Siren, CSL, and CRAM:

Building Large-scale Interactive Systems

Stephen Travis Pope

Center for Research in Electronic Art Technology (CREATE)

Graduate Program in Media Arts and Technologies (MAT)

University of California, Santa Barbara (UCSB)

stp@{create,mat}.ucsb.edu

Overview: CREATE/MAT Sound Group

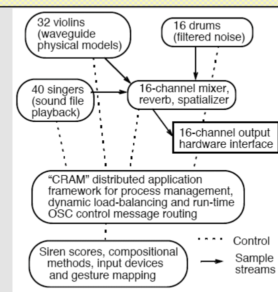
- The Context: Siren, CSL, and CRAM
- Siren: Object models, control mappings, and GUIs for composers and performers
- CSL (“Sizzle”): The CREATE Signal Library for distributed DAudioSP
- CRAM: DBs and tools for managing & monotoring distributed real-time apps
- Examples, evaluation, next steps, discussion

UCSB Labs, Programs, and Teams

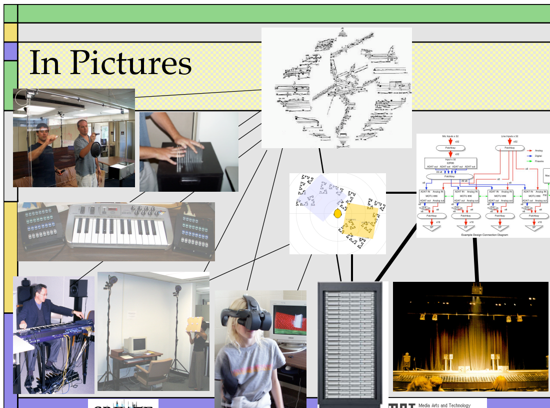
- CREATE
  - JoAnn Kuchera-Morin, Curtis Roads, Ioannis Zannos, STP, etc.
- UCSB Music Dept.
  - Comp: J. Haladyna, L. Hogan, K. Tanaka, J. Feigin
  - Piano, Winds, Perc., Voice, Theory, Ethno/musicology
- Graduate Program in MAT
  - Music, **Art Studio** (LeGrady/IA, Jevbratt/WA, Novak/VA, Peljhan/IA) , **CS** (Turk/CV, Almeroth/Net), **ECE** (Gibson/Comm, Mitra/DSP, Manjunath/ImProc)
- CNSI: California NanoSystems Inst. @ UCSB/UCLA
  - CompuChem, BioInf, MaterialsSci, MolecGenetics, MAT
- UCSB Digital Media IGERT
  - 5 Years @ 14 PhDs per year, assoc. depts

Synthesis/Performance Group Goals

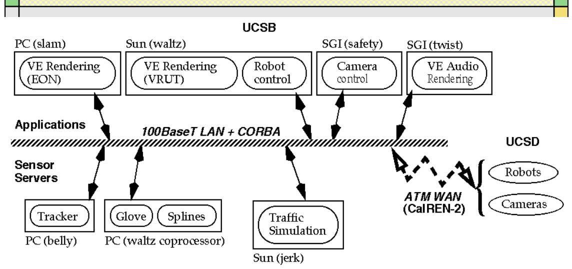
- Support reliable “orchestra-scale” sound synthesis, multi-modal gestural sensing and control, and pluriphonic projection (up to 128 channel output in the CNSI sphere)



In Pictures



Example: ATON DRIVE (1998-2000)

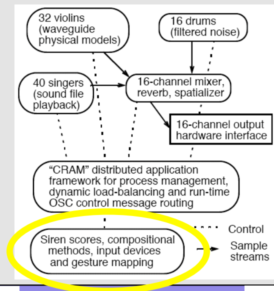


What's Needed?

- Representations and interfaces for composers and performers (Siren)
- Scalable DAudioSP framework (CSL)
- Management infrastructure for distributed RT OO software (CRAM)
- A really cool loud/bright/sensing space to play it in! (CNSI)

CREATE MAT Media Arts and Technology Graduate Program

Part 1: Siren



CREATE MAT Media Arts and Technology Graduate Program

Siren (MODE, HSTK, DoubleTalk)

- Smalltalk-based object-oriented framework for sound/music description and processing, under development since 1984
- Focus on structure representation, control mapping, and composition, rather than on performance, DSP, or notation
- API/Platform for music representation and composition language development

CREATE MAT Media Arts and Technology Graduate Program

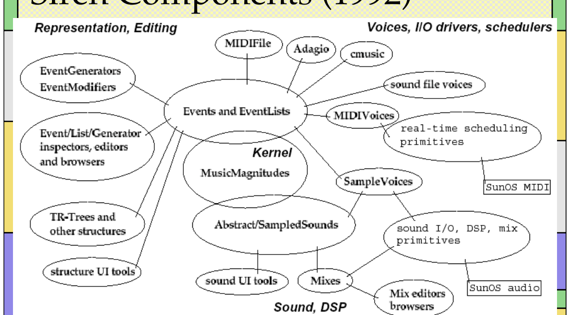
What's Siren?

