cs121 - software development
software architecture
sai/mfsm

alexandre r.j. françois

visiting associate professor of computer science

outline

intro: software architecture

interactive systems

software architecture for immersipresence (sai)

modular flow scheduling middleware (mfsm)

software architecture

- design, analysis and implementation of software systems
  - improve the flexibility and comprehensibility of software systems
  - address modularization as a design issue
    (higher level than language level)
- 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

software architecture

people have been doing software architecture since they started writing computer code

from “boxology” to a recognized discipline

relatively young field (early 90's)

architecture for interactive systems

event-based interaction devices

infinite event-loop model

- event queue

game loop

- rendering loop
- graphics take precedence
- time/latency is important: fps

interactive systems

involve humans

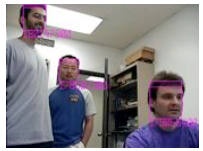
- complex (no matter how complicated...)
- situated
- embodied
- dynamic

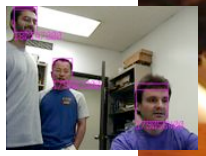
desirable behavior

- robust
- adaptive
- context-dependent

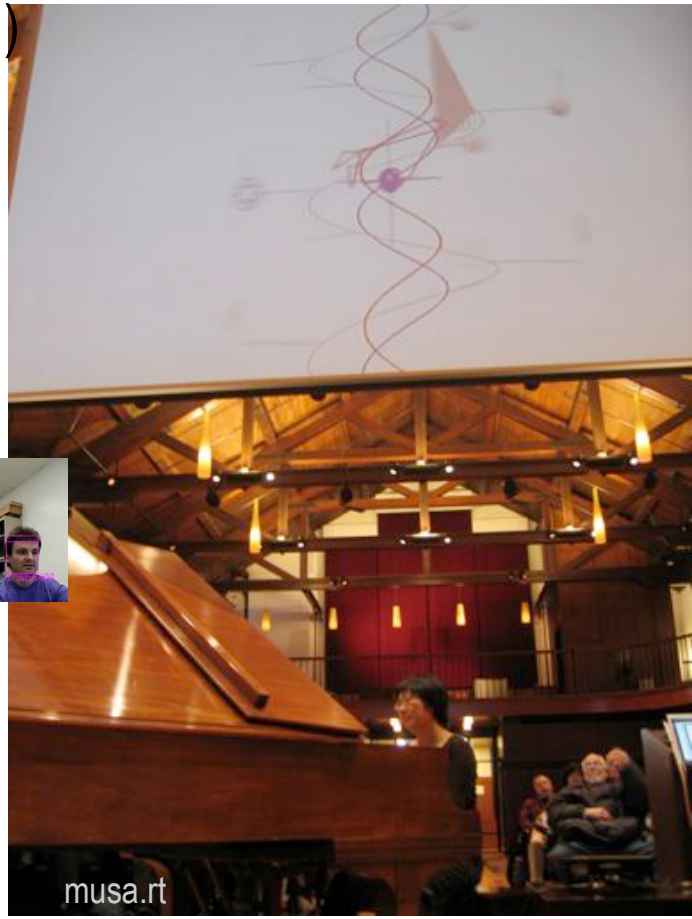

esp


mimi






stevi


musa.rt

challenges

by humans for humans

complex task

beyond straightforward computation

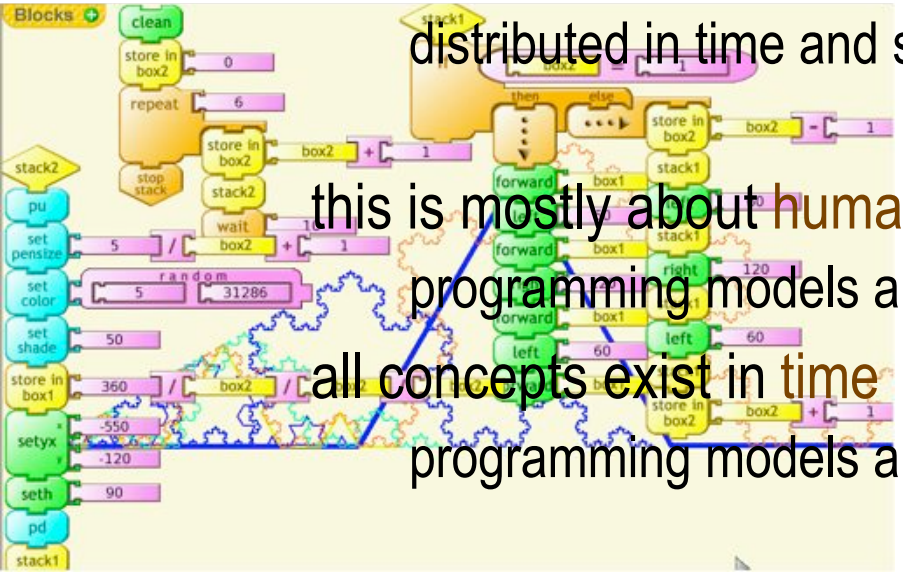
system design

cross-disciplinary teams

system implementation, testing and maintenance

distributed in time and space

turtle art



this is mostly about humans

programming models and languages are machine-centric

all concepts exist in time

programming models and languages abstract time

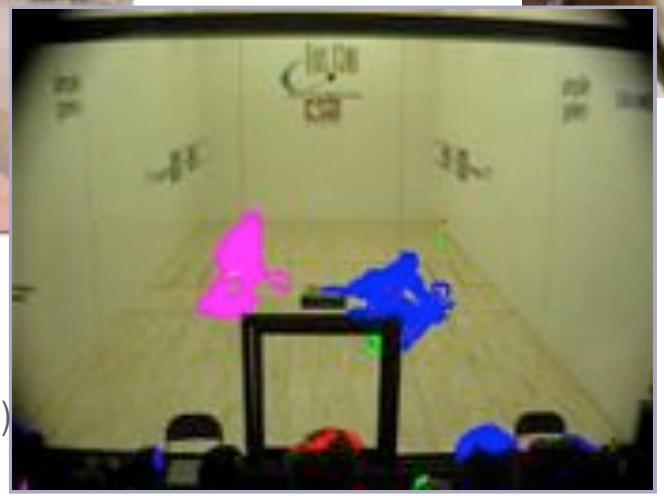
applications -vision-

video painting (1999)

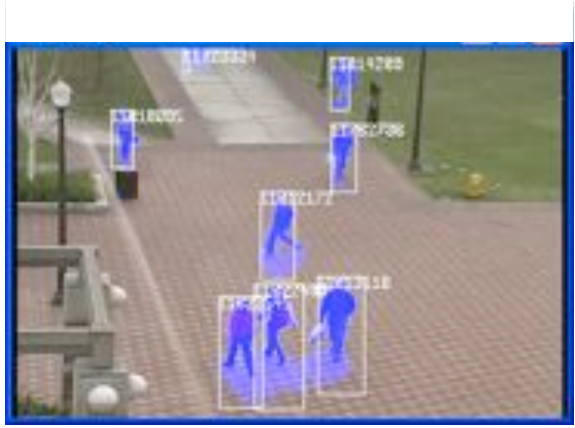
[etcv]

virtual mirror (2000-2002) gra: e.kang

[siggraph2002]



racquetball tracking (2001)



people detection and tracking (2004) funding: arda

[cviu2007]

