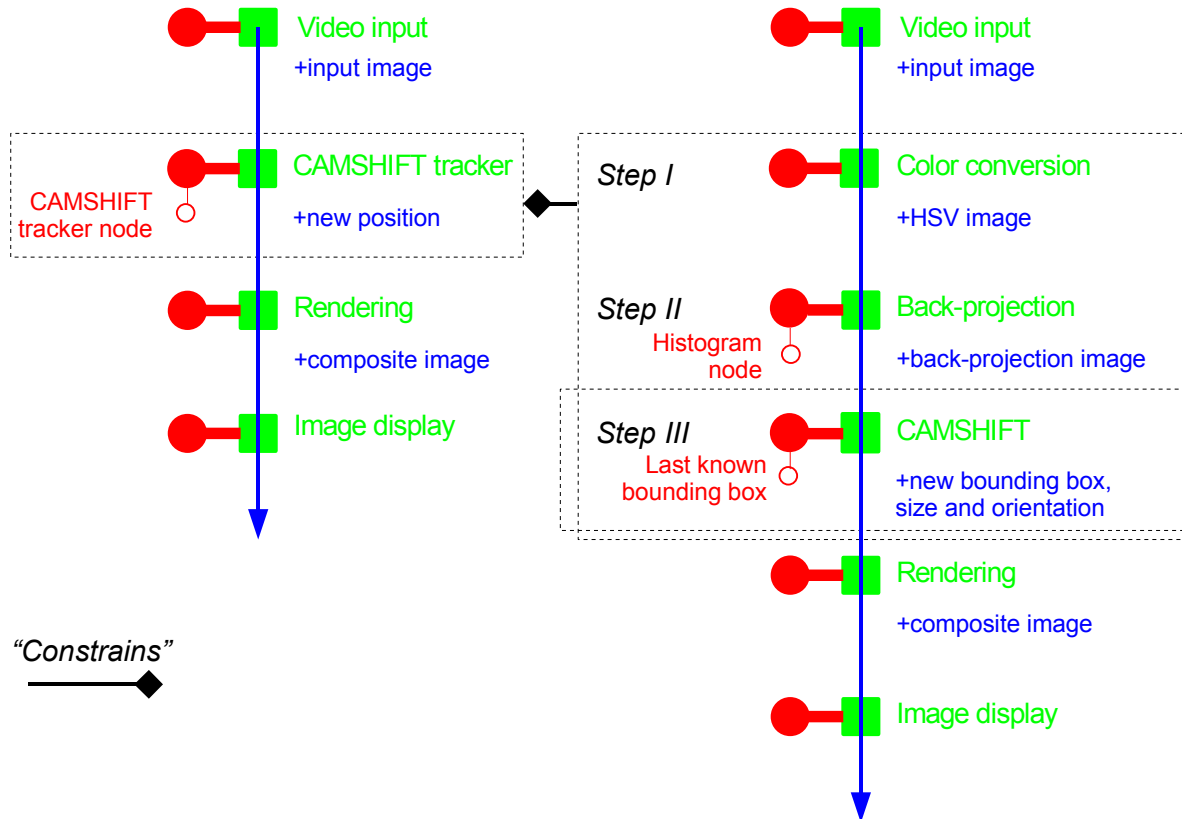
- Tracking algorithm: CAMSHIFT
 - Continuous Adaptive Mean SHIFT (Bradski, 1998)
 - Mean shift: **iteratively** find the mode in a probability density distribution (Comaniciu & Meer, 1997)
- Perceptual User Interface
 - Real-time video processing



