

# Parallel Asynchronous Processing

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

© ARJF 2005



USC **Viterbi**  
School of Engineering

# A-Synchronicity



- The universe is not sequential...
  - ...Your computer is not either!
  - Sequential programming is an artificial “simplification”
- The universe is massively parallel, and processes are asynchronous
  - Physical principles
    - Causality
    - Latency
  - Psychophysical principles
    - Simultaneity

# A Story About a Game

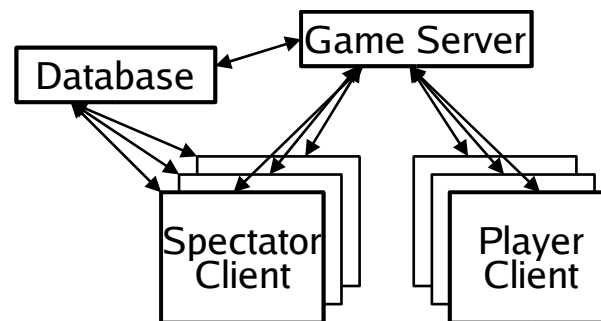


- CS599: “Integrated Media Systems” course
  - Taught at USC with R. Zimmermann
  - In charge of experimental project: a game
- 25 students with
  - No experience in game development
  - No experience in large team project
- Short deadline (2 months)
- High stakes (fail vs. pass...)

# Distributed Soccer Game



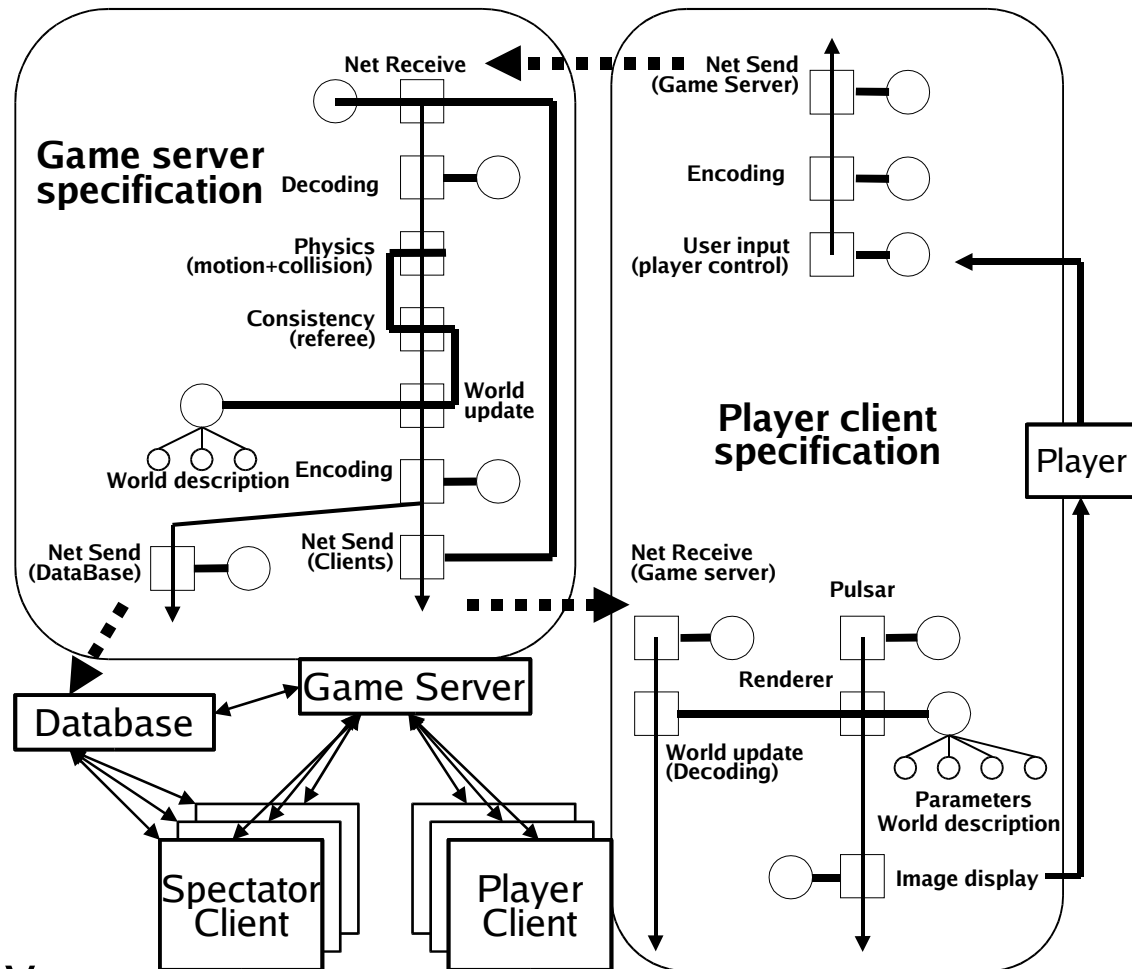
- 3D graphics
- Some physics
- Client-server topology
- Players and spectators
- Database



- Cases?
- Classes?
- Threads??

