SC is a modular suite of software designed to allow designers to compose the behavior of a responsive media environment evolving in concert with contingent activity in a physical space. The media can be rich and fairly eccentric: projected video, spatialized audio, theatrical lighting — generally fields of structured time-varying light and sound, as well as water, mist, animated objects etc. The behavior of the responsive environment evolves according to prior design as well as contingent activity. A key condition is that everything happens in real-time, in concert with the activity of the inhabitants of the responsive environment. SC supports rich and thick experiences with poetic, symbolic, and scientific effects.
Introduction

A responsive environment is a physical space in which the media — light, sound, or some other material — can vary in concert with the activity of people and things inside the common space. A key condition for composing these environments lies in the leverage of the affordances of analog media and the relations among living bodies in a live event. Responsive media are fields of structured light, sound, or some other computationally-modulated distribution of matter or energy that vary accordingly to movement or gesture. Typically, responsive media are continuous in time and space from the point of view of human. Their palpable, real-time, continuous coupling to movement affords rich, painterly, calligraphic, gestural expression with arbitrary, improvisatory nuance; discrete change, modulated by Boolean procedural logic, is a special case.

To facilitate the creation of responsive media environments, we have written a structured set of software abstractions — SC — that simplify the physical sensing of environments and control of media instruments within them. It is intended to allow modular design that enables designers to rapidly pass signals between sensors, data transformations, and media; manipulation and production of dense, continuous media that produce a rich palette of possible media states; continuous evolution of media that can facilitate production of dynamic, responsive, rather than static or repetitive media environments; and designing at the level of metaphor that focuses the attention of developers on the intended effect of media states rather than underlying mathematical representations.

SC is designed to be used by event composers or installation artists who are not software engineers. The scripting language Max/MSP/Jitter, the lingua franca for live media scripting, permits the blending of decades of toolkits for live media processing from developers worldwide, such as cv.jit computer vision; MuBu multibuffer audio signal processing, SPAT spatialized audio; Wekinator machine learning.

We have used SC and its precursors in various installations, media sculptures, and events. For a sampling see the following videos: immersive enactive cloud simulation (vimeo.com/synthesiscenter/clouds), or day in the life (vimeo.com/synthesiscenter/slsa2).
The SC Responsive Media Library

The SC software frameworks are built around *media instruments*: code that transforms a media stream (typically sound, video, or light) into another media stream at real-time rates under the threshold of perceptible latency.

![Image 2: A media instrument transforms a stream of media into another stream in real-time, modulated by public parameters.](image)

Media instruments are redefined continuously by parameters that are functions of features derived from sensor inputs, internal conditions, or the metaphorical state of the event (see *Image 2*). In a typical responsive environment, a suite of instruments process the time-based field media by mapping streams to streams as illustrated in *Images 3 and 4*.

Within the SC framework, nearly 300 utilities and instruments have been created to compose responsive environments, consisting of physical sensing and actuators, acoustic sensing and multi-channel audio, lighting arrays, and streaming video capture and visual projections. SC supports working between real-time media formats by using a standardized modular architecture whereby messages, including real-time streams of manipulated media as well as statistical calculations are passed between components. In this way, media can be easily interchanged or transcoded. To support this style of development, SC has been implemented in the real-time media programming environment Max/MSP/Jitter to make all layers of control legible in a common scripting language accessible to composers who are not software engineers and wish to work across all computational media types. In the following sections we will describe a few example objects from each category: audio, video, lighting, physical, and intermediate complex systems. In addition to the media instruments, SC provides utilities for manipulating data streams, including objects for scaling, easing, mapping, interpolating, and ramping signals as well tools for event detection, time-series analysis, and tools for producing sequences of data, such as Markov random processes and physical models.
Audio Instruments