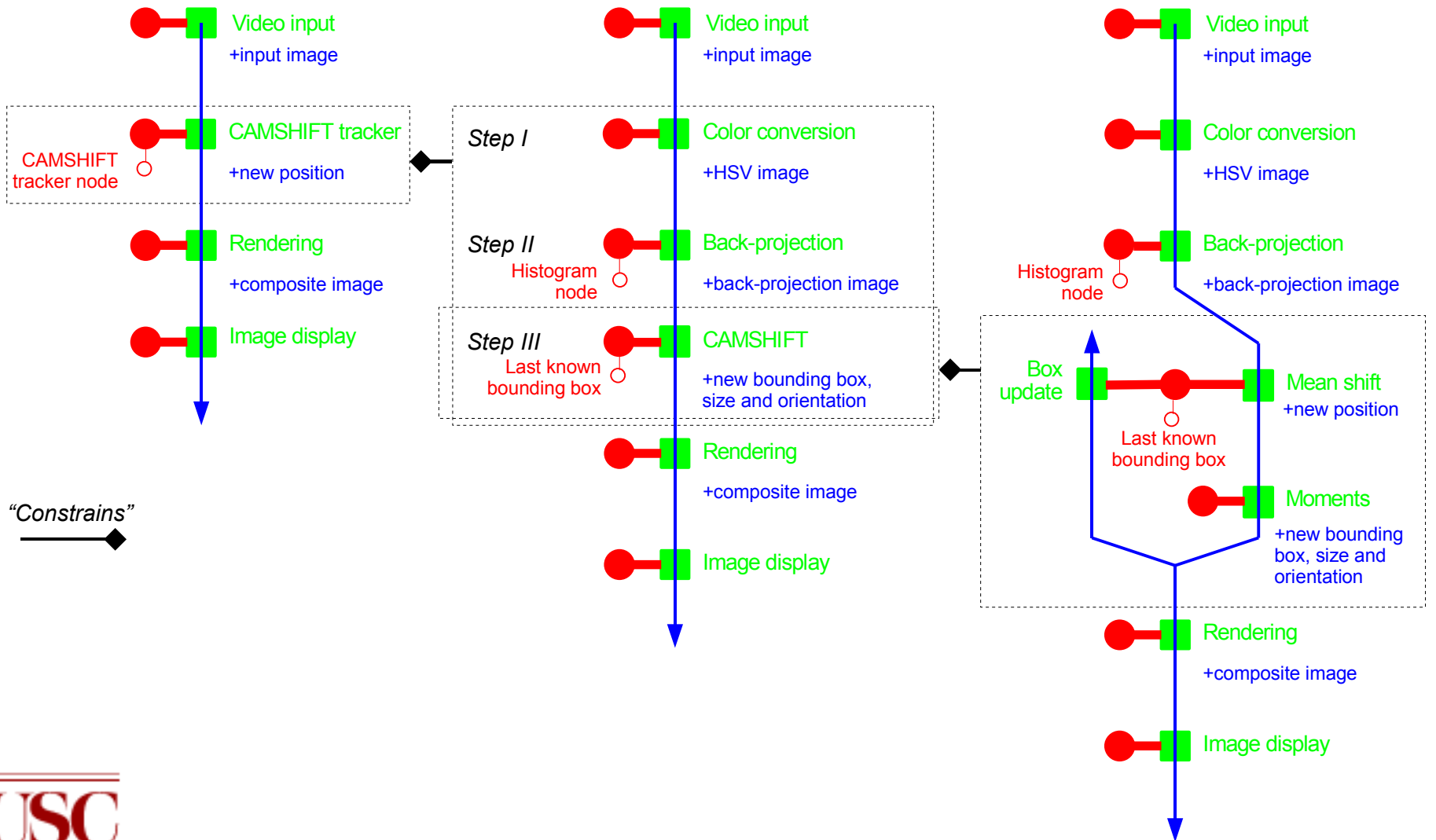
Example System



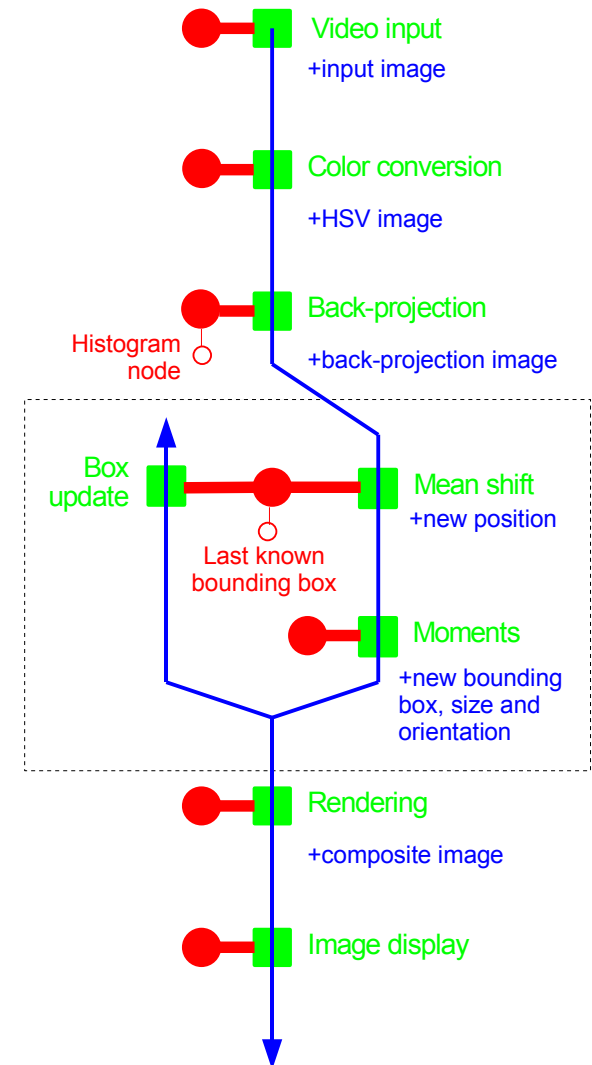
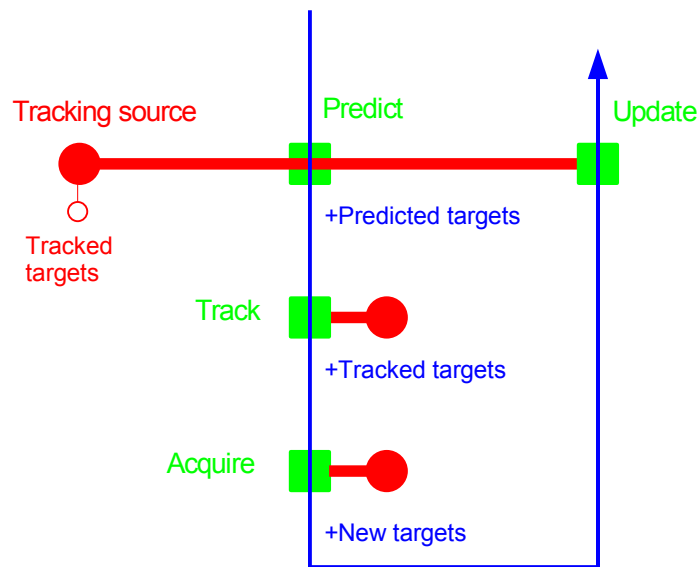
Refinement 1