stevi (2005-2006) pi: g.medioni, funding: etri

applications -music-

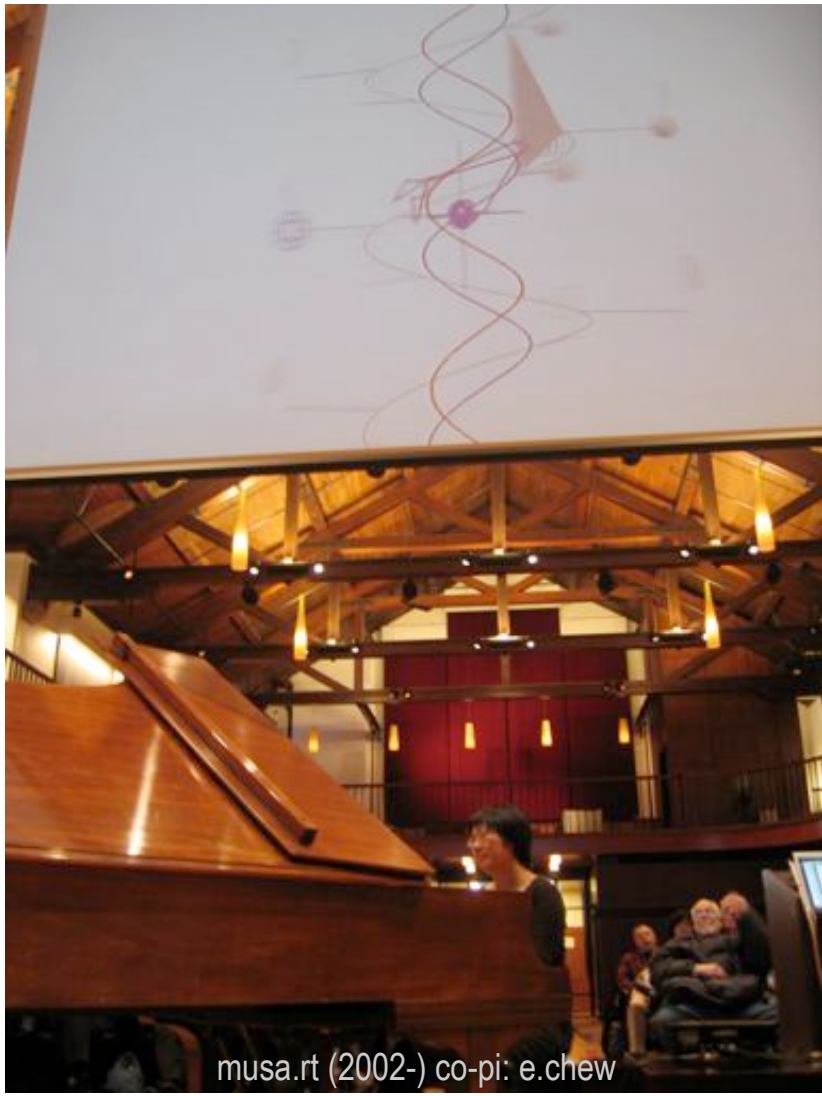
[nime2006]

[cie2005]

[nime2005]

esp (2004-) co-pi: e.chew, gra: j.liu

music computation and cognition laboratory at usc

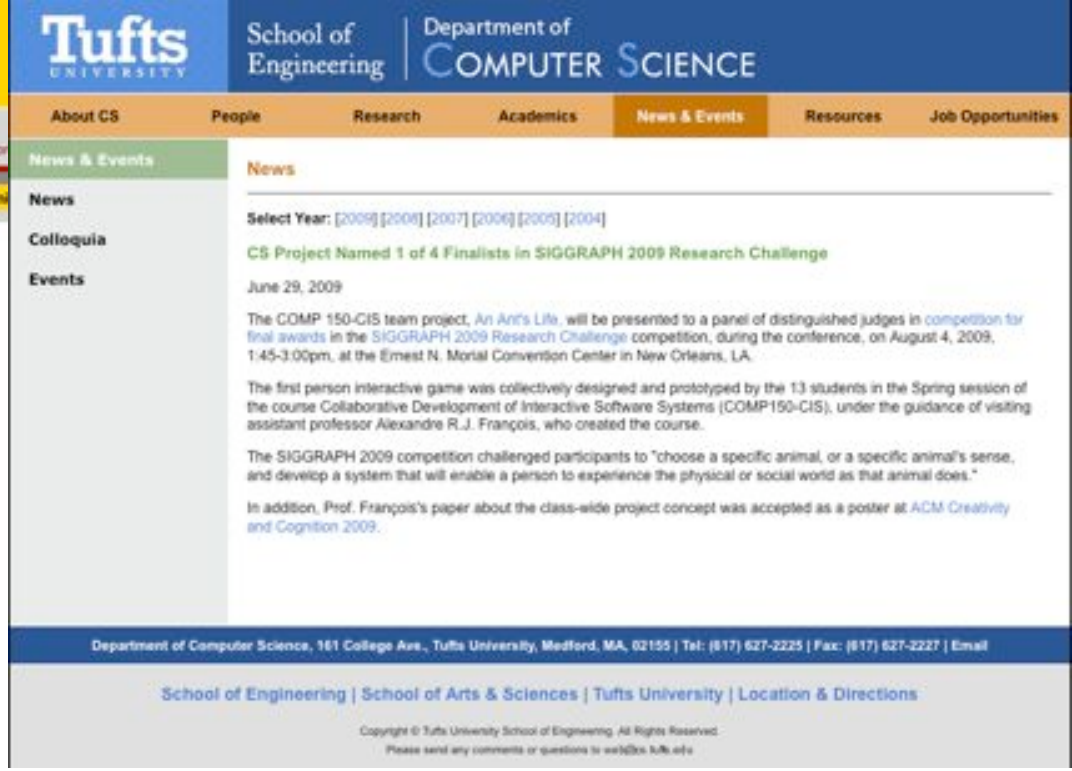


musa.rt (2002-) co-pi: e.chew

mimi (2006-) co-pi: e.chew d.thurmond

[nime2007]

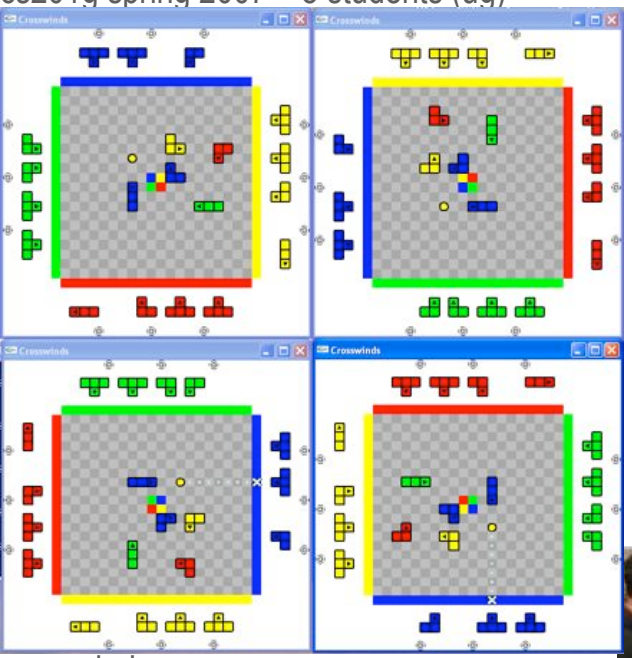
cs201g spring 2007 – 8 students (ug)



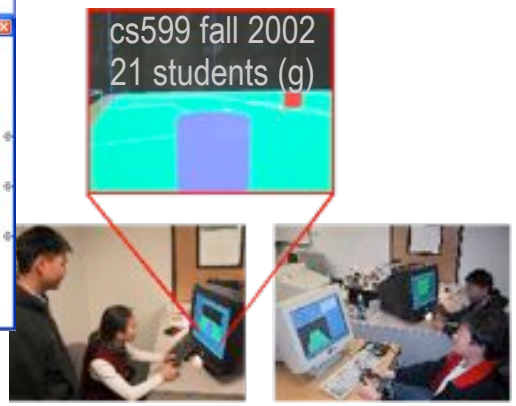
comp150-cis spring 2009 – 13 students (ug/g)