# System Architecture

- Refined specifications (conceptual to logical)
- Common data structures
- Functionalities
  - Rendering
  - Networking
  - Databases
  - Interaction
  - Physics/gameplay



# Project Management

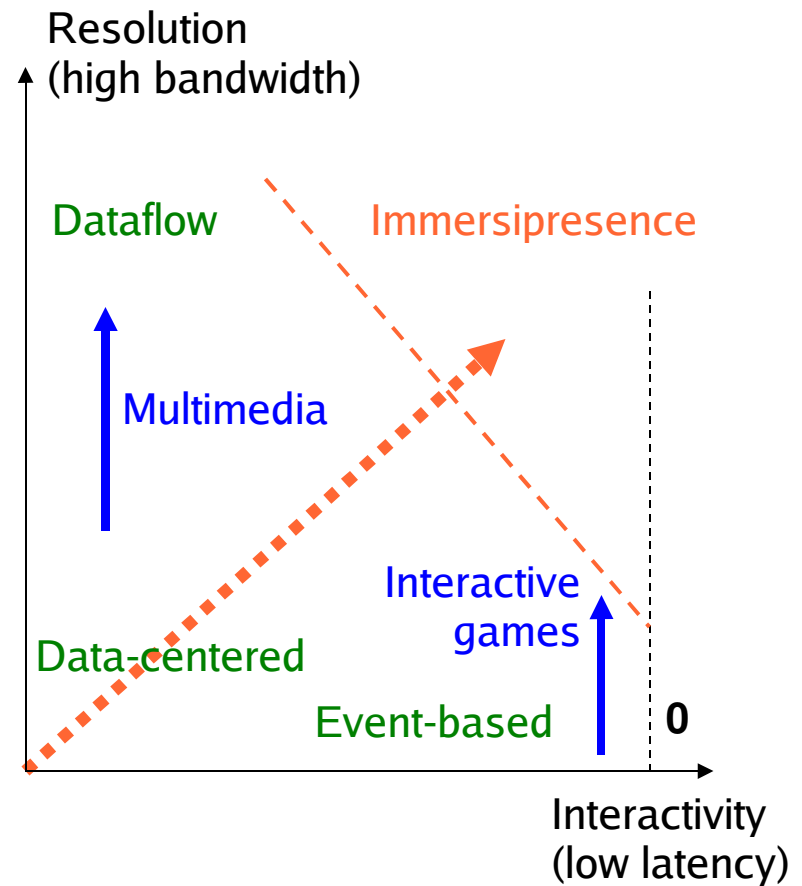


- **Distributed development**
  - Functional teams (1–10 members)
  - Regular cross-testing
  - From initial design to refined working demonstration
- **Complete success...**
  - ...to the surprise of some of the students!



# Related Work?

- Libraries
- Multimedia
- Interactive games
- Software architecture
- Unified approach requires a new model: “Software Architecture for Immersipresence”



# Outline



- Introduction
- SAI: Software Architecture for Immersipresence
- Synchronization patterns
- Examples
- Discussion
- Conclusion

# Outline



- Introduction
- SAI: Software Architecture for Immersipresence
  - Architectural primitives
  - Constraints and semantics
  - Architectural middleware
- Synchronization patterns
- Examples
- Conclusions

# Software Architecture



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

# 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

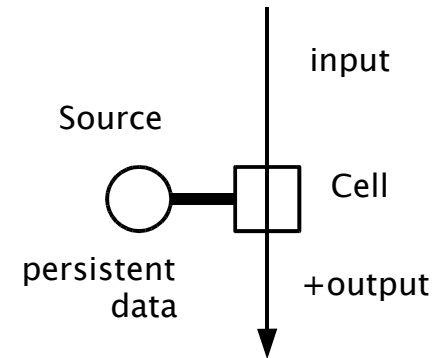


- Interoperability
  - Combine research from different fields/teams
- Efficiency
  - On-line, real-time, low latency
- Scalability
  - Performance evaluation and prediction
- General model for ***distributed asynchronous parallel processing of data streams***