[440 Hz, (1/4 beat), 44 dB]
evtList mapPItches: gamut.
evtList playOn: Voice default.

- **Smoke** music representation language
  - Music magnitudes, events, event lists, generators, modifiers, struct. algorithms, …
  - Organize timing, tuning, timbre, space, gesture, grouping, versioning
- I/O **voices** (players, property-parameter mappers) for many formats: (m11-SC3) note lists, OSC, MIDI, XML, CORBA, …
- Multi-threaded RT **scheduler**
- GUI **widgets** and apps for music
- (OO/R)DBMS interfaces for **persistency**

CREATE MAT Media Arts and Technology Graduate Program

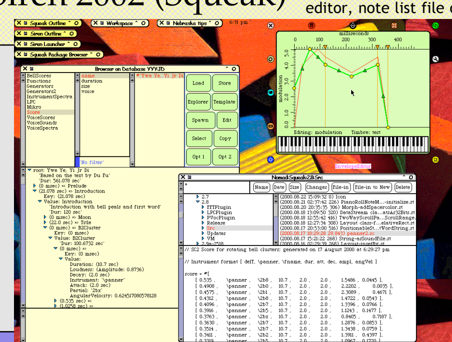
Siren Components (1992)



CREATE MAT Media Arts and Technology Graduate Program

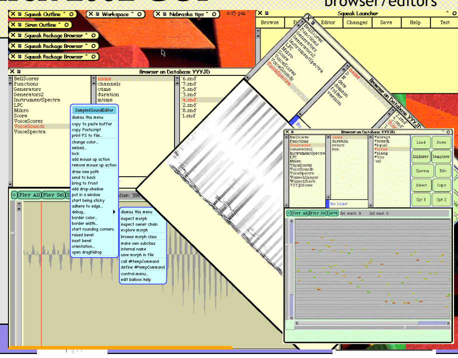
Siren 2002 (Squeak)

Score browser, function editor, note list file output



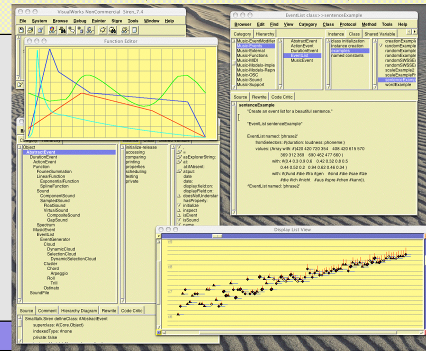
Siren 2002 GUI

Sound/spectrum/score browser/editors



Siren 2003 (VisualWorks)

Demo



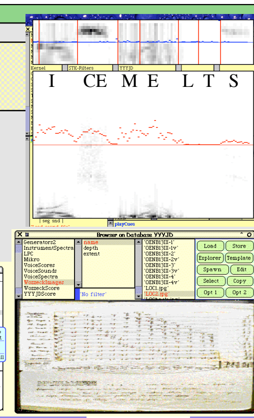
Composition in Siren

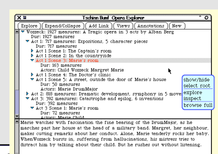
```
RotatingBellCluster class methodsFor: 'HeavenEverywhere'
setup: dataSelector functions: fcnSelector
    "(RotatingBellCluster setup: #b2Data functions: #b2aFunctions) eventList explore"
    | i dat parts clust fcns |
    parts := OrderedCollection new: 16.             "Load partial data"
    dat := self perform: dataSelector.
    i := 1. [i < dat size] whileTrue:               "Create partial list"
      [parts addFirst: (BellPartial named: (dat at: i) freq: (dat at: i + 1)
                        ampl: (dat at: i + 2) dur: 10.7).   i := i + 3].
    clust := self new partials: parts.              "Load function data"
    clust name: #'rBell.b2.14c'.                    "Set your name"
    fcns := self perform: fcnSelector.              "***Get the time-function list ***"
    clust spectralFcn: (fcns at: 1).                "Spectral weighting function"
    clust densityFcn: (fcns at: 2).                 "Time/density function"
    clust velocityFcn: (fcns at: 3).                "Mapping of pitch to angular velocity"
    clust duration: 90; att: 2.0 sec; dec: 2.0 sec. "Set bll cluster parameters"
    clust pitch: 14.07 Hz.                          "= Base * 1.61803398"
    clust ampl: -7.2 dB.                            "Mezzopiano"
    clust voice: #panner2.                          "Add to panner voice"
    ^clust                                          "Answer bell cluster"
```