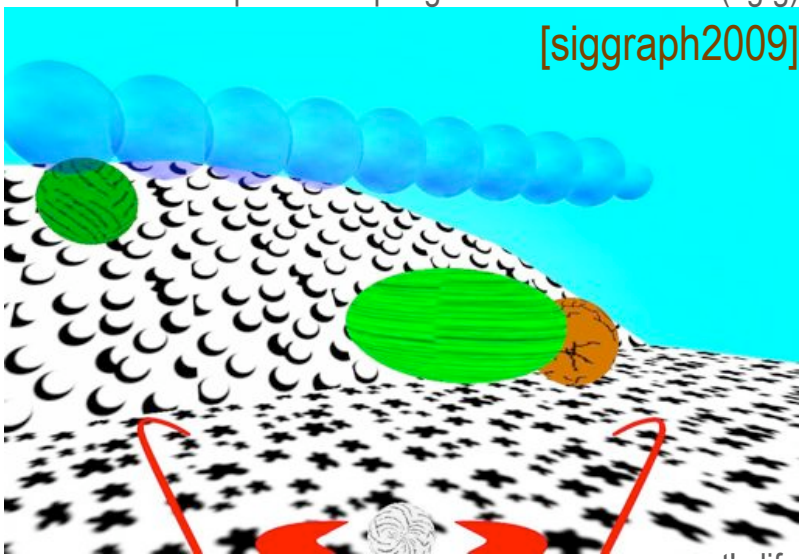
applications
-games-

[c&c2009]



crosswinds





an ant's life

perception

robust, non exact

- never twice the same observation
- never twice the same observer
- interactions between **dynamic systems**

time: synchrony and sequentiality

- causality principle
- *a posteriori* explanations
- result from our thought process
- **No Cartesian Theater**

dynamic systems

differentiate instance (individual) from class (invariants)

termination is irrelevant (and inevitable...)

nothing is immutable

“(useful) mathematics of complex systems” (brooks) ?

(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 interactive systems

interoperability

- combine technologies 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

computation

- algorithms
  - computable functions
- computing for accounting
  - exact
  - repeatable
- time abstracted
  - exact synchrony
  - strict sequentiality
  - lock-step parallelism/concurrency

time in/for computation?

- precedence, causality, synchronization
  - not exact, but not random!
- metric
  - time stamps for all data elements
  - computation takes time
- lifespan of data
  - volatile vs. persistent
- concurrent processing
  - asynchronous
  - decouple throughput and latency

sai principles

time

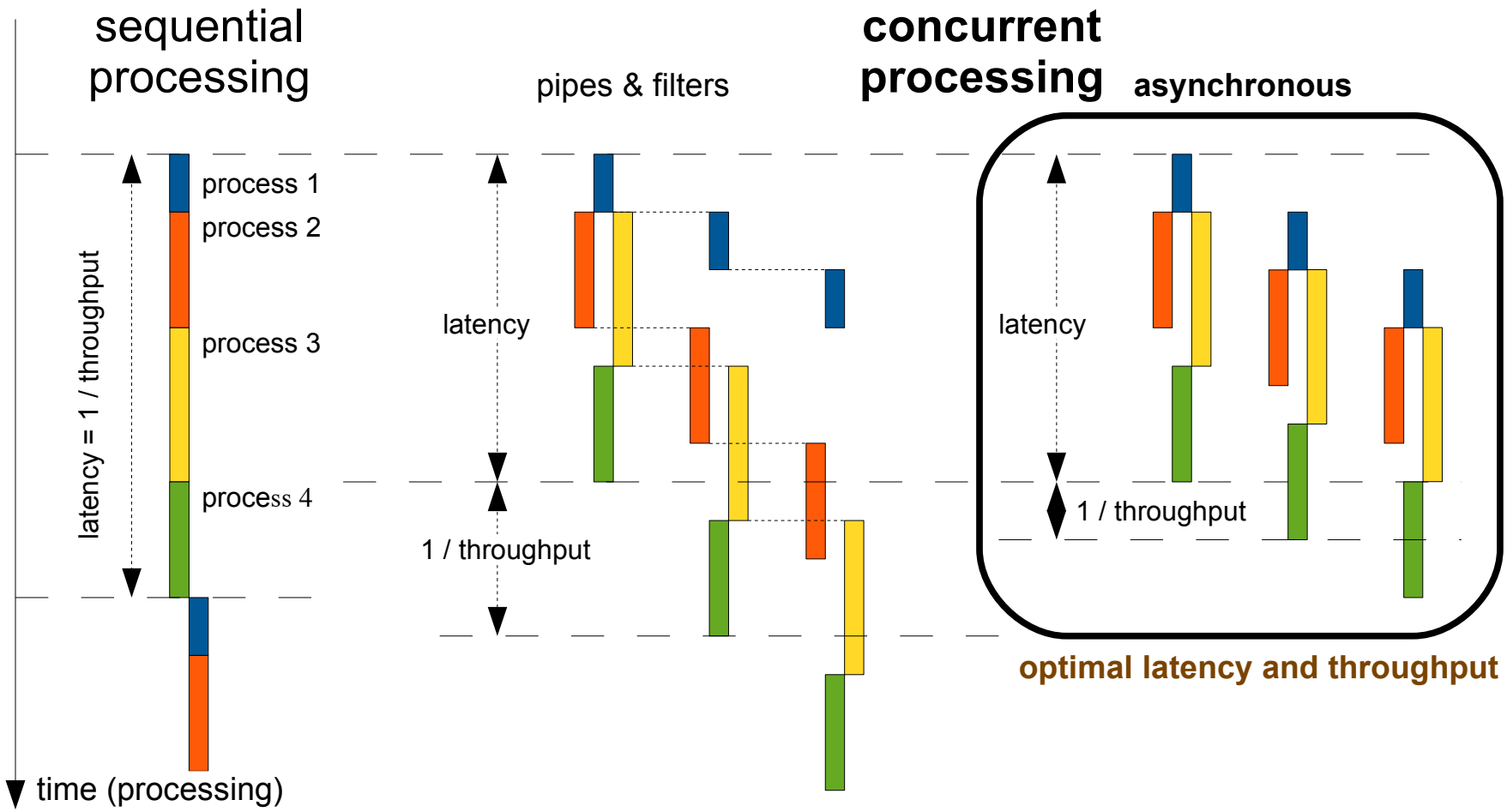
asynchronous concurrent processing

volatile vs. persistent data

architectural style [ICSE2004]

- high level abstractions
- (hybrid model or more general model?)

asynchronous concurrent processing



related work: concurrent computation

process calculi: csp, ccs, acp, pi-calculus

- synchronous message passing
- abstract time

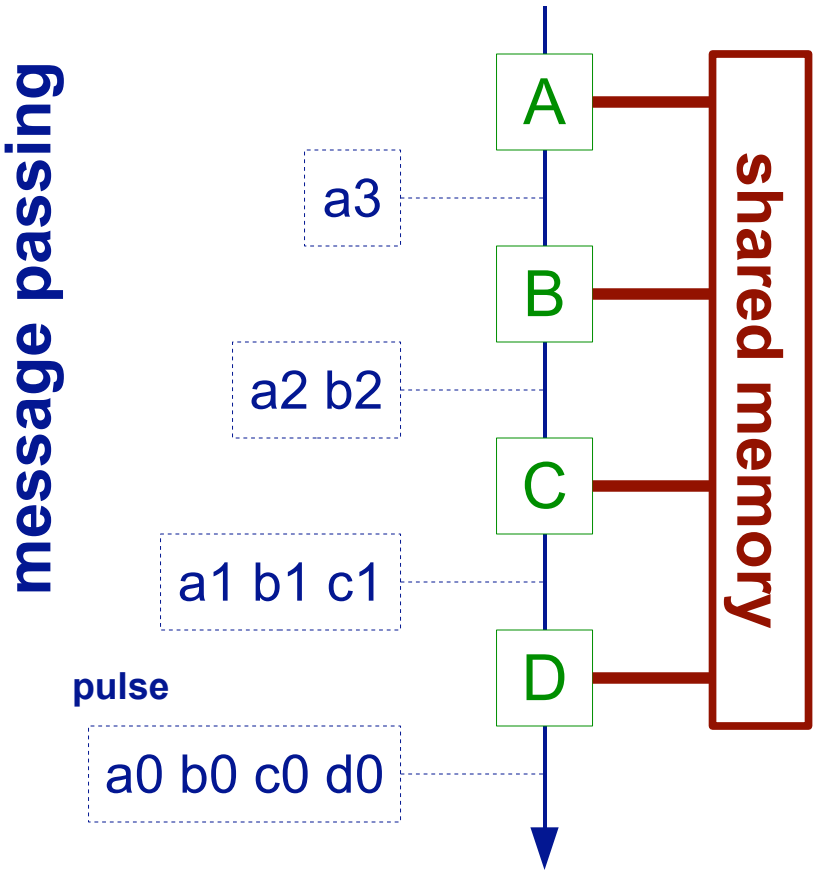
actor model

- asynchronous message passing
- concurrent processing

ptolemy

- heterogenous mixtures of models of computation
- hard real-time systems
- time is an external constraint (add-on)