# 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

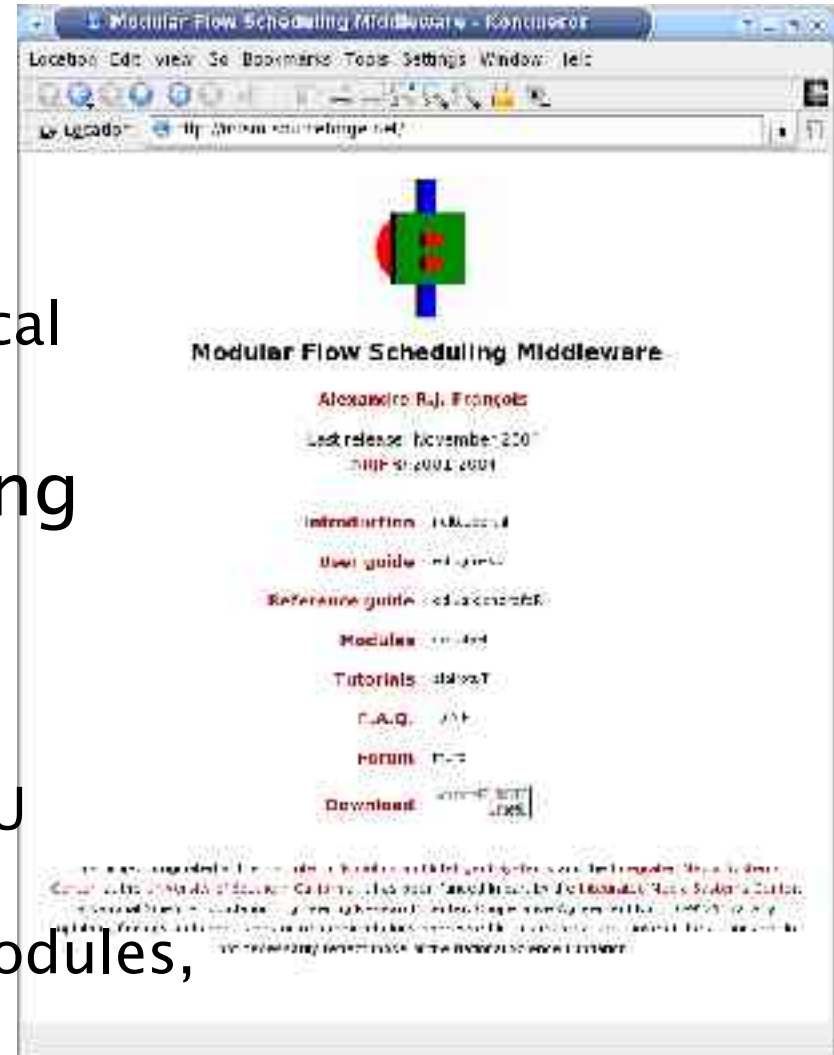


- Cell-cell: volatile data samples (stream)
  - At most one upstream cell
  - Any number of downstream cells
  - Process dependency
- Cell-source: persistent data access
  - Exactly one source to a given cell
  - Any number of cells to a given source
  - Concurrent access
- Processing model
  - Active 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](http://mfsm.SourceForge.net)
  - C++, cross-platform (GNU compiler)
  - Base library, functional modules, documentation, tutorials

