Prelude

• Let's play !…

Outline

• A Software Architecture for Immersipresence
• The Modular Flow Scheduling Middleware
  – Introduction
  – FSF implementation
    • Cells, Nodes, Filters, Handles, etc.
  – Useful modules
  – MIDI Demo

A Software Architecture for Immersipresence (SAI)

Introduction

• A framework for the design and integration of interactive, immersive systems

Challenges

• Data and process nature:
  – Wide variety of data types and process modalities
  – Data streams
  – Real-time manipulation
• Requirements:
  – Efficiency, scalability, extensibility, reusability and interoperability
• S/W engineering:
  – distributed development
  – integrated applications
Related Work

- Multimedia toolkits
  - Object based but static approach to MM
  - Storage/Delivery/Rendering (no interaction)
- DirectX / DirectShow
  - DS: Capture/Process/Render (no interaction)
  - DX: Collection of independent libraries
- Libraries and standards
  - OpenGL, VRML, OpenCV, MIDI, etc.

A Modular System Architecture

Outline

- Motivation
- Flow Scheduling Framework
  - Dataflow processing of data streams
  - Generic data model
  - Generic processing model
- From components to applications
  - Stream routing
  - Synchronization
  - Example applications

Flow Scheduling Framework

- Extensible set of classes that provide basic synchronization functionality and composition mechanism to develop data-stream processing components
  - Generic data model
  - Generic processing model
  - Data-flow programming model
Dataflow Model for Data Streams: Pulses

- $a_0, b_0, c_0, d_0$
- $a_1, b_1, c_1$
- $a_2, b_2$
- $a_3$
- Persistent data
- Volatile data
- $t_0 < t_1 < t_2 < t_3$

Data Model

- Node
  - Type
  - Name (identifier)
  - Local attributes
    - Time stamp
    - Duration
  - Shared attributes (components)
- Flexible and extensible Object-Oriented design
  - Encapsulation
  - Inheritance
  - Polymorphism
- Efficient dynamic binding

Generic Processing Unit

- Cell
  - Processing center
- Active stream
  - Volatile data
  - Can only be augmented
  - Multithreading
- Passive stream
  - Persistent data (source)
  - May be updated
  - Feed-back

Xcell Close-up

- Active stream
- Passive stream
- Male active plug (input from previous cell)
- Female active plug (output to next cell)
- Male passive plug (to source or previous cell on daisy chain)
- Female passive plug (to next cell on daisy chain or self-terminating loop)
- Active pulse flow
- Passive pulse flow
- Passive stream filtering
- Updated passive stream
- Augmented active stream
- Active stream filtering
- Passive stream merging
- Process parameter
- Process input
- Source
- Input
- Xcell
- Output

Outline

- A Software Architecture for Immersipresence
  - Motivation
  - Flow Scheduling Framework
    - Dataflow processing of data streams
    - Generic data model
    - Generic processing model
  - From components to applications
    - Stream routing
    - Synchronization
    - Example applications
Synchronization
- Streams define independent time referentials
- Processing is asynchronous inside streams (multithreading)
- Time transforms require buffering in passive stream
- Operations:
  - Resampling
  - Merging and synchronization

MIDI Demo (Live)

MIDI Demo (Buffer/File)

Virtual Mirror

Graphics

MuSA.RT
Summary

- Parallel, asynchronous data stream processing model with synchronization features:
  - efficiency, scalability, consistency
- Modularity:
  - reusability, extensibility, interoperability
- “Distributability”
- Already used in a number of applications
- Open Source implementation
  http://mfsm.sourceforge.net

Outline

- A Software Architecture for Immersipresence
  - The Modular Flow Scheduling Middleware
    - Introduction
    - FSF implementation
      - Cells, Nodes, Filters, Handles, etc.
      - Useful modules
      - MIDI Demo

The Modular Flow Scheduling Middleware

MFSM

- Open source implementation
  - MS Visual C++ 6.0 (+)
  - Download, install and MIDI demo compilation

Special Package Content

- FSF
  - Core framework elements
- Toolkits
  - Image Toolkit
  - Windows Toolkit
  - More to come…
- Documentation (NEW)
  - Includes Modules
  - MIDI demonstration application