Refinement 2



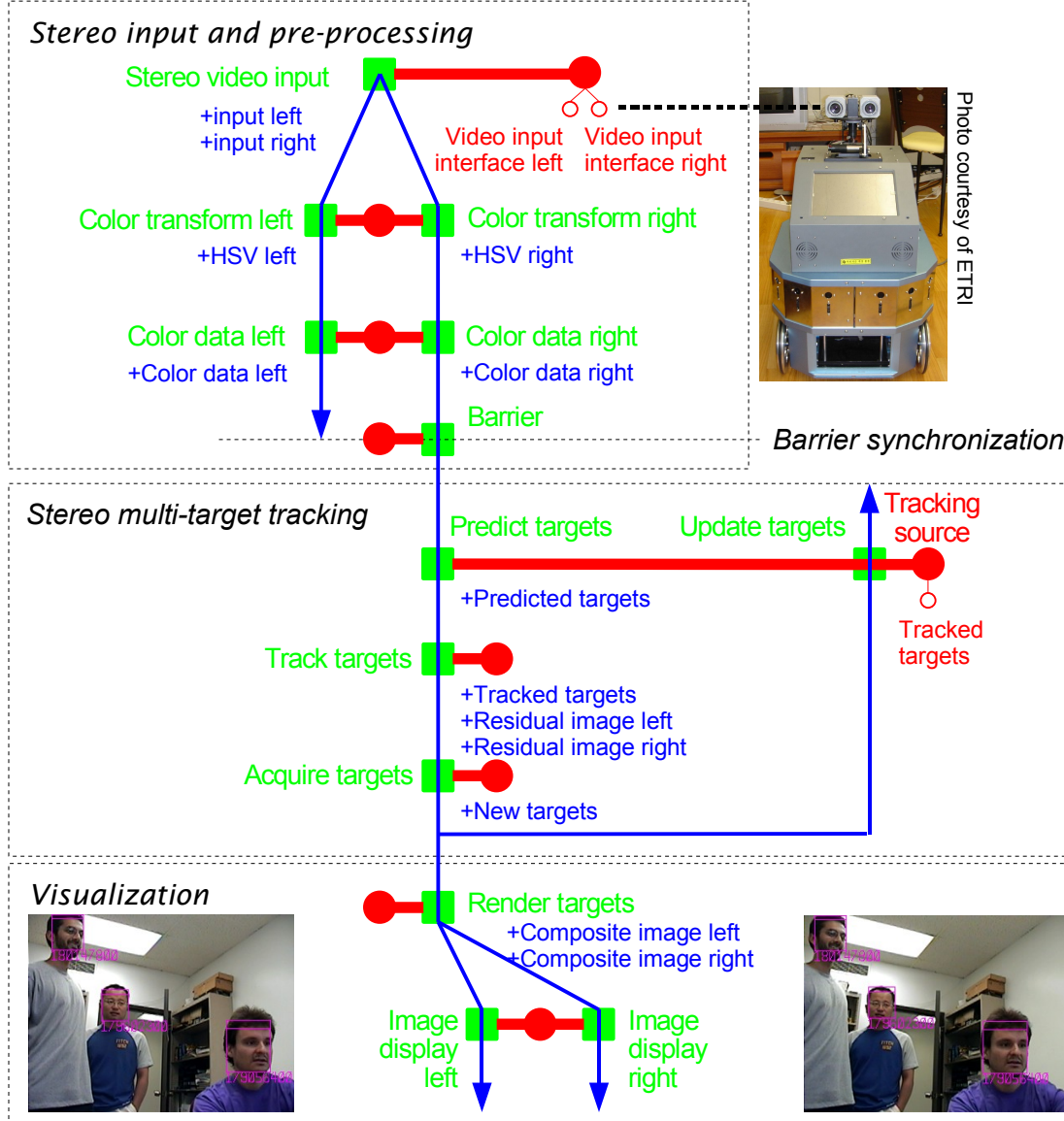
General Tracking Pattern



Vision for Robot: Stevi 1



Co-PI: Gérard Medioni



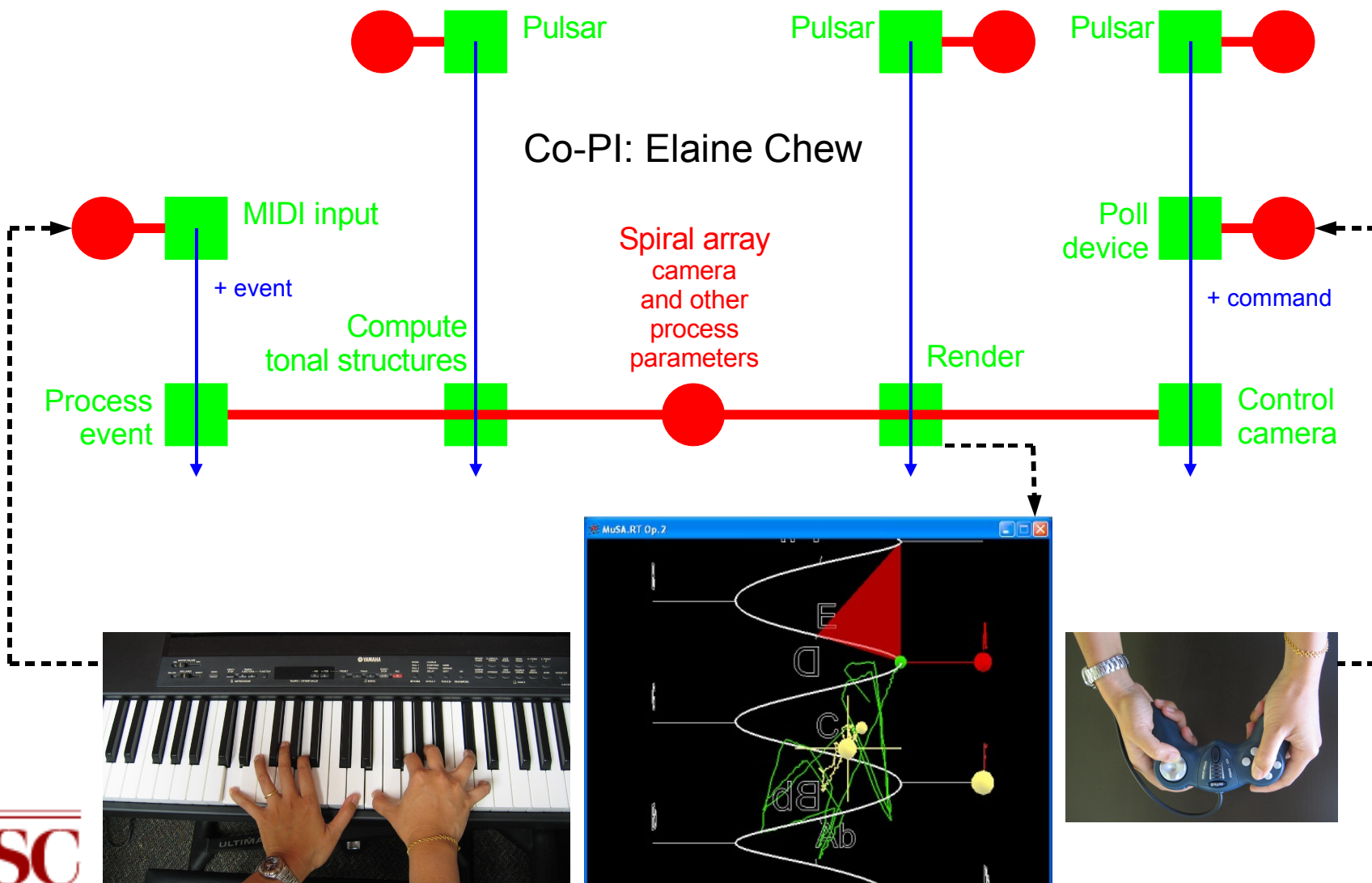
Outline



- Introduction
- Challenges
- Case study
- **More examples**
 - MuSA.RT
 - ESP
 - Improv
 - Soccer game
- SAI
- Summary and Perspectives