volatile and persistent data



processing

persistent data

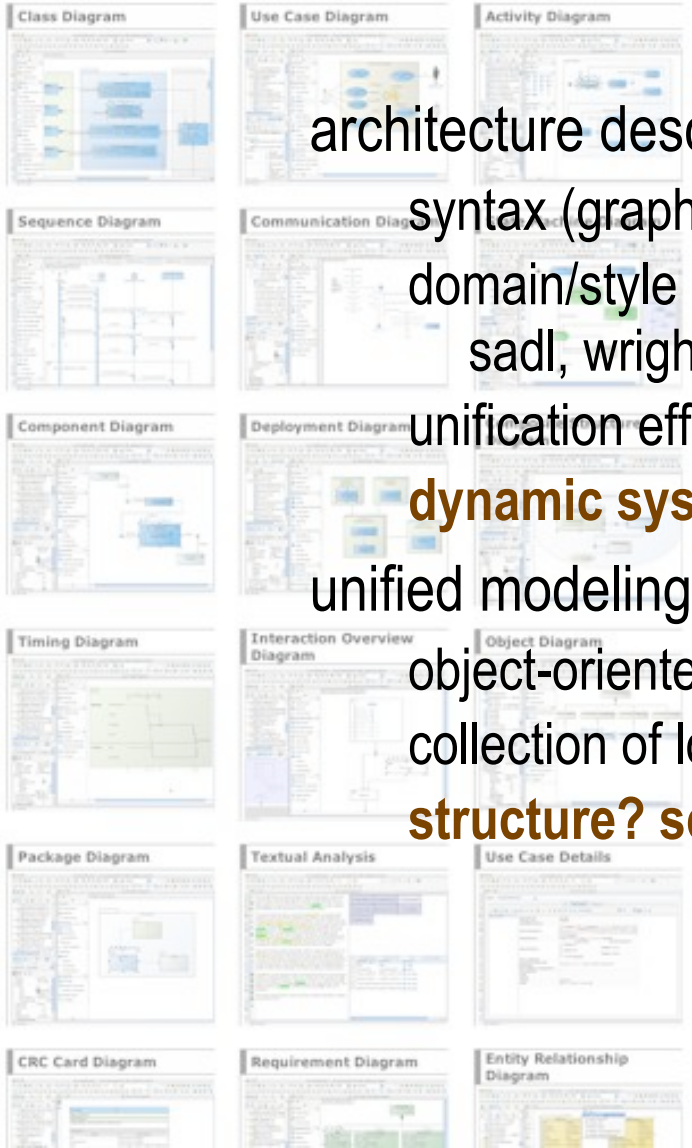
volatile data

$t0 < t1 < t2 < t3$

$d0 = f(a0, b0, c0)$

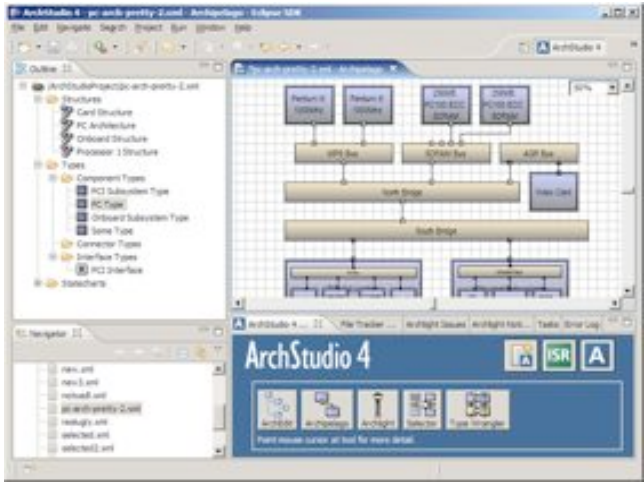
related work: adls and uml

Visual Paradigm for UML



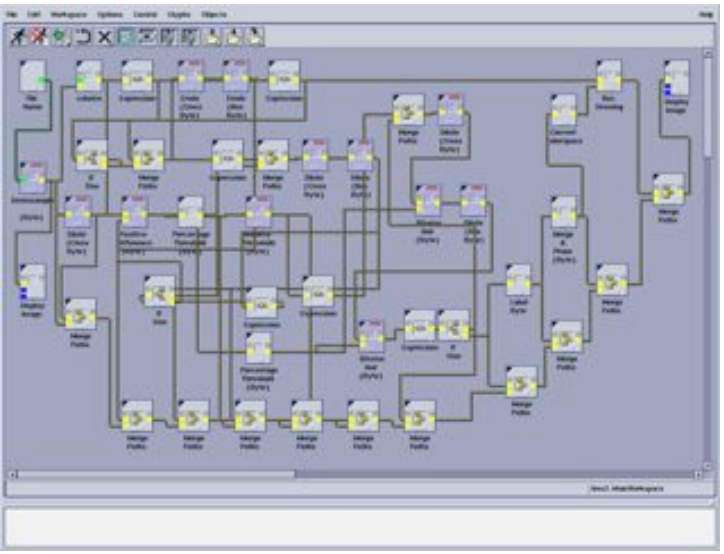
- architecture description languages (adls)
  - syntax (graphical) + semantics + tools
  - domain/style specific: rapide, c2, sadl, wright, etc.
  - unification efforts: acme, alfa
  - **dynamic systems?**
- unified modeling language (uml)
  - object-oriented concepts
  - collection of loosely related standards
  - **structure? scalability?**

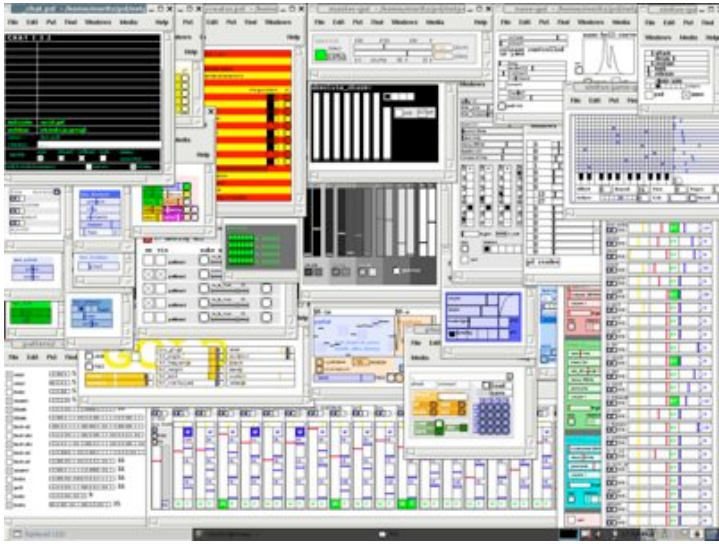
ArchStudio4 (C2)