CREATE — MAT Media Arts and Technology Graduate Program

Recent Siren Apps

- 8S speech segmenter and database
- Opera Browser
- SeSpSp and EMA prototypes





Siren Evaluation

- **Medium-sized**: 250 classes, 12 kLOC core, 24 total
- **Multiplatform**: CPU, OS, GUI, RMI, control, ST-impl.
- **Flexible/Comprehensive**: models, prop. organization, group hierarchy, persistency, IO, GUI
- **Declarative/Procedural**: OO arch., msg-based agent threads, parser generators, grammars, Slang xlators
- **AI** frameworks (rules, nets, logic, CSP, genetic)
- **DB** support: VI, OO (Gemstone, SMS_DB), RDBMS
- <u>Still my favorite</u> (real-estate: productivity).
- Smalltalk's still considered obscure or esoteric.

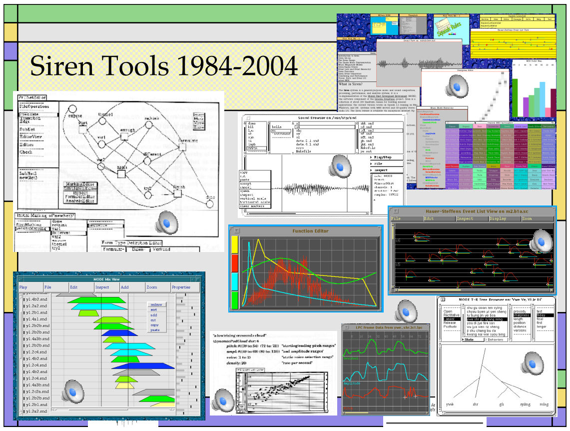
CREATE — MAT Media Arts and Technology Graduate Program

Evaluating Smalltalk as a Platform

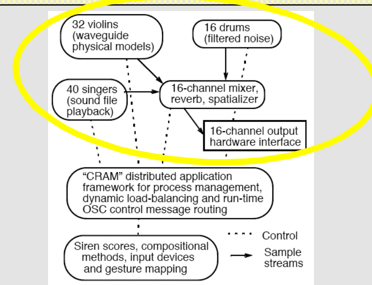
- **Language**: uniform, reflective, malleable, stable/std
- **Run-time**: Run-time types and polymorphism; Compiler available at run time; GC works; persistent run-time state in VI, LWP, thinks it's the OS
- **IDE Tools**: Browsers, refactoring, classes rather than files; awesome debugger and change manager, best GUI painter
- **Delivery Environment**: Change/Config Mgmnt with Envy or Store (parcels/projects); Shared desktops (RemEvts, SHARK, Kansas, DST); External calls: RMI/CORBA/DST

CREATE — MAT Media Arts and Technology Graduate Program

Siren Tools 1984-2004



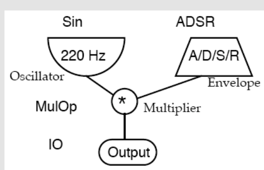
Part 2: CSL (1998-2004)



The CREATE Signal Library (CSL, "sizzle") ("chill?")

Demo

- General-purpose, portable C++ framework for distributed, real-time digital audio synthesis and processing
- Used for stand-alone applications, plug-ins, OSC servers, etc.



CSL Relatives

- Like Cmix, STK, Siren, JSyn, MxV, or CLM
  - Delivered as a library in a general-purpose programming language
- Unlike SuperCollider, Csound, Max
  - Not its own language
  - No scheduler
  - Uses C++ development environment

Why on Earth another one???

- Cmix -- old, flaky
- SuperCollider -- different question, complex
- Csound, Music-N -- not languages, source clarity
- Jsyn -- closed DSP kernel
- STK -- PM-centric, tick model
- CLAM -- way complex
- CLM -- who knows LISP?
- Siren/Squeak -- who knows Smalltalk?

Our Requirements

- Simple, easy to learn
- Flexible, multi-purpose
- Portable
- Scalable
- Embeddable
- Distributable
- Network-oriented
- Debuggable

CSL Background

- "CREATE Oscillator" -- 1998, CORBA_A/V
- MAT 240D course (digital audio synthesis techniques, Spring '01, '03)
  - CO1 (minimal), 2 (full-featured)
  - CSL_lean (redesign from scratch by one person)
  - CSL3
- Designs driven by immediate needs for concrete applications (pieces, theses, etc.)

CSL3 Basics

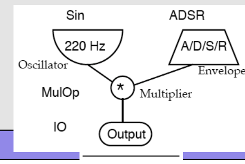
- **Buffer** objects (1-4 classes)
  - Multichannel non-interleaved sample storage
  - "Smart" object, not just a (float **), ptr. mgmnt.
  - Handle malloc/free, filling statistics, etc.
- **FrameStream** classes (Ugens) (many)
  - Respond to the message next_buffer(input, output)
  - Processors have a FrameStream as input
- **Mix-in** classes (vs. wrapper classes)
  - Phased, Positionable, Writeable, Cacheable, etc.