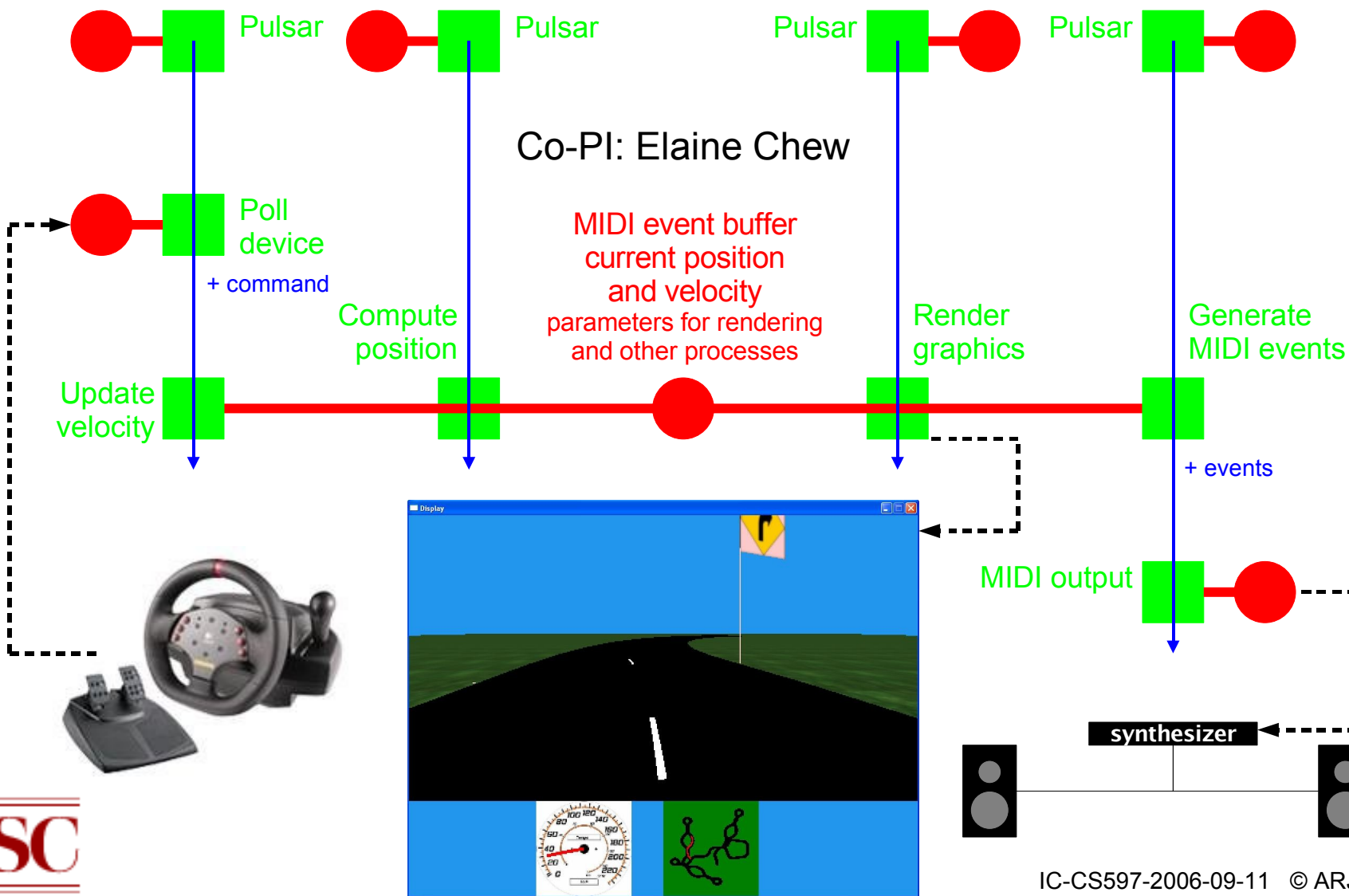
MuSA.RT

Music on the Spiral Array . Real Time

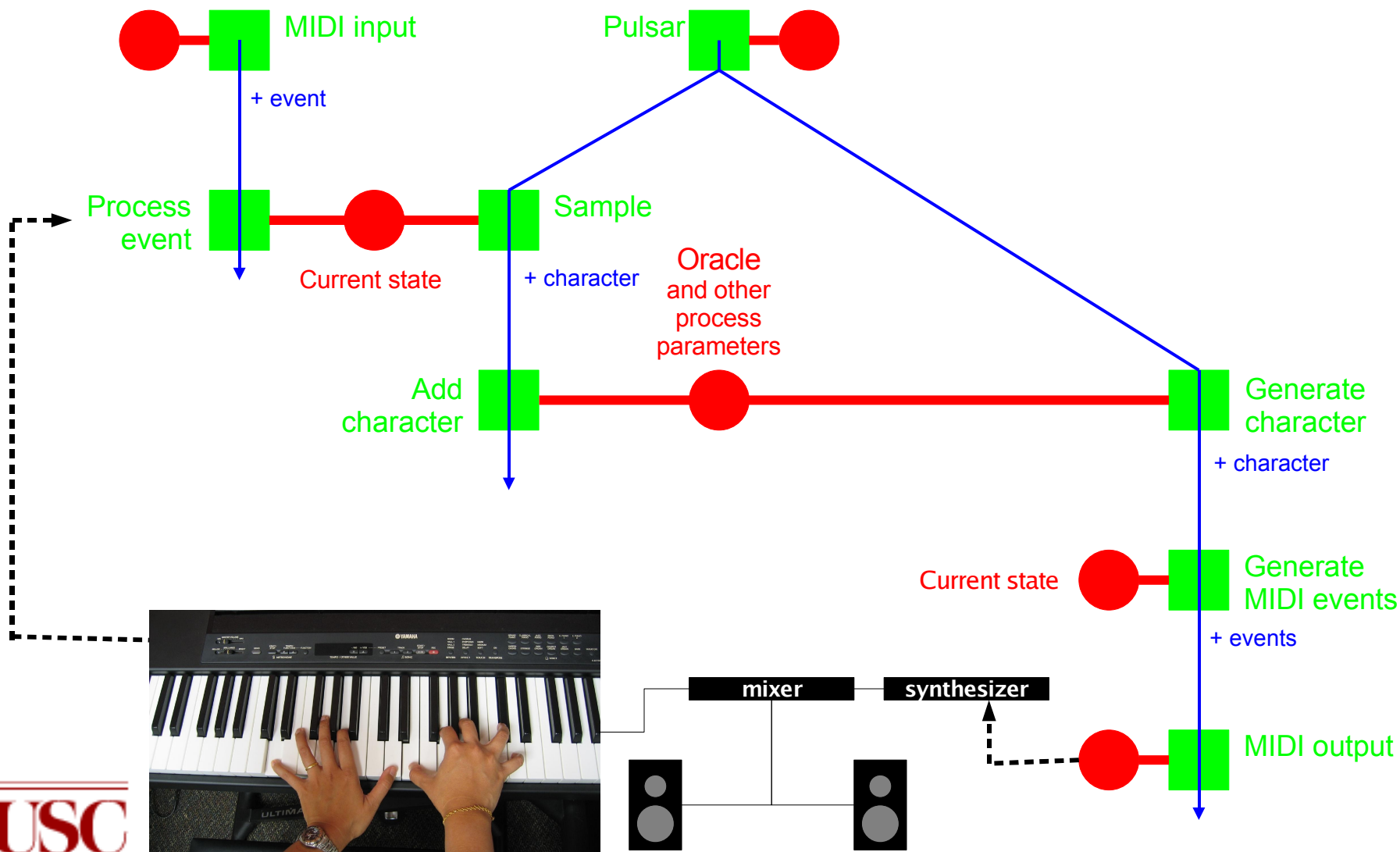


ESP

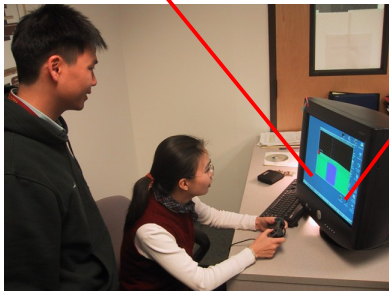
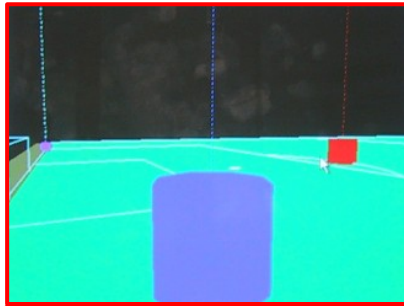
Expression Synthesis Project