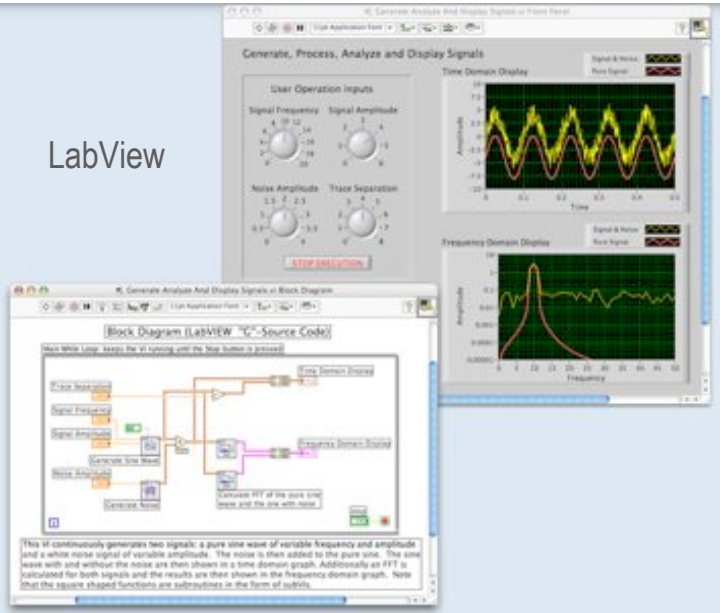
related work: visual programming

Khoros/Cantata

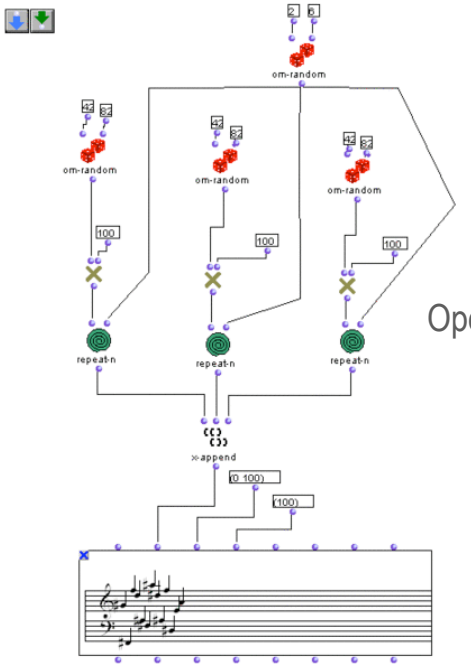




Pd

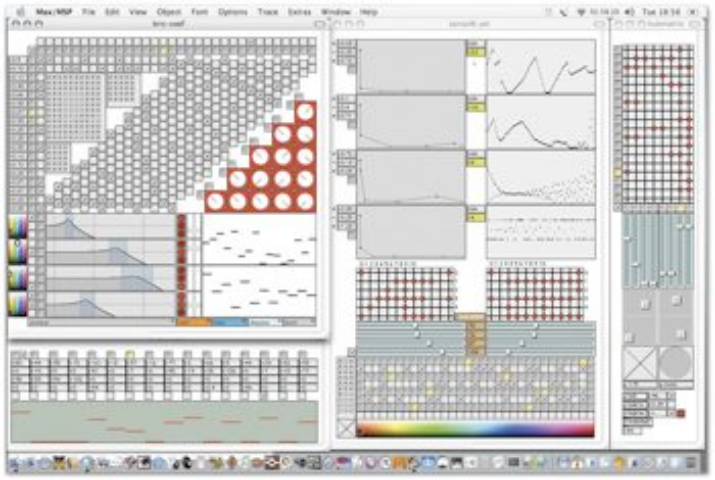


LabView



OpenMusic

Max/MSP



primitives & organizing principles

cell

processing unit (no state)