"Hello world" in CSL

Demo

Sine wave with envelope

```
// Create a sine oscillator -- this is a comment
        Sine osc(220.0);
// Create an ADSR envelope -- args are (dur, a, d, s, r)
        ADSR env(3.0, 0.06, 0.2, 0.2, 1.5);
// Create a multiplier
        MulOp mul(osc, env);
// Plug it into the output driver
        globalIO.set_root(mul);
```



Sine Osc Alternatives

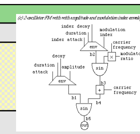
Processor::set_input()
Ugen::set_scale()
Ugen::set_offset()

```
// Use the envelope object as a generator and processor
    SumOfSines osc(220.0, 1, 5, 0.7...);   // make a sum-of-sines
    Triangle env(3.0);                     // triangle envelope
    env.set_input(osc);                    // send osc as input to env
    gIO.set_root(env);                     // env is root

// Use the osc's scale (volume control or AM) input
    SquareBL osc(220.0);                   // make a band-lim square
    Gaussian env(3.0, 0.2);                // envelope with bell width
    osc.set_scale(env);                    // set osc scale to env
    gIO.set_root(osc);                     // osc is root
```

Reverb'd Panning Dual-Env. FM

(7 Ugens, minimal, procedural)



```
        //// FM instrument with stereo panning and reverb ////
    ADSR a_env(1, 0.01, 0.1, 0.1, 0.6);    // create ampl. env., ADSR(dur, a, d, s, r)
    ADSR i_env(1, 0.001, 0.1, 0.5, 0.5);   // create FM mod. index env.
    i_env.set_scale(110);                  // scale i_env by base freq.
    Sine car, mod(110);                    // create 2 sine oscs: carrier & modulator
    mod.set_scale(i_env);                  // scale the modulator by the i_env
    mod.set_offset(220);                   // add in the base freq.
    car.set_frequency(mod);                // set the carrier's freq. to the modulator
    a_env.set_input(car)                   // plug the carrier into the a_env's input
    Sine pos(0.25);                        // create an LFO for panning
    Panner pan(a_env, pos);                // plug the a_env into a stereo panner
    Stereoverb verb(pan);                  // plug the panner into a stereo reverb
    gIO->set_root(verb);                   // plug the reverb into the output
//  gMixer->add_input(verb);               // or add it to a global mixer
```

Demo

CSL FrameStream Details

- Core FrameStream methods
  - next_buffer(in, out) - fill in a buffer's worth of frames
  - next_sample(in, out) - answer 1 sample; adjust phases…
  - is_fixed_over(in) - is the receiver's value fixed over range?
  - is_active() - are a graph's envelopes on?
- Policies for handling next_buffer() with multi-channel I/O buffers: call mono_next_buffer() and iterate (vs. copy - FanOut, Splitter)

CSL Sources, Controls, and Processors

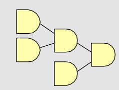
- **Sources**
  - Oscillators (perfect, BL), SumOfSines, Noise, SoundFiles, Chaotic/IteratedFS, IFFT, Physical Models, Granulators, Signal windows
- **Control**
  - Envelopes, LFOs, LFNoise, ProbDists, DynamicVariables, OSC, MIDI, GUI, CORBA, XML, note lists, Feature extractors, Input followers
- **Processors**
  - Operators, Mixers, Filters/banks, Reverbs, (N-M)Panners, DelayLines, FDN, WaveShape, Lo-latency Convolution, FFT/IFFT, LPC/FIR
- **Support**
  - RingBuffer, ThreadedFrameStream, BlockResizer, RateConvertor, Splitter/Joiner, FanOut (needed), Interleaver/Deint., Test main()s
  - Tools: FIR/Reverb IR Design, Spectrum DBs, Control-mapping

CSL I/O - The Pull Model

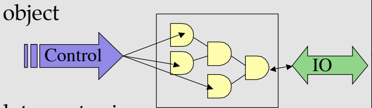
- The IO object
  - Holds onto a DSP graph (root node, may be mixer)
  - Receives periodic call-backs from somewhere (driver such as PortAudio, CoreAudio, UDP, TCP, file I/O scheduler, etc.)
  - Sends next_buffer() to the root of its graph with IO buffer pointers filled in (timing base freq.); may interleave
- Subclasses for PortAudio, CoreAudio, UDP/TCP RemoteIO, FileIO, streaming, etc.