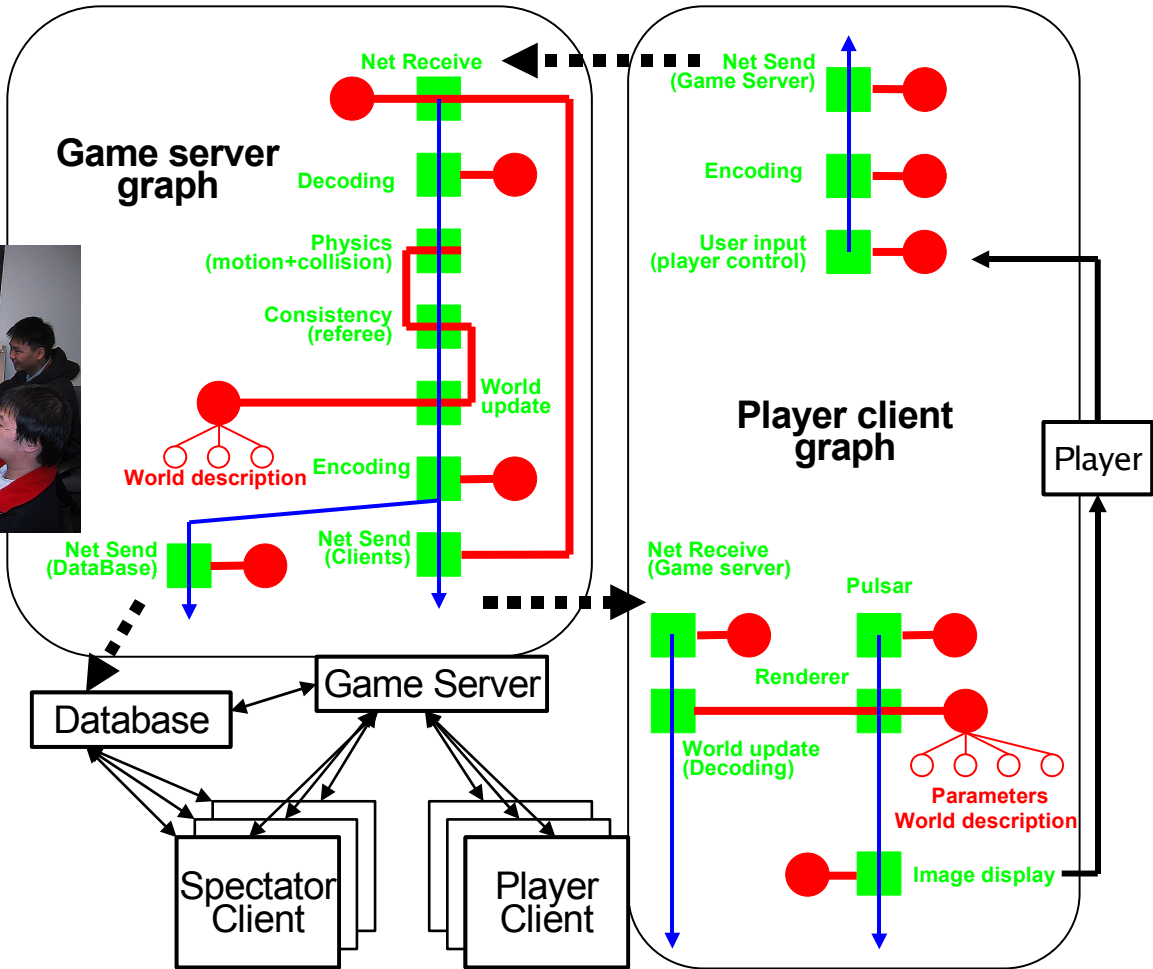
Improv



Distributed Game Project



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



Outline



- Introduction
- Challenges
- Case study
- More examples
- **SAI**
 - Architectural style
 - Tools
- Summary and Perspectives

SAI Principles: Time



- **Metric**
 - Time stamps for all data
 - Computation takes time
- **Lifespan of data**
 - Volatile vs. persistent
- **Concurrent processing**
 - Asynchronous
- **Architectural style [ICSE2004]**
 - High level abstractions
 - Hybrid model or more general model?