SC’s extensive collection of audio utilities includes several modules for audio effects and filtering, and reverberation effects to simulate different acoustic environments, granular synthesis, synthesis of parameter trajectories and sequences for pitches and frequency bands, a multi-band filterbank instrument that allows movement within a dense stream of video to manipulate audio streams, and several utilities for producing audio in different multi-channel setups aided by IRCAM’s Spat audio spatialization package.4

Video instruments

The collection of video utilities and instruments encompass everything from grabbing video from different sources to computer vision algorithms for feature extraction and post-processing of graphical output. SC has abstractions that wrap methods of reading data from cameras, files, and the Syphon Framework. A cornerstone of the SC package is the array of computer vision abstractions. These patches cover methods of background subtraction, edge detection, optical flow, and presence detection. All of these techniques have been built as GLSL fragment shaders to allow for higher resolution tracking while still increasing performance in comparison to traditional CPU-based versions.

In addition to methods of quantifying video inputs, SC is also capable of visualizing the results from complex system simulations. Two notable methods

---


5 Syphon, by Tom Butterworth & Anton Marini http://syphon.v002.info/
are through the use of particle systems and vector fields. The particles can either act as tracers, being guided by the output field of a complex system, or behave as agents, feeding data back into the system that is moving them. Vector fields reveal trends of change within a complex system’s output by drawing poles pointing in the direction of the current delta.

SC also provides methods of modifying visualizations through post-processing effects. The list includes blooms, blurs, palette-based recoloring, and temporal shifts. Most notably, the “time-space” effect produces an image consisting of delays on a per-pixel basis. The system stores incoming frames of video into a buffer, and uses a mask to determine where in the buffer to sample for each individual pixel. This temporal processing method allows for a visual history of any video stream and also opens up opportunities for artistic use of delays.

Lighting instruments

SC’s lighting abstractions rationalize accessing and controlling DMX lights or other hardware via a common messaging protocol. Pre-built abstractions handle everything from scaling incoming data along the response curve of individual lights to communicating the correct DMX address with the chosen interface. This allows composers to blend in a state-of-the-art professional theatrical lighting to produce the highest quality visual qualia.

Physical sensors and actuators

The library of physical sensing patches supplies methods of reading and parsing data from different sensor configurations and controlling physical media. It includes modules for communicating with a network of WiFi-enabled development boards that control arrays of ultrasonic atomizers, interface with a custom bend-sensing glove interface, and parse data from x-io Technologies XOSC I/O board, which transmits IMU data and the state of up to ten additional sensors wirelessly via UDP bundled in the Open Sound Control protocol.^[6][7]
Complex systems simulations

To produce rich behaviors governed by dense, complex dynamical processes, a core component of the SC system is a set of simulations for complex dynamical systems models of physical material systems, ranging from Navier-Stokes and lattice Boltzmann fluid dynamics, Gray-Scott reaction-diffusion chemical systems, to more specific systems, such as a charged body simulation (by Rawls) and our Experiential Model of the Atmosphere.

In the charged body simulation, gradients of electrical charge can be used to manipulate video. Bodies are assigned initial positive or negative charges upon entering a physical space, which can be controlled by the designer. As objects or bodies move within the video, their charges interact based upon the values held by neighboring objects. The resulting charge gradients can be used to assign properties to different pixels within a video stream. A previous use of this system determined how attractive or repulsive an object was to virtual particles within a fluid simulation.

EMA is a real-time, steerable simulation of warm cloud physics that allows media developers to move between virtual states of dry air, water vapor, and condensed liquid water under the influence of atmospheric dynamics.

The SC state engine: continuously evolving media

To produce continuously evolving behaviors within responsive environments using the SC instruments, we have created a state engine utility that allows designers to determine how the environment will evolve in response to the sensed state of the environment and current state of the media instruments or underlying simulations. The purpose is to meaningfully evolve the behavior of the entire ensemble of rich media instruments suggested by a continuous model of state evolution. We use a continuous dynamical system in place of boolean, procedural, or stochastic (probabilistic) logics that are commonly implemented using finite state machines to provide rich behavior that responds to arbitrary fine nuance, and evolves robustly to arbitrary, even unanticipated, activity.