The Big Picture of CSL

- Basic DSP graph



- Connected to control input (OSC, MIDI, GUI, CORBA, XML), and IO object



- Buffering and latency tuning

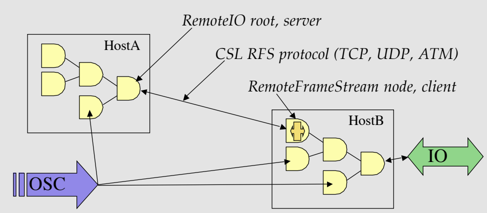
CSL DSP Graph Flexibility

- Sub-graphs can run at different:
  - Sample rates (for control),
  - Buffer sizes (for transforms),
  - Numbers of channels (for efficiency),
  - Buffer formats (interleaved or not),
  - In different threads, etc.
- These can be changed (within reason) at run-time (e.g., for load- or traffic-balancing)

Multi-host CSL Graphs

Demo

- Distributed sub-graph processing with RemoteIO and RemoteFrameStream, RFS protocol, buffering



RemoteFrameStream Details

- Uses simple protocol, LAN/WAN-oriented (we use switched 1000BaseT & TCP)
- Relatively careful (packet header/trailer, sequence numbers, format packets)
- Double-send optional with UDP/ATM
- RFS uses ThreadedFrameStream with variable-sized (zero-possible) RingBuffer

Control, Latency, Scheduling

- All CSL processing is triggered by output requests (pull model, buffer size = control rate)
- Slow computations should use ThreadedFrame-Stream or transform/convolver threads
- Control may change asynchronously; query is_processing() optional (semantics of control)
- Latency determined by buffer size, amount of cacheing in graphs, remote links (few msec for small buffers, <1 msec doable [?])
- Dynamic graphs are rare; no time or event models

Instruments and OSC/MIDI/XML

- Instrument object
  - Holds onto a DSP graph; adds "reflective" accessors
  - Generates OSC address spaces, MIDI maps, etc.
  - Server `main()` function loads an instrument library and publishes an address space on a listener socket
  - Example: // C++ accessor decl.

```
list[0] = new Accessor("du", set_duration_f, CSL_FLOAT_TYPE);
list[1] = new Accessor("am", set_amplitude_f, CSL_FLOAT_TYPE);

-->         /i1/        instrument 1's OSC address space
             /i1/du:  set-duration command
             /i1/am: set-amplitude command
```

CREATE MAT Media Arts and Technology Graduate Program

CSL Portability

- MacOSX, *nix, Linux, MS-Windows (gcc relatives)
- Cross-platform APIs
  - PortAudio for RT sound IO†
  - LibSndFile for sound file IO
  - PortMIDI for MIDI†
  - LibNewRan for probability distributions
  - FFTW for FFT†
  - CyberX3D for VRML, OpenGL†
- Issues
  - C++ compiler, socket/thread code, GUI
  - Base sample data type (float vs int)

† = may use platform-specific APIs (CoreAudio, DSP_FFT, etc.)

CREATE MAT Media Arts and Technology Graduate Program

Using CSL

- As a library
  - Link a graph and IO into your application, game, GUI, etc.
- For plug-ins
  - AudioUnits or VST with GUIs; call-back to `next_buffer()`
- For OSC, MIDI, CORBA, XML-RPC, etc. servers
  - Stand-alone instrument groups as soft-synths; RemoteIO
- With CRAM
  - Multi-host control/server/output configurations
- The `main()` function creates graph or mixer, may spawn threads, then registers an IO call-back object

CREATE MAT Media Arts and Technology Graduate Program

CSL "beep" main (all of it!)

Demo

```
//   Beep_main.cpp -- the simplest CSL "main" program -- a 3-second beep
#include "CSL_All.h"                  // CSL "kitchen sink" include
using namespace csl;                  // Use C++ CSL namespace
                                      // MAIN -- plays a 3-second beep
int main (int argc, const char * argv[]) {
    PAIO gIO;                         // PortAudio IO object
//  FileIO gIO("csl_test.aiff");      // File IO object
    Sine osc(220);                    // create a sine oscillator at 220 Hz
    gIO.set_root(osc);                // plug it in to the IO
    gIO.open(); gIO.start();          // open/start the IO
    sleep_sec(3);                     // sleep 3 seconds (CSL blt-in fcn)
    gIO.stop(); gIO.close();          // stop/close the IO
    return(0);                        // exit
}
```

CREATE MAT Media Arts and Technology Graduate Program

CSL Example: Se/Sp_Sp (2002)

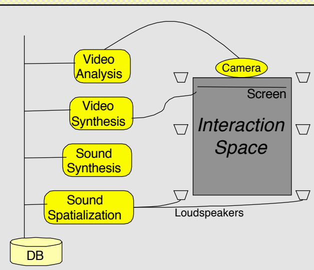
Sensing, computation, (m-) presentation (MVC)

Camera-based multi-user sensing (aware space)