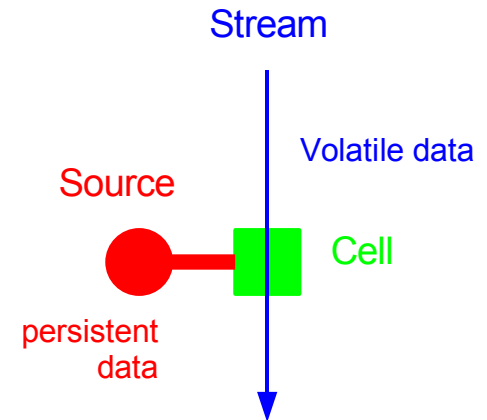
Software Architecture



- **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

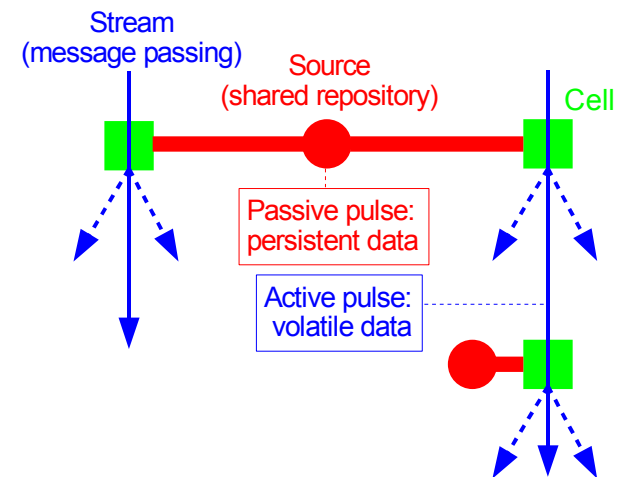
Architectural Primitives

- 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



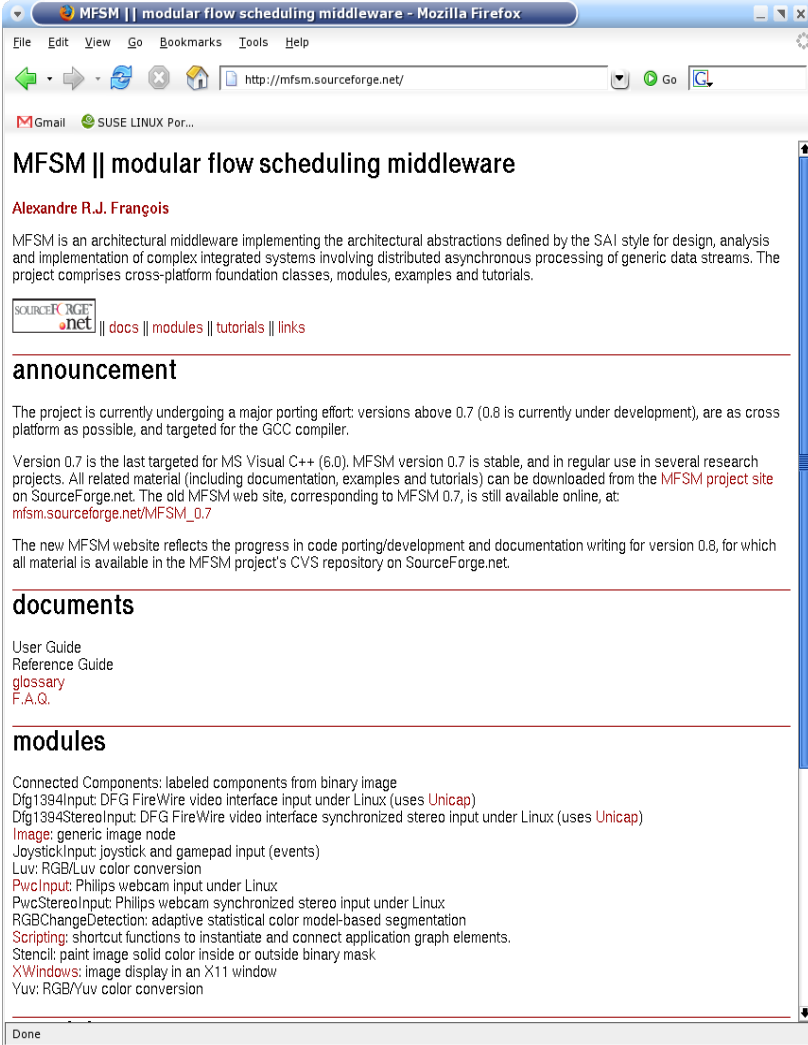
Constraints and Semantics

- 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 cell
 - Any number of cells to a source
 - Concurrent access
- Processing model
 - Active pulse triggers cell process
 - Process may add to active pulse
 - Process may modify passive pulse