repository

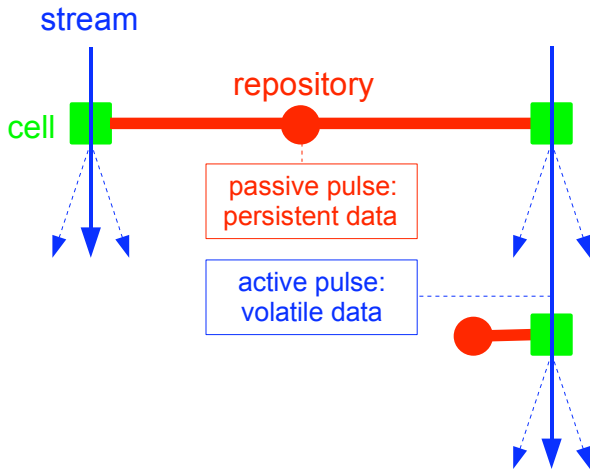
shared persistent data

stream

flow of volatile data

process dependency

process trigger

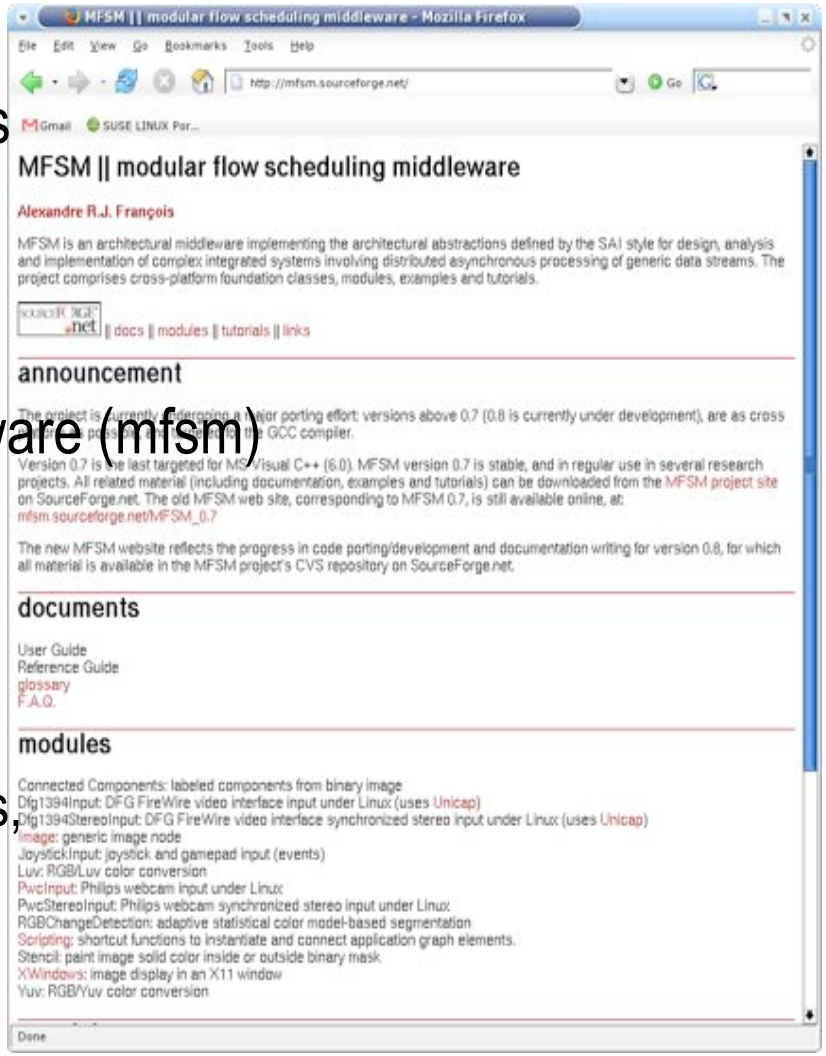


pulse

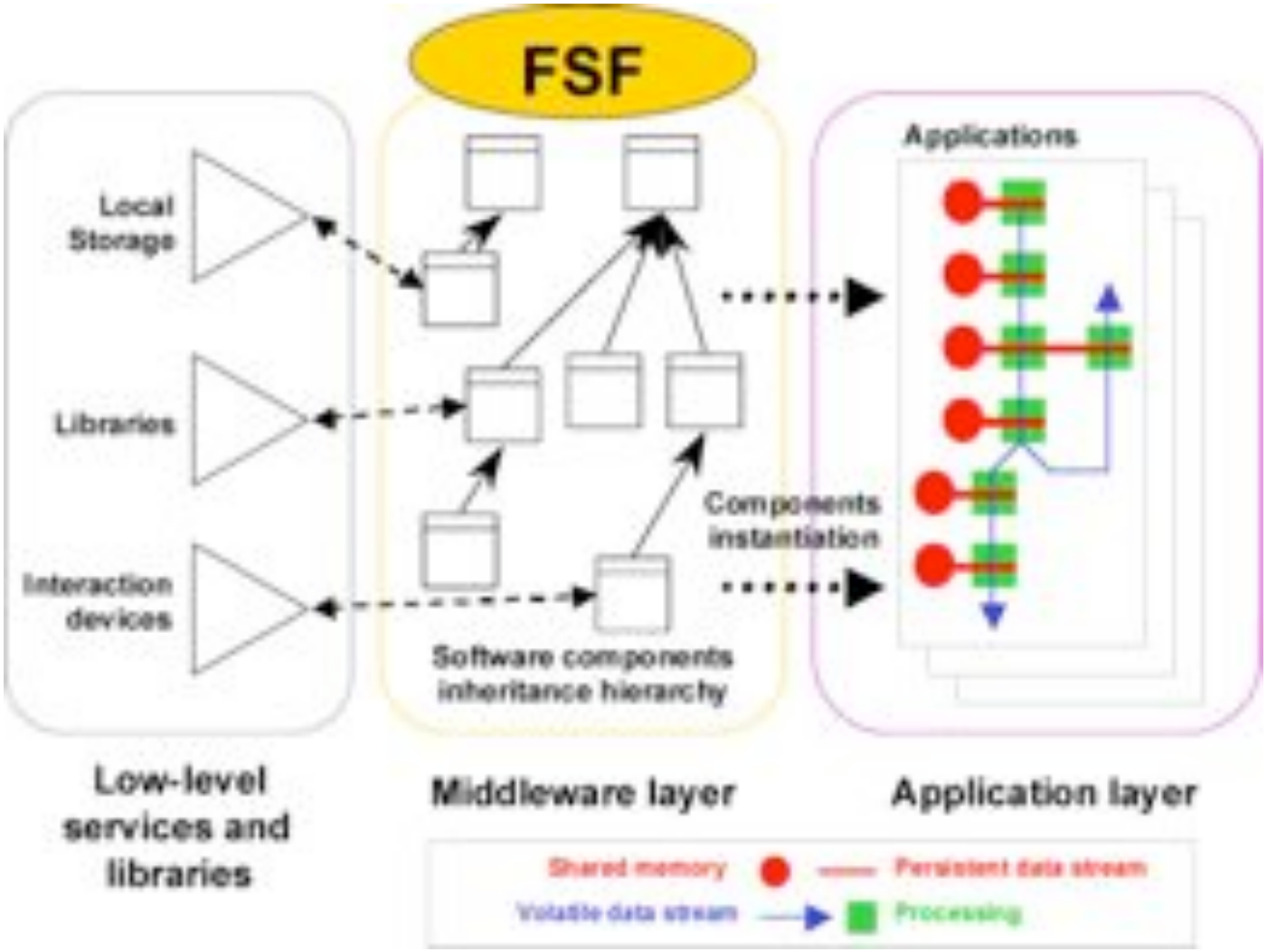
active (volatile) vs. passive (persistent)

architectural middleware

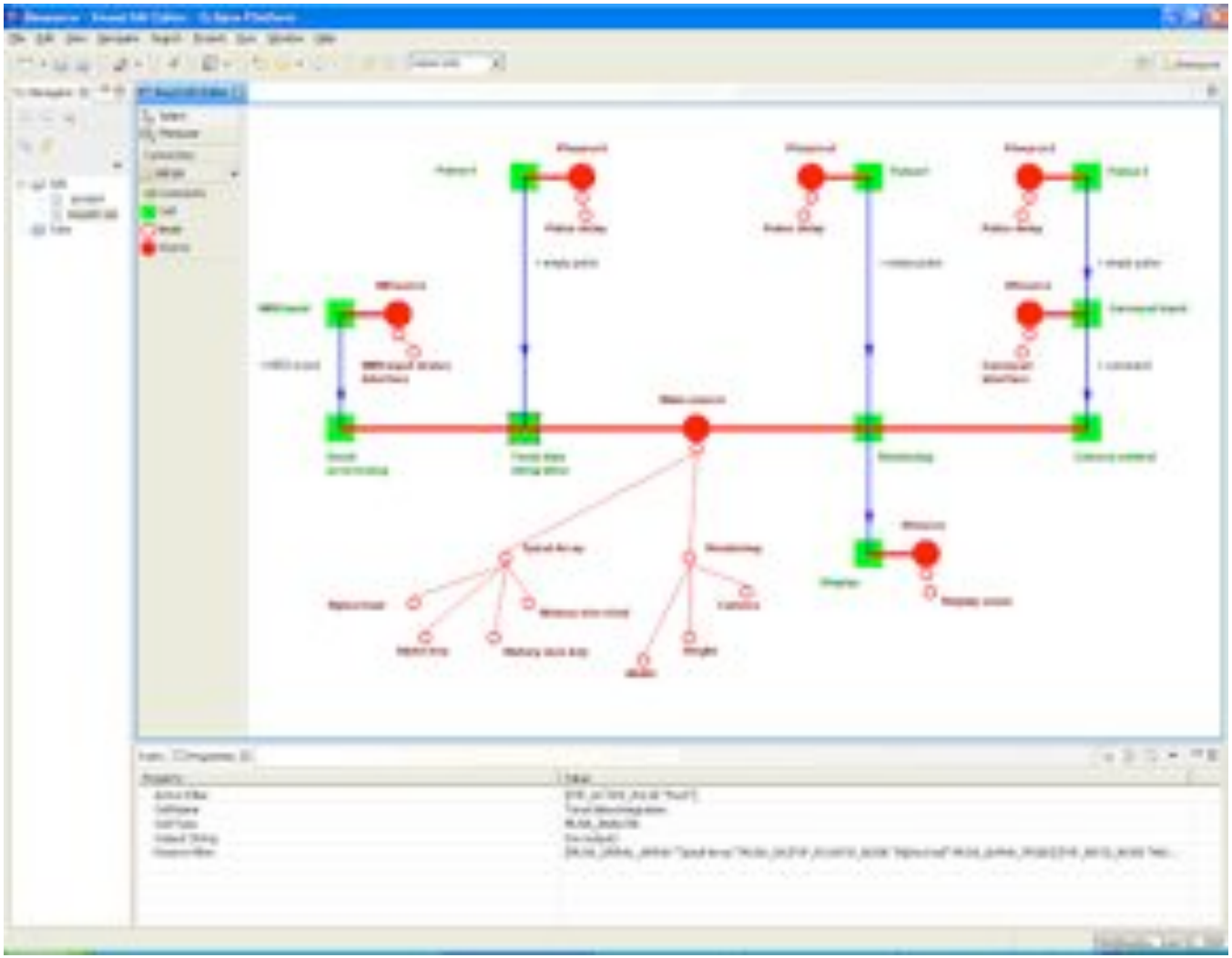
- support architectural abstractions
  - pulse, source, cell, etc.
  - direct mapping from logical specification to code!
- modular flow scheduling middleware (mfsm)
  - open source project: mfsm.SourceForge.net
  - c++, cross-platform
  - (gnu compiler)
  - base library, functional modules, documentation, tutorials