Computer vision SW follows mvmnt & grouping among attendees; sends OSC msgs to 3 servers

Synchronized multi-camera projection and 6-ch. surround sound

Port from SC2 to CSL2

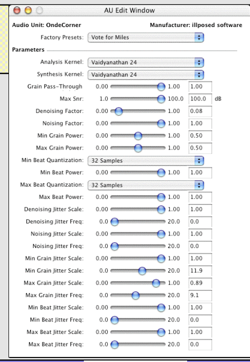


CREATE MAT Media Arts and Technology Graduate Program

Sensing/Speaking Space @ SFMOMA Feb. '02

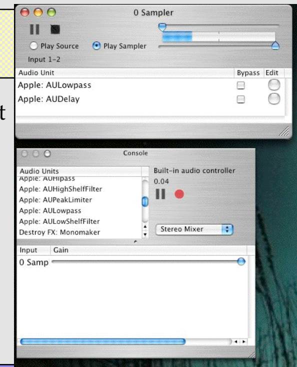
Example: OnDeCorner

- CR's AudioUnit plug-in for experimenting with wavelet transforms
- Pluggable FWT code
- Play to DAC or file



CREATE MAT

Example: Ouroboros



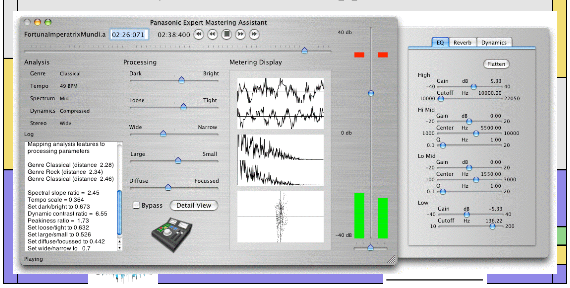
- CR's AudioUnit host application for processing sound files and live input
- Extensions planned for remote AudioUnits

CREATE MAT

Example: Expert Mastering Assistant

Demo

Process: Analysis, GenreDB, Mapper, DSP, Interact



Generating CSL Graphs/Events

- Using scripting languages
- Smalltalk Slang translator
- From XML
- DragNDrop "patcher" GUIs
- Storing signals and graphs in an OODB
- Instrument libraries and event stores
- Auto-gen of flat namespace for C RMI

CREATE MAT

Example: LUA Patcher (worked, but failed)

CSL as a library for a scripting language

```
-- Lua program for a panning chaotic oscillator
panning_chaos = function {}
     lorenz = Lorenz{};
     envargs = {0.5, 0.0, 0.0, 0.003, 0.5, 0.5, 0.0};
     envelope = Envelope{envargs};
     panner = Panner2{lorenz, envelope};
     audio_out{panner};
end
```

CREATE MAT

So we know it all, right?

- NOT!
- Many open architecture, design, modeling, implementation, deployment, issues
- Some basic choices we're still debating
- Some real dilemmas, limitations, principles
- Tensions between our design bias towards simplicity and "creeping featurism"

CREATE MAT

Open CSL Design Issues

- Basic models: buffer-based, event-based, signal-based
  - Current pull-model driven by PortAudio and CoreAudi APIs; granularity of events
  - Need a unification of types (semantics) of buffers (samples, FFT frames, FWT frames, IRs, etc,)
  - Signal semantics: operators on buffers vs. procedural ugens?
- How to support dynamic graphs in a simple system (punt)
- That latency thing, polynomial ctrl interpolation, clock sync.

Speed Hacks & Optimizations

- User-visible optimizations
  - is_fixed_over(), is_active() -- used
  - is_linear_over(), is_polynomial_over() -- ?
- Several kinds of buffers (cache optim.)
- Control interpolation?
- DSP graph-to-SMP allocation
- Managed sample-rate conversion
- Better C++ compiler (IBM or Intel/AMD)
- Many interesting optimizations would greatly complicate the system (our guess)

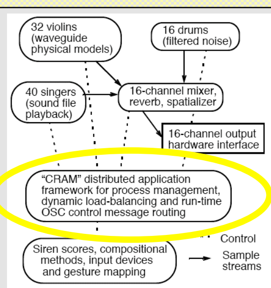
CSL Wish List

- Better IDE/CVS/GUI (Smalltalk)
- Better base libraries (Smalltalk)
- Better MM libraries (Java, CommonLISP)
- Untyped language (SuperCollider)
  - ST Slang + operators?
  - Scripting language bindings?
  - Java with OO RMI?
- Language with GC (SC3, Smalltalk)?
- Otherwise, just like CSL3!

Next Steps for CSL

- Unified buffer class (cache usage)
- New synthesis techniques: MAT 240D '03: design issues raised
- Tighter integration with CRAM planner, monitoring, IO/DB managers
- Siren OSC routing in CSL server farms
- MAT 240F going on now!