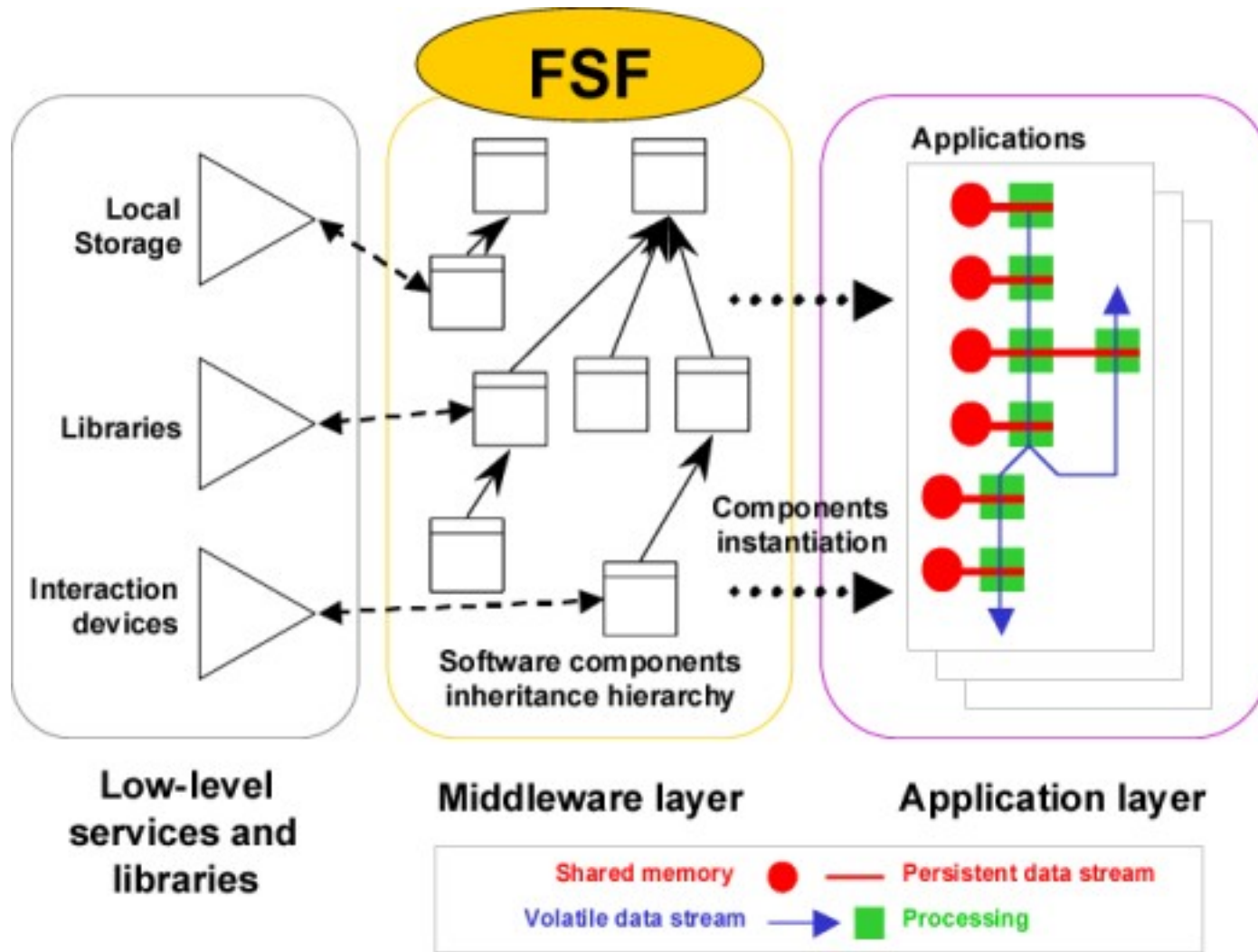
Architectural Middleware

- 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

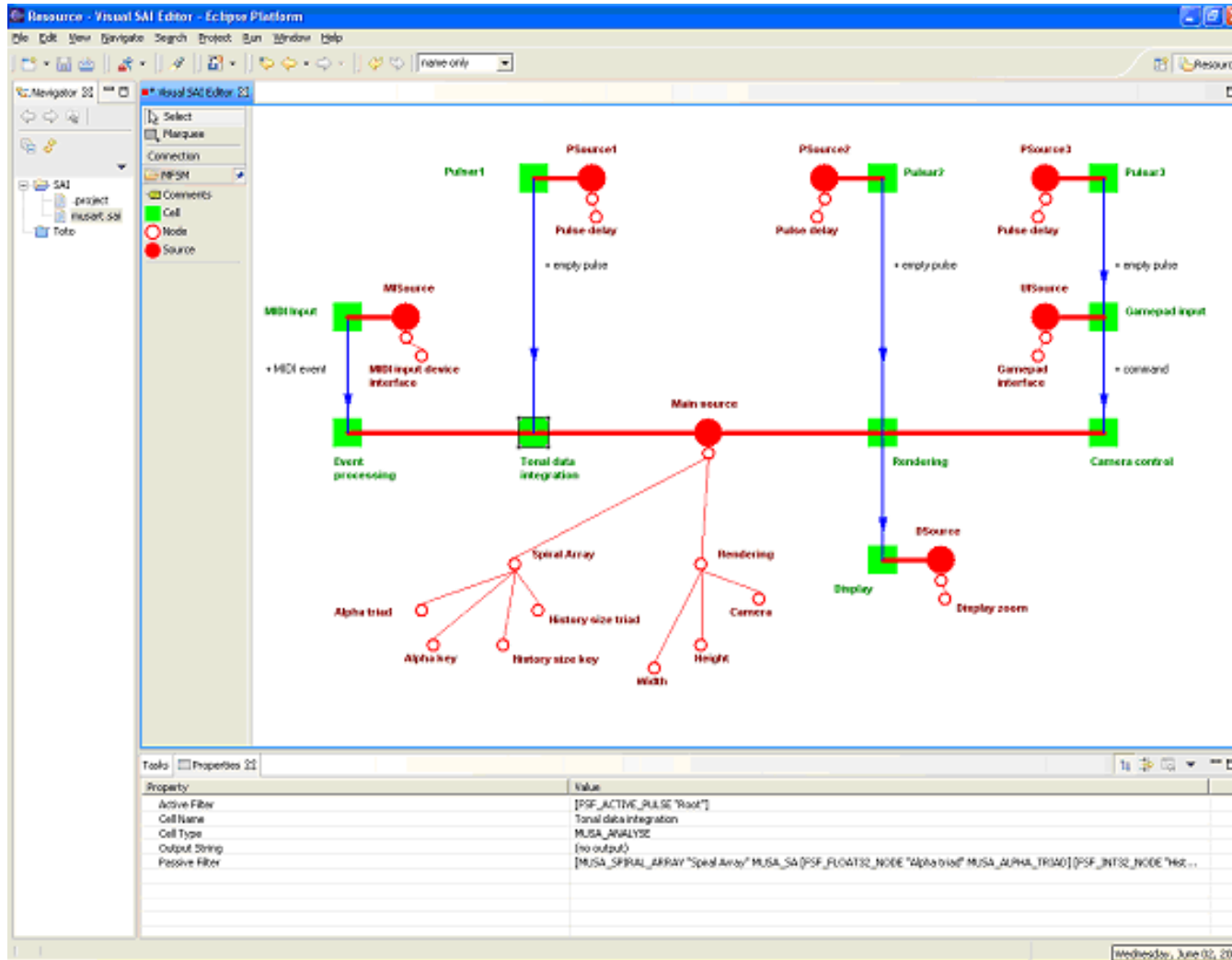


The screenshot shows a Mozilla Firefox browser window displaying the MFSM project page on SourceForge. The page title is "MFSM || modular flow scheduling middleware". The author is listed as Alexandre R.J. François. The page content includes a description of MFSM as an architectural middleware implementing the SAI style, a navigation menu with links to docs, modules, tutorials, and links, and three main sections: announcement, documents, and modules. The announcement section states that the project is undergoing a major porting effort to GCC, with version 0.7 being stable and version 0.8 under development. 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 Organization



VisualSAI



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

Outline

- Introduction
- Challenges
- Case study
- More examples
- SAI
- **Summary and Perspectives**

Summary



- Interaction => Dynamic & Integrated
 - Goes beyond traditional definition of computation
- SAI: Software Architecture for Immersipresence
 - Design and analysis of complex software systems
 - Architectural style, patterns
 - Tools: MFSM architectural middleware, Visual SAI
- Applications:
 - Interactive music systems, Computer vision and graphics systems, Distributed Interactive games, etc.
- For more information:
 - <http://iris.usc.edu/~afrancoi/sai>
 - <http://mfsm.sourceforge.net>

Perspectives



- Design Language for Interactive Systems
 - Communication among all parties involved in creation, realization, analysis and maintenance
 - Computation model for interaction (**time**)
 - *Ontology* of patterns
- Tools
 - Design: e.g. VisualSAI
 - Analysis: safety, liveness
 - Middleware: cross-platform, hardware co-design?
 - Open-source modules
- More fun projects!!
 - Games, simulations
 - Artificial intelligence

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.

Interaction and Computation

Alexandre R.J. François

Computer Science Department

alexandre.francois@usc.edu

www.alexandrefrancois.org

© ARJF 2006



USC Viterbi
School of Engineering