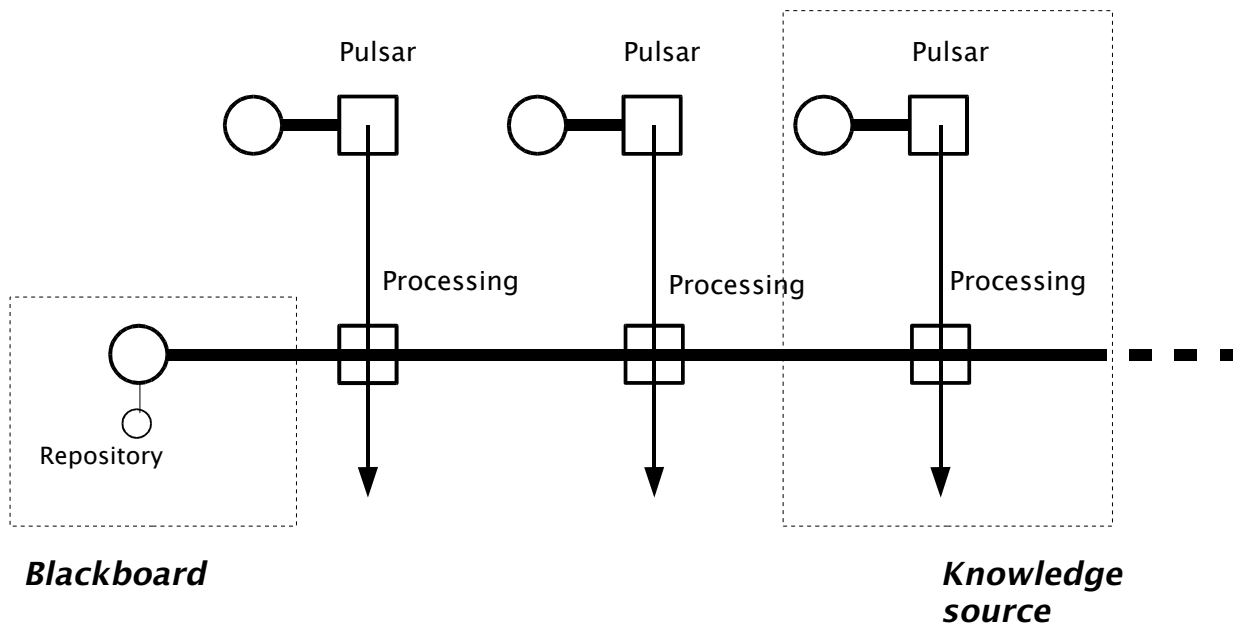


# Outline

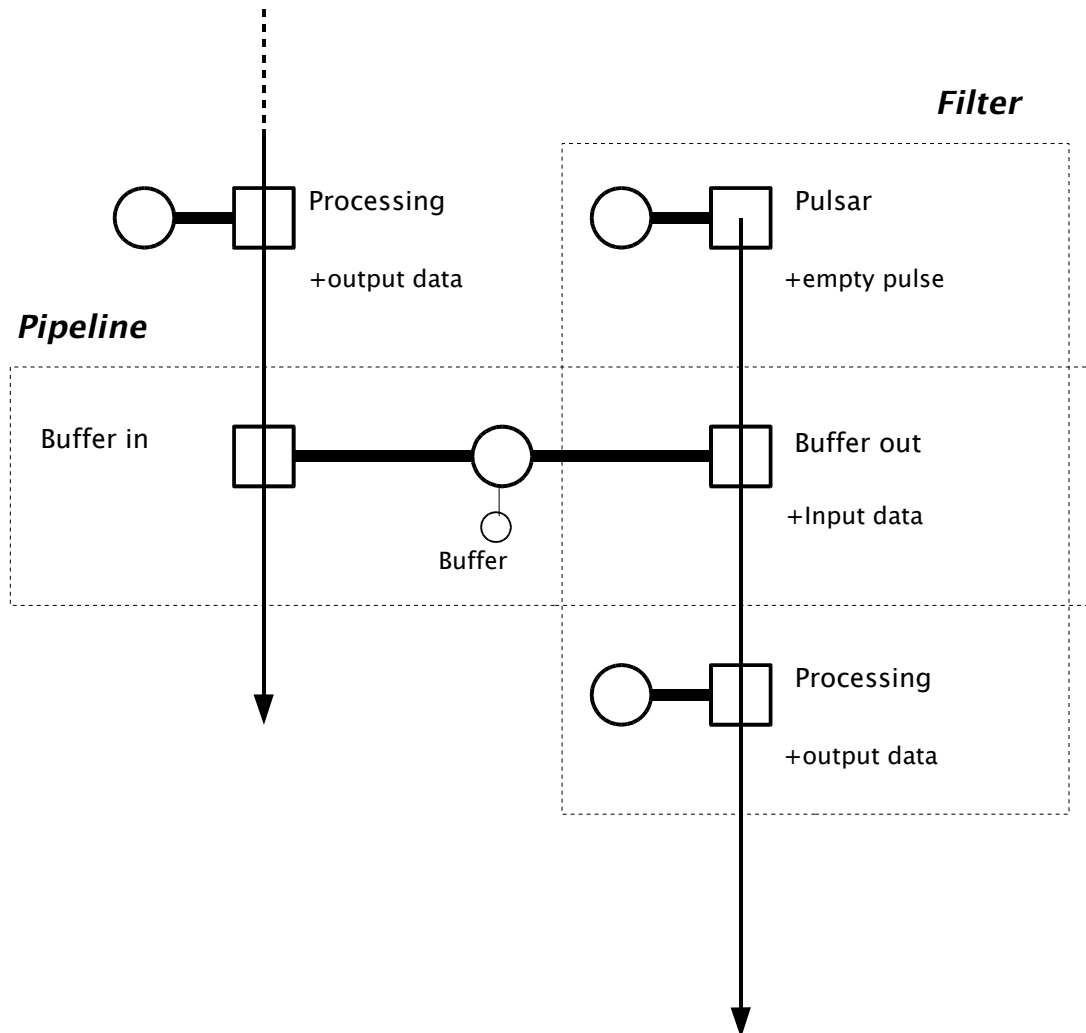


- Introduction
- SAI: Software Architecture for Immersipresence
- Synchronization patterns
  - No synchronization
  - Time precedence
  - Barrier
  - Feed-back/forward
- Examples
- Discussion
- Conclusion