mfsm architecture



VisualSAI



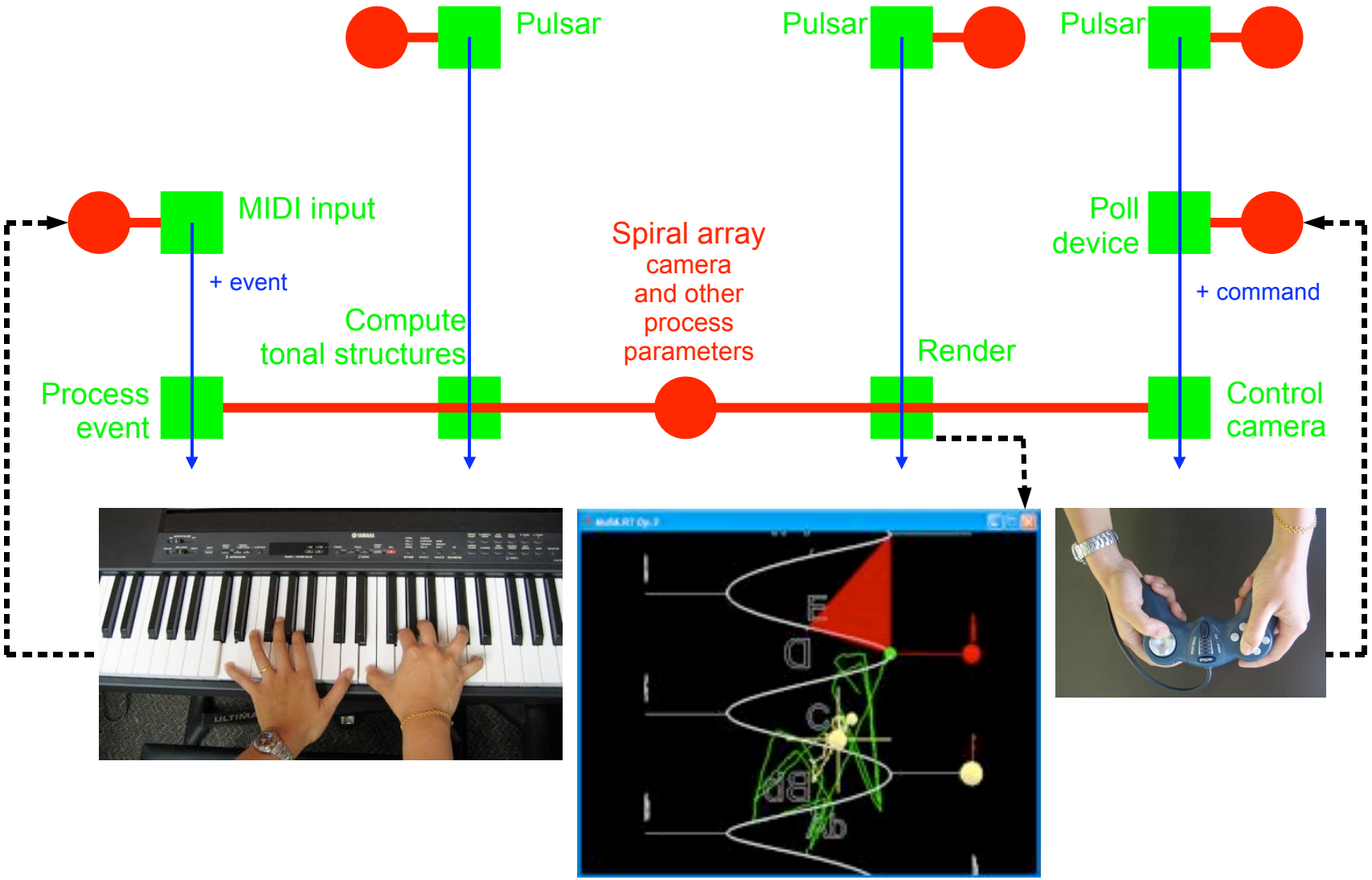
loops, time and concurrency?

3d engine event loop

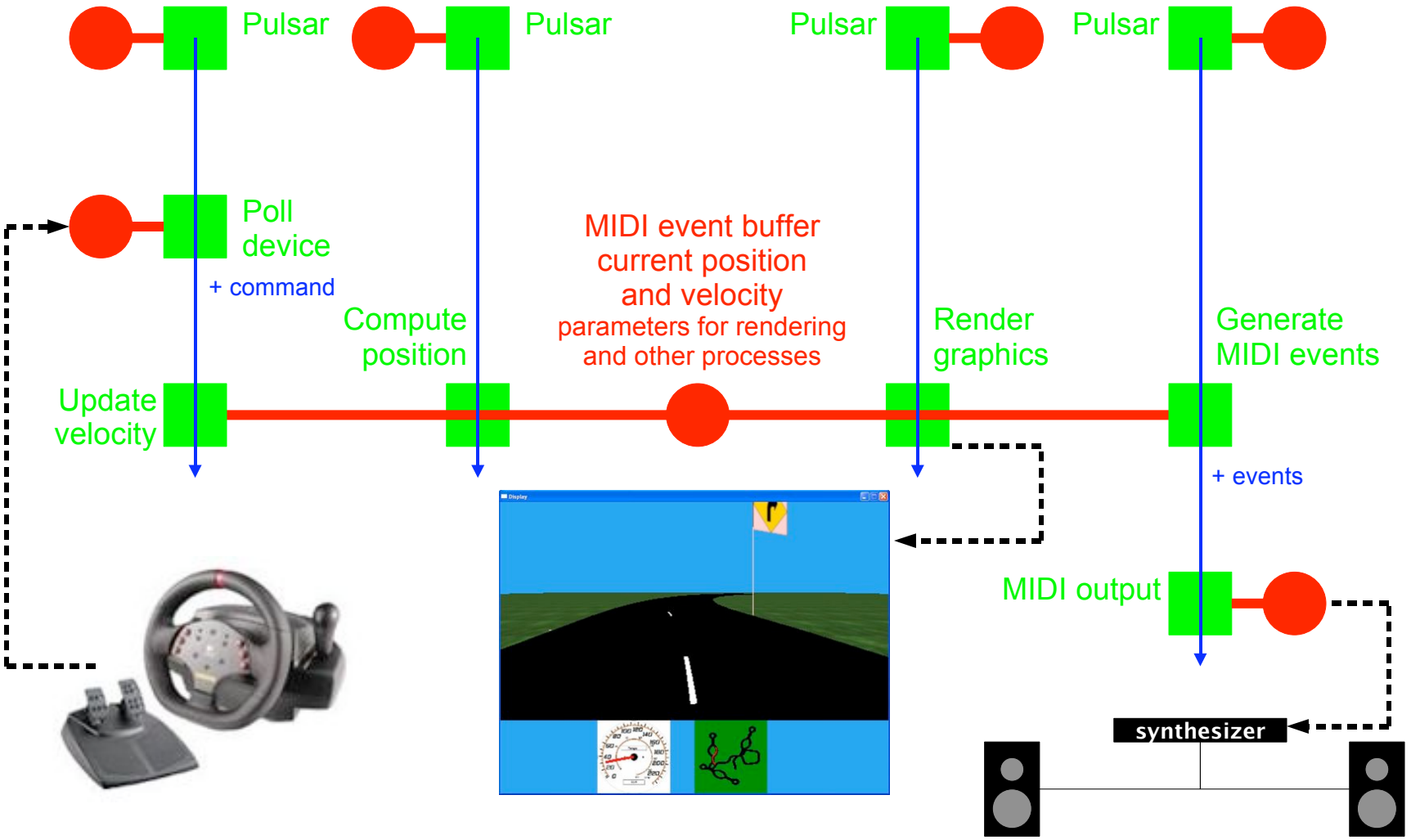
```
while(1){
     process_user_inputs();
     process_nw_inputs();
     process_physics();
     process_ai();
     render();
}
```

=> concurrent streams?