Part 3: The CRAM DPE



Managing Siren and CSL: CRAM

- CRAM: Yet another **Distributed Processing Environment** (DPE, Cluster Mgmnt. literature)
- Framework to deploy, start/stop, and monitor multi-host distributed real-time OO applications
- Provides fault-tolerance and load-balancing*
- CRAM is 3rd-gen. DPE implementation at CREATE (1996-2004) (HPDM/TAO, Yellow/CORBA_AV)
- Designed for robustness, simplicity, and low overhead; limited services and scalability/replication

CRAM DPE Operation Model

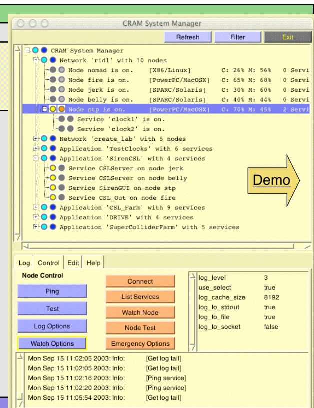
- **Applications** are composed of services
  - which probably talk among themselves apart from the management system
- Any program can be a service (lightweight wrapper)
- **Nodes** (HW) can run services
  - and monitor their hardware and services
- The **system manager** talks to a node and its services
  - Start/stop, load-balance, monitoring, fault-detection/restart, etc.

Components of a DPE

- Node manager program (Node)
  - Daemon that provides "remote exec" functionality and status/performance monitoring; runs on all machines
- Service interface (Service)
  - Wrapper code that is added to all application server programs managed by the DPE
  - Adds socket listener threads for mgr messages
  - May add init/restart functions between server obj. & app.
  - May integrate logging, status monitoring into app. code
- System manager (Manager)
  - Talks to nodes to administer services for distr. apps
  - Uses DBs for network, services, and app. configurations
  - May offer an expert/constrained configuration planner

CRAM Manager

- Network/Node
- Node/Service
- Application/Service
- Log/Control pane
  - Run-time monitor
  - Planning
  - DB play-back



Using CRAM

- Write a Service class for the app
- It handles main() (creates the service object) and start/stop messages (start/stop the base app)
- (Optional) add run-time option-setting, logging, app. status monitoring (handle CRAM msgs)
- Create a service type record in the DB
- Add applications to the DB
- Go!

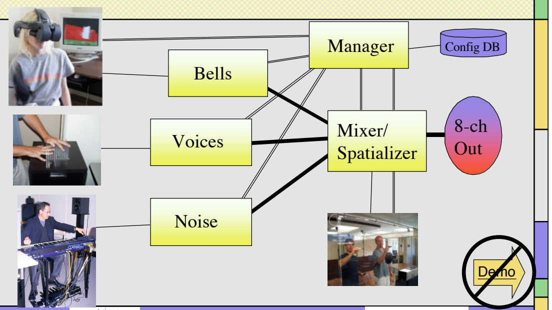
CRAM Databases (Simple RDBMS)

- Networks
  - Nodes and their properties
    - HW, OS, MIPS/MFLOPS, LAN, special I/O
- ServiceTypes
  - Name, arguments, options
  - HW/OS/IO requirements (for configuration)
- Services
  - Type and actual run-time arguments (net IO)
- Applications & History
  - Lists of services (& their options) on nodes

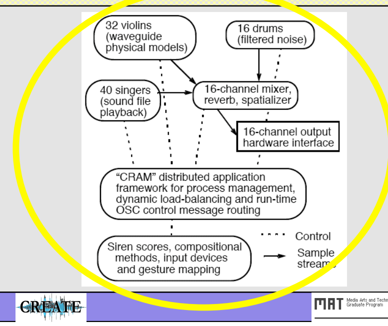
CRAM Implementation Details

- Node/Service source = 4kLOC C++, 2.5 kLOC Java, 0.8 kLOC ST80 (w. DB-IO)
- Manager & DB source = 4kLOC Smalltalk [~50% auto-gen]
- Uses low-level UDP/TCP protocols
- Several levels of failure recovery, node-service-discovery, heart-beat monitors
- Logging and monitoring to DB, replay