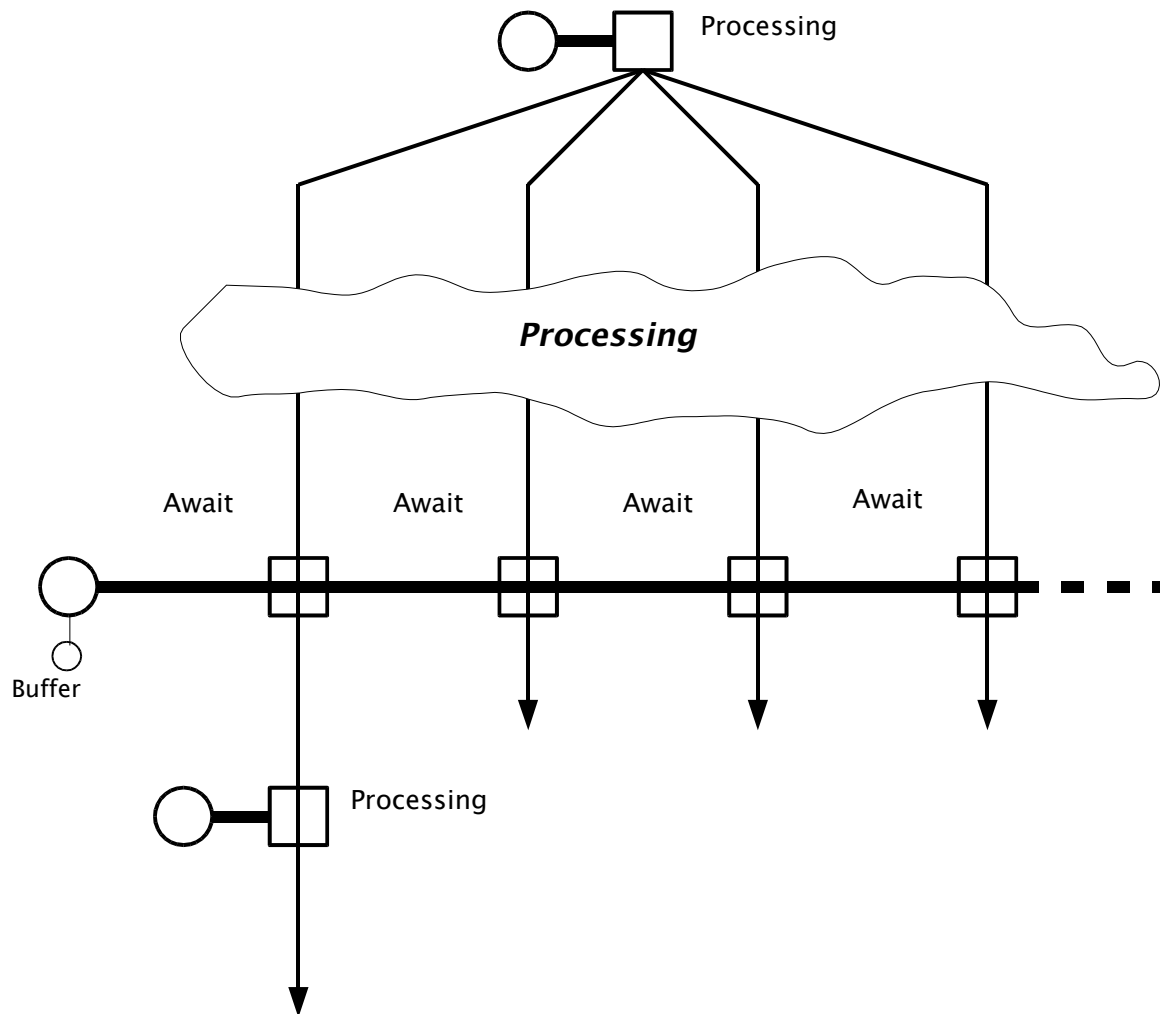
# No Synchronization



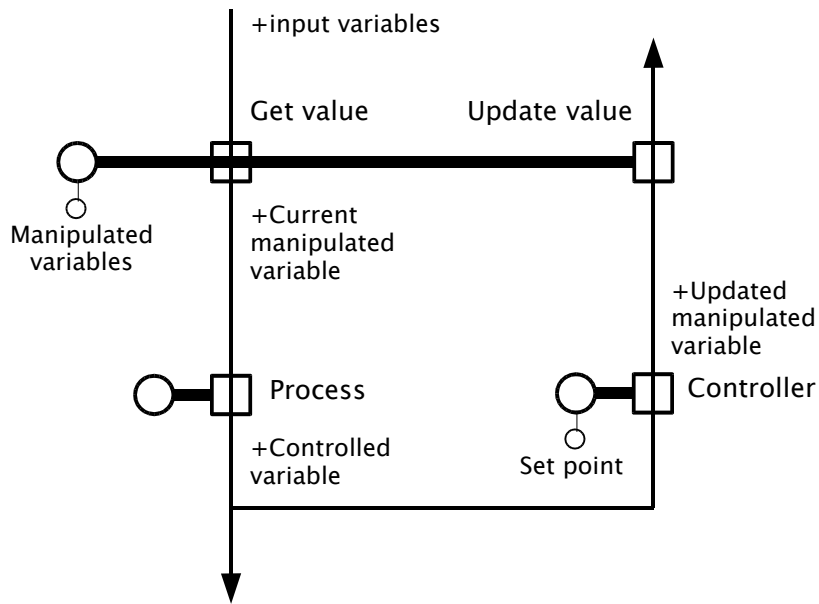
# Time precedence (pipeline)



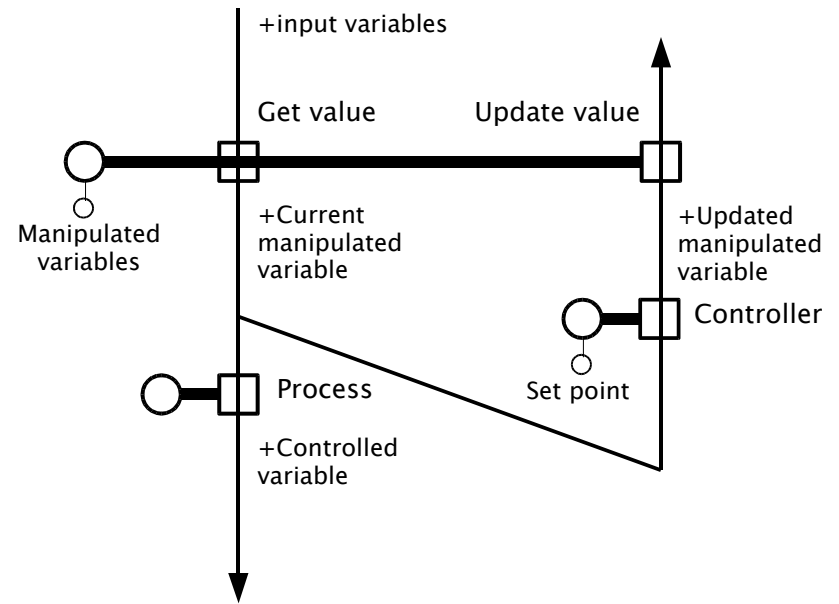
# Barrier Synchronization



# Feed-Back/Forward



*Feed-back*



*Feed-forward*

Race conditions?