A module is a set of cells and nodes which together implement a given functionality (or by extension a set of related functionalities).
A (module) library is a packaged group of related functional modules
A toolkit (library) is a module library implementing functionalities of general interest

Design Choices

- Uses C++ and STL
  - Efficient, standard and popular
  - Template classes: compact and reusable code
  - Standard template containers
  - Run-time type management features
  - Portable
- Core is independent of any external libraries
- Specific modules might rely on external libraries
  - E.g.: DirectShow, DirectPlay, etc.
Outline

- A Software Architecture for Immersipresence
- The Modular Flow Scheduling Middleware
  - Introduction
  - FSF implementation
  - Cells, Nodes, Filters, Handles, etc.
- Useful modules
- MIDI Demo

FSF Objects

FSF Objects

- System
  - Clock
  - Factory info

- Node
  - Constructors
  - Attribute access

- Handle
  - Active
  - Passive

- Factory
  - Node
  - Cell

- Cell
  - Constructors
  - Process

- Source

FSF Objects

Nodes and Pulses

- CNode
  - Name
  - Time stamp
  - Duration
  - Constructors and operator=
  - Clone
  - GetTypeID
    - Not used for filtering
    - See Factories

- Pulses
  - CPulse derived from CNode
    - Name: “Root”
    - Time stamp
    - CActivePulse
      - Branching counter
    - CPassivePulse
      - Garbage collecting

- System
- Clock
- Factory info

Cell and Source

- CCell is a virtual base class
  - Passive connections (M/F)
  - GetTypeID (see Factories)

- CSource
  - Passive pulse
  - Derived from CCell for simpler cell/cell and cell/source passive connection management

Xcell

- Derived from CCell
- Active connections (M/F)
- Active filter, Output name
- ActiveThread (StartActiveThread)
  - Triggered by Cell on Active M connection
  - Executes in new thread (created by StartActiveThread)
  - Filters Active Pulse
  - Calls process function if both passive and active handles are not NULL
  - Releases Active Handles if relevant
  - Calls StartActiveThread for all cells on F Active connections
- Process function
  - Called by StartActiveThread static member
  - Called only if passive and active handles are not NULL
  - Handles guarantee that relevant nodes on passive and active pulses are not destroyed during execution

Filters and Handles

- Filters
  - Template classes
  - Base non-template class for pointer containers
  - Different matching implementations for active/passive filters (different semantics)

- Handles
  - CActiveHandle
  - CPassiveHandle
    - Lock for pointed node (garbage collecting)
Type Nodes and Type Cells

- **Type nodes**
  - Template class for simple data types
    - long (32 bits), float (32 bits), double (64 bits), bool (1 bit), std::string, more?
  - Non template base class
- **Type cell**
  - Template class for Type Node value access cell
    - Process function has modified semantics
      - If active handle not NULL: Set value
      - If Active Handle is Null: Get Value
  - Non Template base class
  - Currently not usable for Get Value in MFC app

System and Factories

- **System**
  - One unique instance
  - Unique clock (time stamps)
  - Factory lists (maps)
- **Factories (Node and Cell)**
  - Needed for run-time instantiation of objects of arbitrary type
  - Used in scripting functions
  - Will be used for serialization and programming environment

Outline

- A Software Architecture for Immersipresence
- The Modular Flow Scheduling Middleware
  - Introduction
  - FSF implementation
    - Cells, Nodes, Filters, Handles, etc.
  - Useful modules
  - MIDI Demo

MIDI Module

- See MFSM documentation

Rendering Module

- See MFSM documentation

Image Display Module

- See MFSM documentation
Outline

- A Software Architecture for Immersipresence
- The Modular Flow Scheduling Middleware
  - Introduction
  - FSF implementation
  - Cells, Nodes, Filters, Handles, etc.
  - Useful modules
  - MIDI Demo

MIDI Demo

- MFC dialog-based application
- Uses:
  - Image Toolkit
  - MIDI Module for I/O
    - Added image drawing cell
    - Image Display Module

MIDI Demo Graph

Next MIDI demo

- What if no MIDI input device?
  - Read from file…

MIDI Demo Graph (Buffer/File)

Summary

- MFSM is one possible implementation of SAI
  - Core is very weakly platform dependent
- Extensible: more components will be developed (cf class project…)
- Several modalities exist for the application environment (MFC or better…)