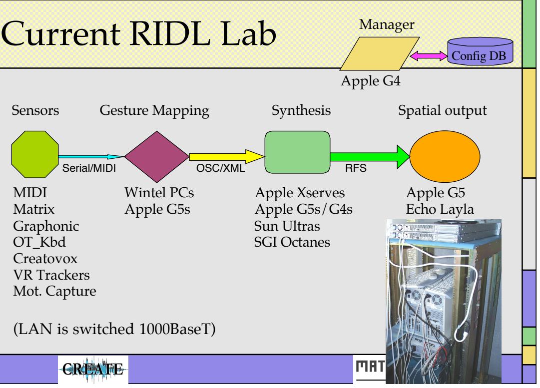
CRAM Configuration for CSL



Part 4: Putting it All Together



Current RIDL Lab

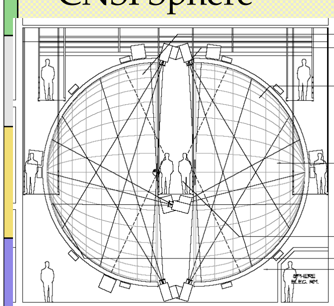


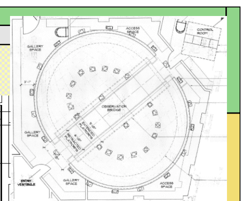
(LAN is switched 1000BaseT)

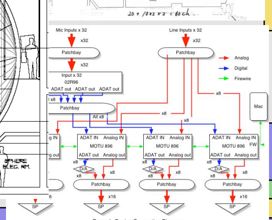
Next Gen Lab: CNSI

- MAT in CNSI: labs, studios, workshops, sphere
- CNSI compute infrastructure
  - Traditional vector supercomputer
  - 1024-node Linux cluster
  - Multimedia processing cluster (TBD)
- Sphere (3-story) I/O space
  - 12-channel overlapping video output
  - 128-channel sound output
  - Camera-based and (multi) other input sensing

CNSI Sphere







How? DSCP!

Distributed Sensing, Computation, and Projection = *MVC on steroids*

Back-end application models are scientific/numerical/simulation

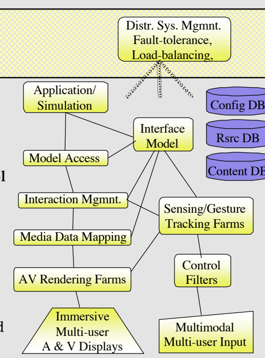
Multimodal multiuser **sensing/control** and tracking/mapping farms

Application = sensing/tracking policies + output data mappings

Presentation/interaction via CNSI Sphere, LAN/WAN streaming

Infrastructure uses CRAM mgmnt

DBs for configurations, resources, and media content (renderers)



Challenges to CRAM & DSCP

- True fault tolerance with off-the-shelf HW/OS
  - Robust node manager (emergency thread monitor/restart), TAO, SqVM
  - Debugging (LAN) network problems
- The plethora of protocols, IPv6, ATM, FireWire
- Performance and scalability issues
  - Node monitoring (<u>top</u>-ing)
  - Logging (capture)
  - Service creation, sync, init, restart

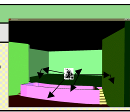
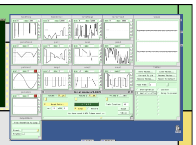
CREATE MAT Media Arts and Technology Graduate Program

Conclusions: Siren, CSL, and CRAM for DSCP

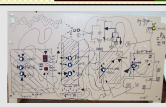
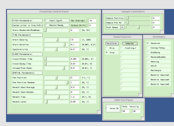
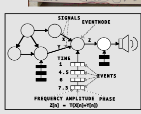
- For our requirements, we really had to start from scratch for most of the components.
- The KISS principle (or XP) paid off in simplicity, flexibility, and ease of use.
- There are many things we could have done other ways (we're still debating; that's the whole fun of it!).
- <u>We'll be installing DSCP in CNSI in 2005/6!</u>
- See create.ucsb.edu/{Siren, CSL, CRAM}

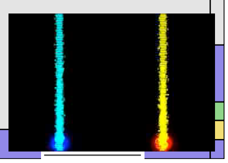
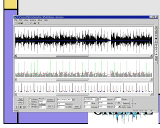
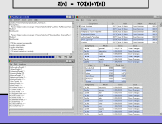
CREATE MAT Media Arts and Technology Graduate Program

Related Projects at CREATE

- Auralizer & VRML
- Pulsar Generator
- Creatovox
- MusicVisualization
- FMAK DB
- TimeMachine
- InteractEMGroup
- Creatophone
- Time-DDecomp
- SC_3 Work

Recent Compositions (2000-03)

- *— Sensing/Speaking Space (pt. 2)*
  Interactivity, speech DB, spatialization
- *— Four Magic Sentences*
- *— Gates Still Open (2nd development)*
  Phrase grammars, DSP
- *— Eternal Dream (finale)*
  Large-scale form, speech DB, DSP
- *— Leur Songe de la Paix (1st/2nd mvmnt.)*
  DSP, structure-extraction, speech DB

CREATE MAT Media Arts and Technology Graduate Program

Demonstrations

- **Siren** models, GUI, sound tools, browsers, music representation, composition code
- **CSL** examples: instruments, client/server
- **EMA** mastering application, DB/DSP
- **CRAM** DPE manager and tools

CREATE MAT Media Arts and Technology Graduate Program

Thank You ! (Q & A)