# Outline

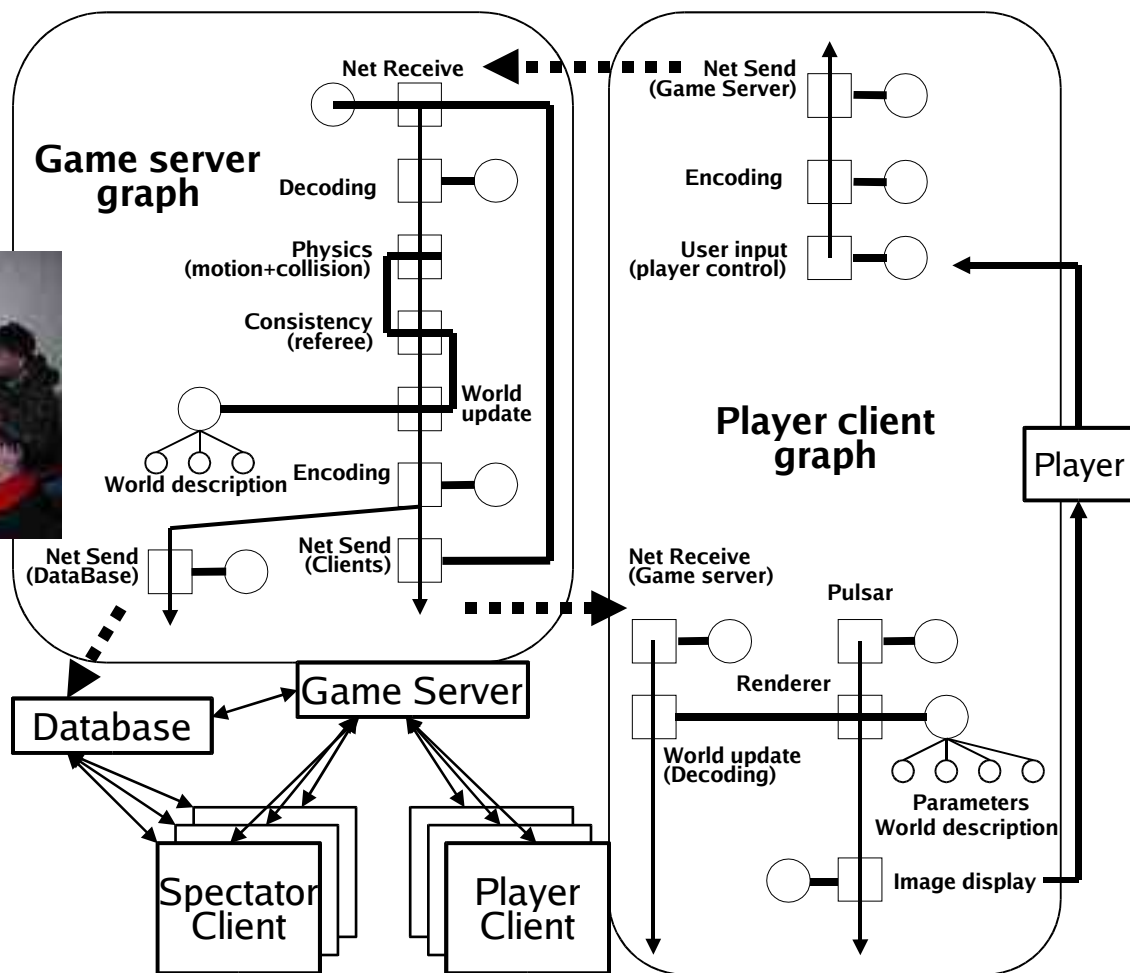


- Introduction
- SAI: Software Architecture for Immersipresence
- **Examples**
  - Distributed Interactive Game
  - MuSA.RT
  - CAMSHIFT: Design by Iterative Refinement
- Synchronization patterns
- Discussion
- Conclusion

# Distributed Game Project

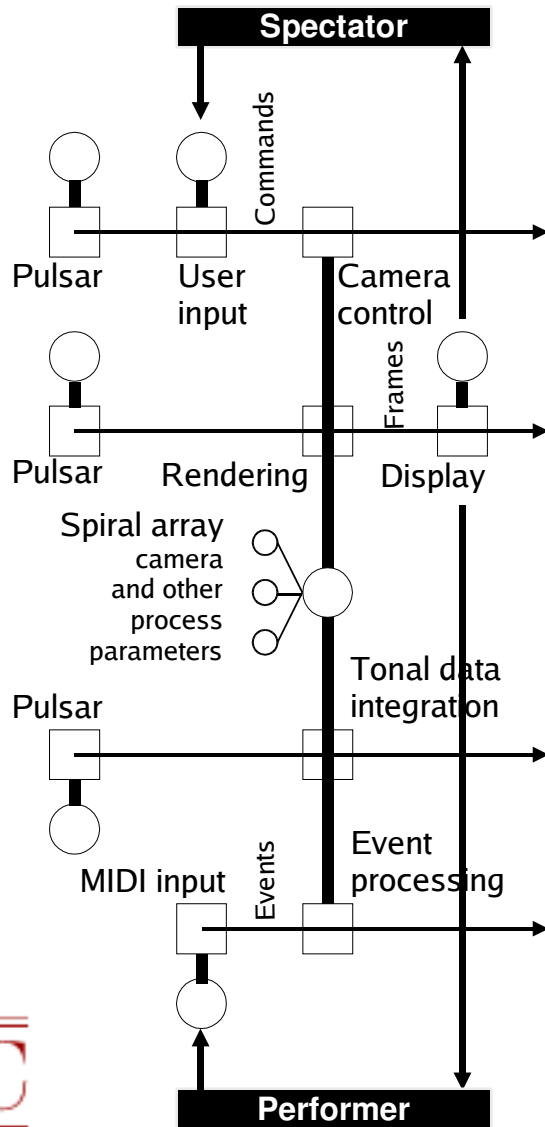


25 students, 2 months  
 Distributed development  
 Real-time multiplayer gaming  
 with database  
 recording/replay



