# TML Wiki

–Table of Contents



# Ozone

Ozone is the current incarnation of a media choreography system designed by Sha Xin Wei for the dynamical shaping of a responsive media environment according to the activities of its inhabitants and the prior intent of its composers. If you are an Ozone creator, please read the TML WIKI entries in the Media Choreography category, especially the internal technical reports on the physics and engineering design specs behind Eerm (Sha & Visell, sponge, 2001), tgvu-code (2001-2004), txOom (FoAM, 2002), and applications such as MeteorShower (2007) and the software part of Remedios Terrarium (2008).

The current code can be found in an SVN repository at http://tml.ath.cx:8080/svns/MaxLibraries/TML/. The version of the system that is currently running in the lab can be found in /TML/pro/lab. The current version of Oz, the part of the system that handles sensor data acquisition, feature extraction, fusion, and media choreography can be found in /TML/oz. A working set of video-instruments can be found in /TML/ozone. The current sound-instruments can be found in TML/pro /lab/snd.

This page is usually out of date.

# People

- Sha Xin Wei, system architecture, experiment design, media choreography
- Harry Smoak, media choreography, lighting, experiment design
- Michael Fortin, computational fluid dynamics and video
- Morgan Sutherland, state engine, sensor fusion, media choreography, project management
- Navid Navab, realtime sound
- Julian Stein, realtime sound, camera processing
- Omar Faleh, dynamic lighting
- Tyr Umbach, visual programming, state engine

# Previous

- Tim Sutton, realtime sound
- Jean-Sebastien Rousseau, realtime video
- Delphine Nain, computational fluid dynamics and video
- Yon Visell, Emmanuel Thivierge, state engine

# Network

## Protocols

- OSC (UDP)
- Jitter Matrices (TCP)

## Machines

- Derrida (IP: **10.0.0.1**) [Il Y A]
- Deleuze (IP: **10.0.0.10**) [Calligraphic Video] – **currently sick**!
- Peirce (IP: **10.0.0.11**) [Lighting]
- Langer (IP: **10.0.0.12**) [Secondary Video]
- Guattari (IP: **10.0.0.13**) [Responsive Sound, state engine]
- DMX interface (IP: **10.0.0.77**) [(will be) Enttec NMU]
- Kinect (IP: **10.0.0.91**) [Camera acquisition]

## OSC Routing

[Not currently in use October 2010] For making available data in various formats using OSC, Michael F. has objects in the svn/tml/net folder. Of interest is:

- tml.core
- tml.log
- tml.networkHelper
- tml.finder
- tml.networkResource
- tml.networkFetch

The best place to get an overview of how they all work is to do a help on tml.networkResource. (probably we should rename this to a patch like tml.networkOverview, even if other help files just link to it)

Basic steps:

1. Create one instance of tml.core somewhere in the networking section of a system. Only one instance required - take care to not have more than one (you just wast CPU power otherwise)!
2. Create one instance of tml.networkHelper. Put your host's internal IP (ie. the local Ozone 10.0.0.x range) in the left inlet, the router's broadcast address in the right (10.0.0.5). That's it. For a basic interface, Michael made a gui to set the IP addresses, but this isn't necessary. It is no longer necessary to loadbang the ips into networkSerttings as it uses pattr to save them.
3. To query what's available on the network, open tml.finder and bang it, and look at the results out the left outlet. You can drill down the namespace with standard OSC slash syntax as an argument to tml.finder, but it isn't necessary. Specifying a namespace limits the number of results returned (specify nothing and all available resources are queried).
4. Use tml.networkResource to register a namespace with data available. Should look like this: “tml.networkResource /ozone/this/is/my/namespace f 0.”, which denotes the namespace, the type of data expected, and a value placeholder. The value placeholder doesn't yet do anything, but it could later..

5. Use tml.networkFetch to subscribe to a service, with the full OSC namespace as the only argument. There is a stay-alive argument as part of the protocol - so if a networkFetch goes offline, the stream will stop sending data to it. (networkFetch takes care of the stay-alive ping)

That's it. There might be a little bug with it skipping the head node of the namespace (ie. /ozone).. we'll have to take a look (but it seems to be fixed at the moment).

# Video

Deleuze performs responsive video synthesis. Tyr is in charge of this. teumbach@gmail.com or electioneering@gmail.com

# Sound

CHMM-03 performs responsive sound processing and synthesis. Navid is in charge of this.

(Old note from Tim: Currently an overview of the sound system (signal flow and software details) is available in svn/tml /snd/doc. It's current, but needs to be updated wrt to the main sound instruments, sensing mechanisms and OSC schema. For interests' sake, this is a link to the first Meteor Shower incarnation documentation.)

# Oz

Oz is the backbone of Ozone. It extracts features from sensor data, fuses that data into meaningful bundles, maps it, and choreographs the other instruments using the state engine 'Eerm'.

## State Engine

The state engine uses sensor date to compute a 'state' for the system, which consists of the location of tokens {t1,t2,…,tn} in simplicial complexes {s1, s2,…,sm}. The vertex of each simplex (n-dimensional triangle) corresponds to a pure state, given a metaphorical meaning by the state composer. The simplicial complexes are composed to represent a pre-defined narrative structure. The movement of the tokens tn inside the “state space” formed by the simplicial complexes constitutes movement through the narrative.

### Current Lab Topology

As of October 2010, the state topology will have 2 states. Metaphorical meaning of the states yet to be decided upon.

~~Currently implemented, as of early March, 2009, is a three state topology with states **Calm**, **Storm**, and **Desert**. * **Calm** is activated when people are moving close to the screen. * **Storm** is activated when people are moving in the center of the room, near the screen, but not too close to it. * **Desert** is activated when people are moving near the door or in the “dev” and “workspace” areas. Phenomenally, you should experience the room *activating* as you walk into the center. As you walk close to the screen, the visuals should freeze. As you settle down in the “dev”/“workspace” area, the room should be as it was before you entered: some variety of calm distinct from that of the **Calm** state.~~

### State API

- /tml/ozone/state/[state_name] f
- /tml/ozone/state/vector f1 f2 … fn

## Sensors

The lab is outfitted with the following sensors as of January 2011:

- 1x Colour Analog camera, zoom lens (positioned on the grid in the front, near the door, pointing towards the centre

of the room)
- 1x Analog camera, wide-angle lens (positioned on the grid in the middle of the south wall)
- 1x Analog camera, lens (positioned on the grid over the white table)
- 1x Analog camera, zoom lens (positioned on the west wall pointing toward the middle of the floor)

The analog sensor streams are digitized in the following manner:

- **Camera feeds** are captured to Pierce with 2 firewire digitizers. The feeds are made available over the network via OSC. Additionally, certain features are computed on the capture machine. Michael has written a new network protocol. More information to follow.
- **Microphone feeds** are captured to CHMM-03 with the M-Audio firewire audio interface.

## Sensor API

- /tml/ozone/sensor/camera/[ID]
- /tml/ozone/sensor/mic/[ID]

## Software Architecture

See diagram in ACM Multimedia article 2009.

ozone_system.pdf

projects/ozone.txt · Last modified: 2012/07/25 09:10 by xinwei

---