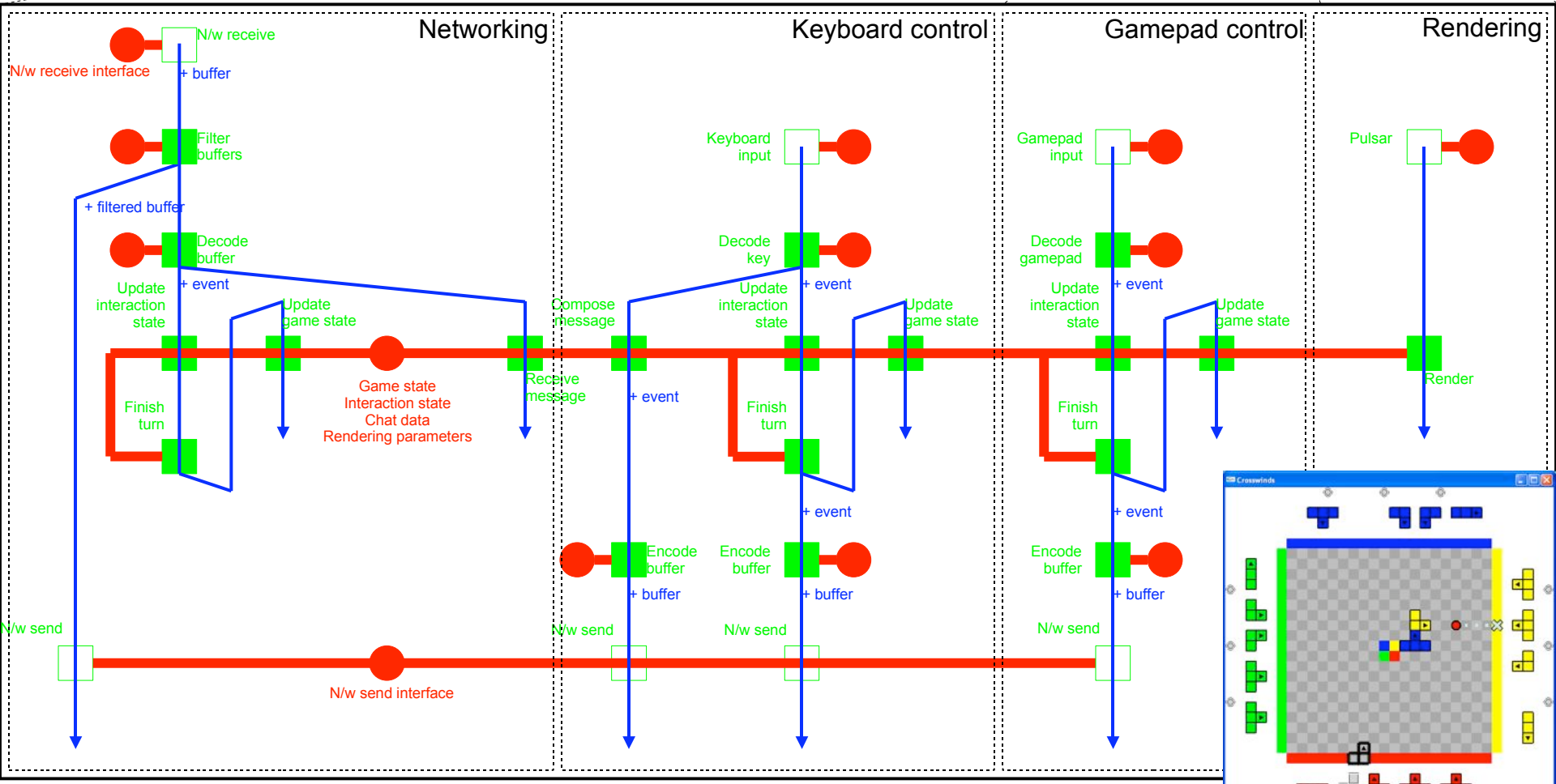
musa.rt architecture



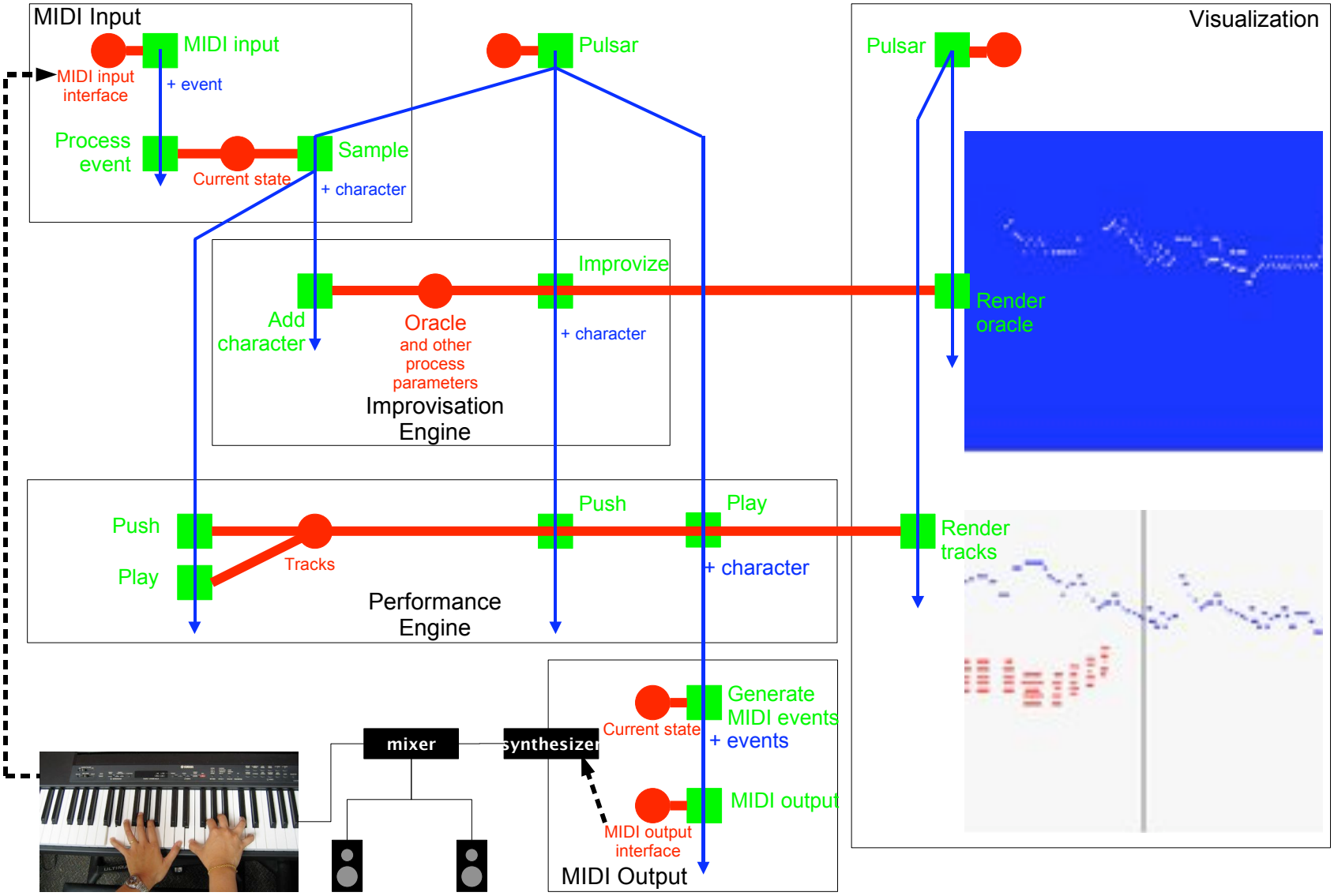
esp architecture



crosswinds architecture



mimi architecture



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

project architecture: a first draft...

- client-server architecture
  - game server
  - player clients
    - platformer client
    - builder client
  - maybe others...
- game state
  - description
  - communication protocol for updates
- other communication concerns
  - chat? voice? video?