# MuSA.RT

Music on the Spiral Array . Real Time



Co-PI: Elaine Chew



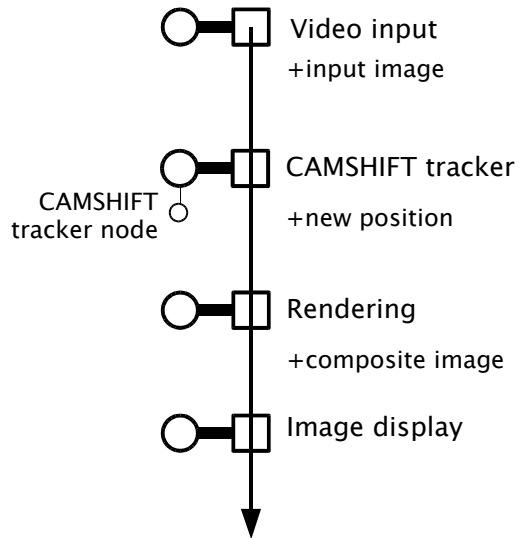
# Head Tracking



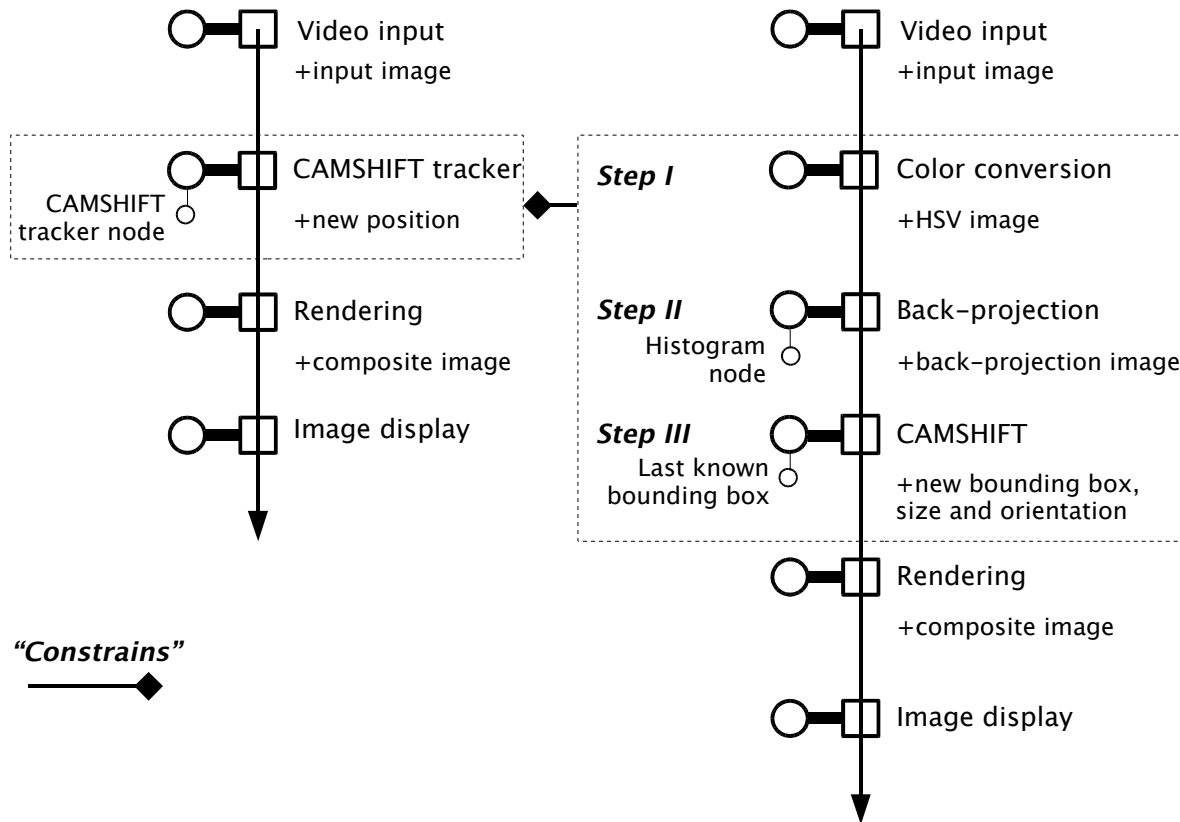
- Perceptual User Interface
  - Real-time video processing
- Tracking algorithm: CAMSHIFT
  - Continuous Adaptive Mean SHIFT (Bradski, 1998)
  - Mean shift: iteratively find the mode in a probability density distribution (Comaniciu & Meer, 1997)



