Software Architecture for Immersipresence

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

- Video input
  + input image

- CAMSHIFT tracker
  + new position

- Rendering
  + composite image

- Image display
Refinement 1

- Video input + input image
- CAMSHIFT tracker + new position
- Rendering + composite image
- Image display

Step I
- Color conversion + HSV image
- Back-projection + back-projection image
- CAMSHIFT + new bounding box, size and orientation
- Rendering + composite image
- Image display

"Constrains"
Refinement 2

Video input
+input image

CAMSHIFT tracker
+new position

Rendering
+composite image

Image display

Step I
Color conversion
+HSV image

Histogram
node

Step II
Back-projection
+back-projection image

Step III
CAMSHIFT
+new bounding box, size and orientation

Rendering
+composite image

Image display

“Constrains”

Last known bounding box

Box update

Mean shift
+new position

Last known bounding box

Moments
+new bounding box, size and orientation

Rendering
+composite image

Image display
General Tracking Pattern

- Tracking source
  - Tracked targets

- Predict
  - +Predicted targets

- Track
  - +Tracked targets

- Acquire
  - +New targets

- Update
  - +Predicted targets

- +Tracked targets
- +New targets

- Tracking source
  - Tracked targets

- Video input
  - +input image

- Color conversion
  - +HSV image

- Back-projection
  - +back-projection image

- Histogram node

- Box update
  - Last known bounding box
    - +new position

- Mean shift
  - +new position

- Moments
  - +new bounding box, size and orientation

- Rendering
  - +composite image

- Image display
Vision for Robot: Stevi 1

**Stereo input and pre-processing**
- Stereo video input
  - +input left
  - +input right
- Color transform left
  - +HSV left
- Color data left
  - +Color data left
- Color transform right
  - +HSV right
- Color data right
  - +Color data right
- Barrier
- Video input
  - Video input interface left
  - Video input interface right

**Barrier synchronization**

**Stereo multi-target tracking**
- Predict targets
  - +Predicted targets
- Track targets
  - +Tracked targets
  - +Residual image left
  - +Residual image right
- Update targets
  - +Updated targets
  - +New targets

**Visualization**
- Render targets
  - +Composite image left
  - +Composite image right
- Image display left
- Image display right

Photo courtesy of ETRI
MuSA.RT
Music on the Spiral Array . Real Time

Co-PI: Elaine Chew

Spiral array camera and other process parameters

Camera control

User input

Commands

Rendering

Frames

Display

Tonal data integration

Event processing

Events

MIDI input

Spectator

Performer

Pulsar

Pulsar
ESP
Expression Synthesis Project

Co-PI: Elaine Chew

- MIDI event generation
- Pulsar
- Visual rendering
- MIDI event buffer
  current position and velocity parameters for rendering and other processes
- Driver input
- Velocity update
- Position computation
- Frames
- Display
- Events
- MIDI output
- synthesizer
Distributed Game Project

Game server graph

- Decoding
- Physics (motion+collision)
- Consistency (referee)
- World update
- Net Receive
- Net Send
- World description
- Encoding
- User input (player control)
- Game Server
- Database
- Spectator Client
- Player Client

Player client graph

- Net Receive
- Encoding
- User input (player control)
- Game Server
- Database
- Renderer
- Pulsar
- Parameters
- World description
- Image display

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
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.