This continuous state evolution model is not a finite state machine because the formalism admits continuous ranges of change and unbounded continua.

13 We exploit Max’s powerful pattr abstractions for saving and restoring the settings of any object’s parameters, and have implemented a design pattern for application programming that leverages either pattr or odot.

of possible states: a state can be the formal combination of any number of ingredient states. The state engine does not describe the state of component code or physical devices, which we call parameterization or presets. Rather, in our terminology, a state refers to a metaphorical description of the event as experienced by the inhabitants in the environment during a live event. For example the state of an event could be characterized by terms such as “the beginning,” “nighttime dormancy,” “people are bored,” or “stormy,” which are nominally associated by the composer to combinations of features that can be derived from sensors (cameras, microphones, photocells, piezoelectric sensors, etc.). This interpolation of a state evolution layer in between the sensor data and the parameters controlling the software/hardware media instruments allows the composer to design rich behavior while at the same time freeing her/him from locking that behavior design into particular technology or particular technical instantiations.

It is also important to emphasize that the behavior is neither a fixed sequence (fixed tree of locally determined linear sequences of action) nor random (stochastic). More profoundly, the designer does not determine what actually happens, but rather the way the environment will tend to evolve in response to any activity. Thus the state evolution acts on the space of potential, not actual activity. Nonetheless, the designer can condition the behavior as precisely and narrowly as desired. In practice, this system is best for medium to coarse qualitative changes in the behavior of arbitrarily rich complex environments, whereas specific action-response logics can be written using conventional ad-hoc code.

In practice, SC’s state engine is implemented as an interface where designers can a) define a number of states, b) assign nominal sensor values to each state, and c) bind parameters of the media instruments to these states. The designers can then arrange the states in a simplicial complex, allowing for both linear movement between “fundamental” states and more complex regions in the potential state space where the current state may be a combination of many fundamental states. (There is nothing sacred about which states are fundamental and which are (convex) sums of fundamental states.)

The system’s state evolves as a function of both the incoming sensor features and an intrinsic dynamic based on minimizing an energy functional.
The vector distance of current sensor data from the nominal sensor vectors assigned to each fundamental state is computed. It contributes to the energy of the current state of the environment. The physical model then evolves so as to minimize this energy by adjusting the position of the player state. In addition, several parameters allow designers to give each state a certain amount of static energy to fine-tune the contribution of the state to the movement of the player state. An example state topology leading an environment through different times of day and seasons is shown in *Image 5*.

In this topology, the state of the media environment can move between several metaphorical states associated with different seasons of the year and times of day. Within each season, the system state can evolve continuously between the different times of day, but significant seasonal state transitions will only occur overnight. In this image, the media environment is experiencing an overnight transition between summer and fall. If this topology were to drive EMA, for example, each state could be associated with different physical variables, such as average ground temperature or humidity, associated with different seasonal climates. The state engine is implemented in a graphical interface that makes it easy for designers to arrange the state topology, name states, and assign sensor data values and media parameters to each state.

**Conclusion**

We have outlined a software framework usable to designers to compose behaviors for responsive environments. The system’s key features include:

1. Rich robust evolution of behavior in concert with arbitrary, improvised activity by inhabitants;
2. Evolution of state based on continuous dynamical system rather than discrete finite state model;
3. Real-time media stream processing instruments created by time-based media artists for aesthetic and experiential impact;
4. Implementation in lingua franca for real-time media processing;
5. Transcoding.
Synthesis at Arizona State University provides a place for experimentally inventing and fusing fresh practices of understanding and enriching how the world works with fresh practices of making meaning. Sha Xin Wei founded this atelier for fusion research in 2014 with colleague research faculty Todd Ingalls and Brandon Mechtley to extend and expand on 15 years of prior work in the domains of responsive environments and live event / movement-based research creation at the Topological Media Lab / Concordia / Montreal, and at Arts, Media + Engineering / ASU.