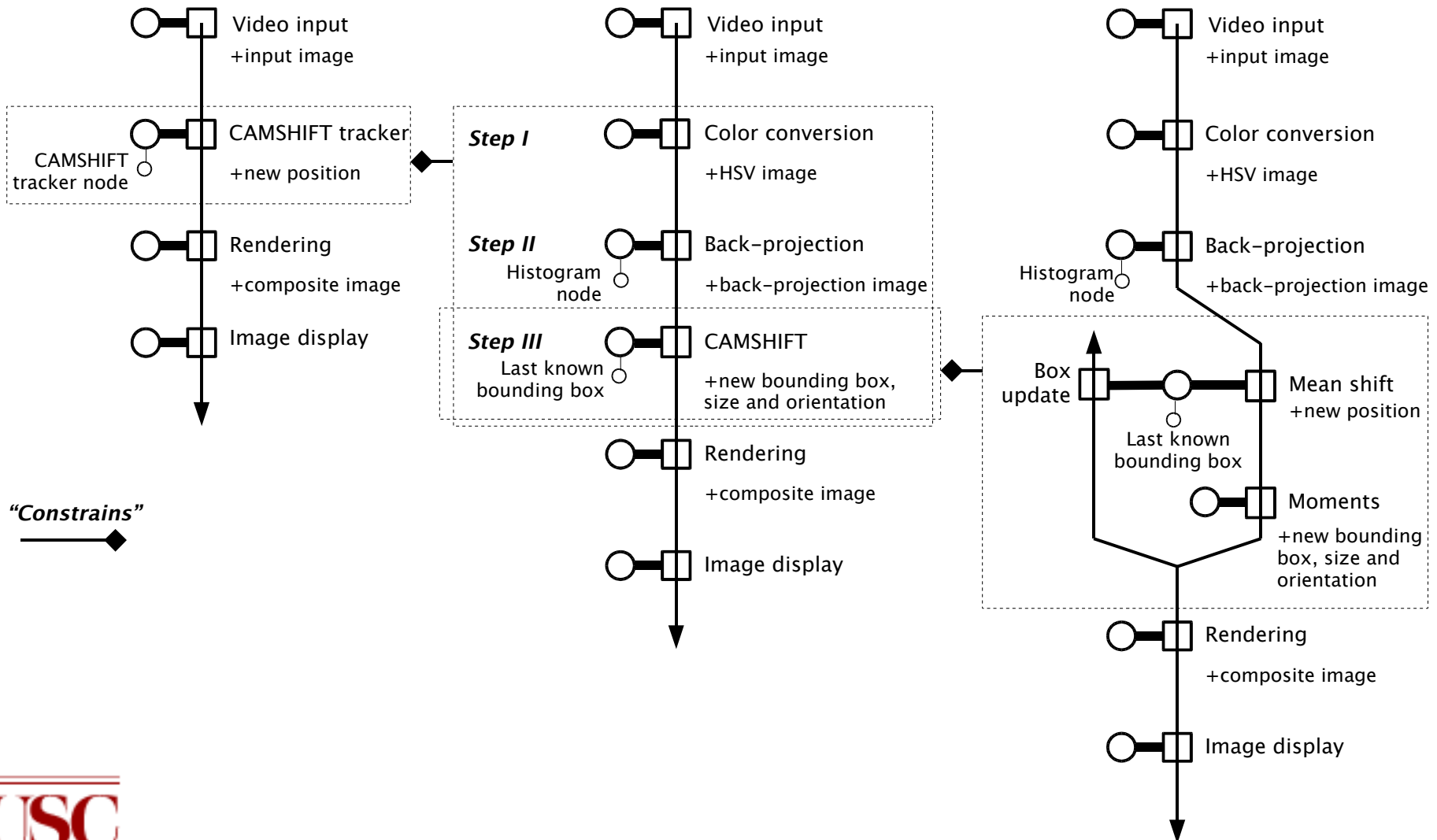
# Example System



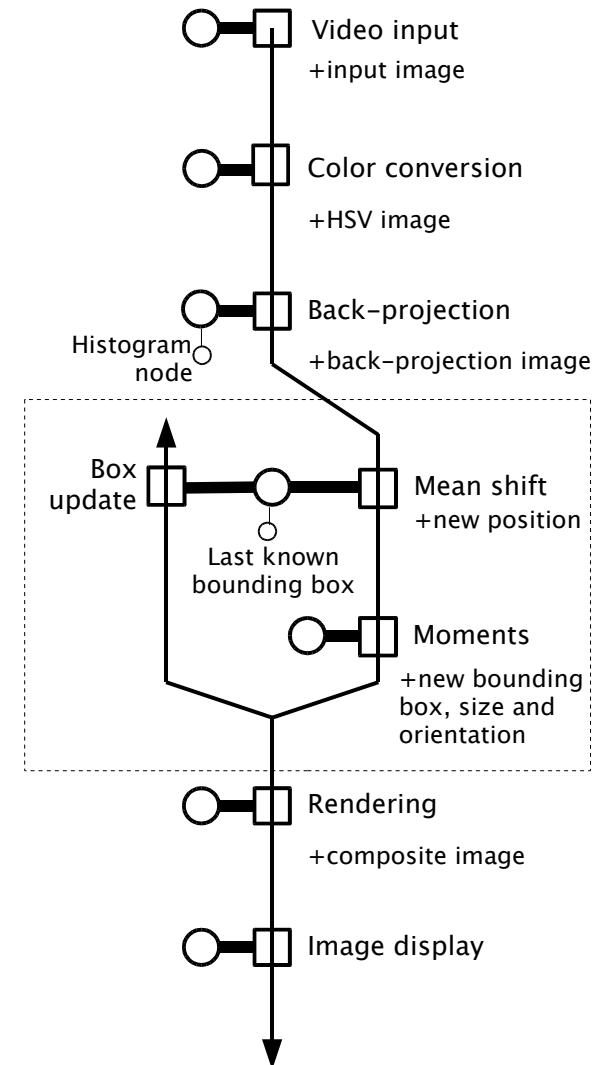
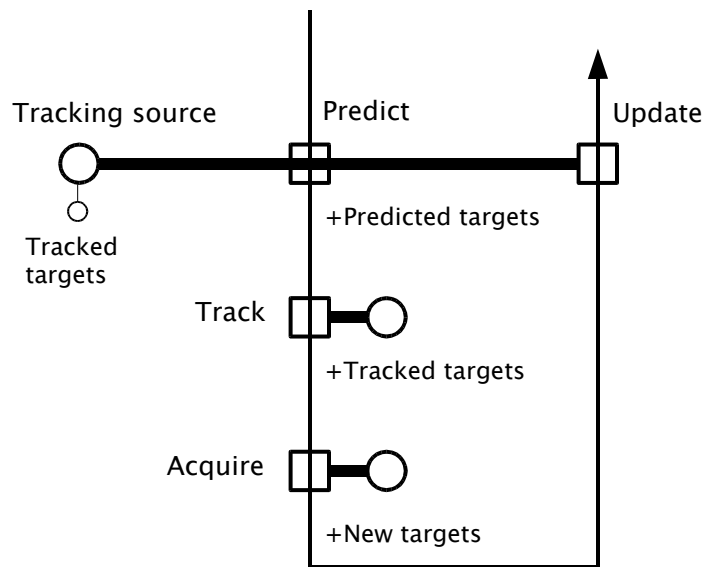
# Refinement 1



# Refinement 2



# General Tracking Pattern



# Outline



- Introduction
- SAI: Software Architecture for Immersipresence
- Synchronization patterns
- Examples
- Discussion
  - Architectural properties
- Conclusion

# Architectural Properties



- Model time explicitly in data and processing
- Compositional model
  - Modularity
  - Parallelism
  - Distribution
  - Scalability
- Allow optimal designs
  - Asynchronous parallelism allows optimal throughput and latency
- Naturally supports concurrent execution and distributed processing

# Architectural Properties



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

# Outline



- Introduction
- SAI: Software Architecture for Immersipresence
- Synchronization patterns
- Examples
- Discussion
- **Conclusion**
  - Summary
  - Research directions

# Summary



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

# Perspectives



- Theory
  - Further formalization and study
  - New computational model
  - ***Ontology*** of computational patterns
  - Influence language and compiler design
- Tools
  - Design: e.g. VisualSAI
  - Analysis: safety, liveness
  - Middleware: cross-platform, hardware co-design?

# 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.

# That's All!

Alexandre R.J. François

Computer Science Department

[alexandre.francois@usc.edu](mailto:alexandre.francois@usc.edu)

<http://iris.usc.edu/~afrancoi>

© ARJF 2005



**USC Viterbi**  
School of Engineering