Extended Abstract
A Ubiquitous Computing Framework for Authoring Interactive Environments in the Arts

Eitan Mendelowitz
Department of Computer Science
Smith College
Nothampton, MA

Jeff Burke
Center for Research in Engineering,
Media, and Performance
University of California, Los Angeles
Los Angeles, CA

1 Introduction

In the media arts, theater and dance there is increasing interest in creating artworks that combine sensing with realtime media control to create interactive experiences. Artists often couple digital projection, environmental lighting control, real-time music/sound generation, animatronics, and other media elements with sensors such as motion detectors, accelerometers, and vision systems to create compelling, immersive, and reactive physical environments. These technologies allow artists to create systems of relationships among and between sensed events in the physical world, purely virtual components, and digitally controlled media elements to engender new aesthetic experiences.

Artists working with these technologies are adopting distributed computing for many of the same reasons others turn to networks, including flexibility and scalability. Distributed computing allows artists, often dealing with very limited resources, to take advantage of now common economies of scale.

However, artists new to working with distributed technologies for physically interactive digital media face significant technical hurdles that prove time consuming even for the more experienced. The initial step of interconnecting devices can be labor intensive while having very little relevance to the final experience of the work itself. Developers are often forced to create (or recreate) methods for making connections between devices and the authoring platforms they may use for a given piece.

2 Related Work

There is a great deal of research which focuses on the development of frameworks and programming languages in support of applications using distributed sensing and actuation. Such environments provide the application-developer tools to deal with heterogeneous sensors, distributed resource allocation, and distribution and aggregation of data, state, and context.

Sensor network research focuses on creating applications using distributed sensors while ubiquitous computing frameworks often focus on facilitating application creation through the collection, representation, and processing of distributed data.

3 The Needs of Artists

While work in sensor networks and ubicomp contributes ideas and techniques useful for the creation of distributed applications in the arts, it is rare to see these frameworks used by artists. Artists typically create their artwork using a variety of specialised software and hardware systems networked together in an ad-hoc manner.

It is quite common for artists to create one program for sound in Max, one program in Processing (or Flash) for visuals, and a number of smaller programmes for sensing and application logic. These programs are often running on multiple machines due to computational costs or I/O bottlenecks and are often interconnected over the network using one-off protocols over UDP.

So why don't artist use one of the existing solutions available from the sensor network or ubiquitous computing communities? It is because existing frameworks were created with different application domains in mind and fail to meet the needs of artists.

4 Framework Requirements in Support of the Arts

Because the arts are such an under-examined application domain the technical requirements needed to support such work is hard to determine. The fol-

lowing design requirements derive from a broad examination of existing works, informal conversations with digital artists, the results of the NSF sponsored a workshop on *Creativity Support Tools. Design Principles for Tools to Support Creative Thinking*, and personal and professional experience in creating physically interactive art installations.

In order to support the authoring of physically interactive environments in the arts, a framework must support: (1) low latency networking, (2) distributed state management and synchronization, (3) extensibility, (4) integration with existing tools used in the arts, (5) experimentation, and (6) appropriate levels of abstraction.

While many techniques pioneered in sensor network research can inform the creation of an ubiquitous computing framework the systems themselves are focused on minimizing computation, network, and energy usage. These optimisations are often made at the expense of data throughput and latency thus disqualifying them from use in most physically interactive applications.

The service, context, and event based data abstractions used by many ubiComp frameworks like Giai and PatchPanel often seek to abstract away details important to artists and incur network latency, while the data-flow approaches of NDFS and Aria hinder run-time manipulation of networks, experimentation, and extensibility.

5 Authoring with Kolo and Nebesko

Kolo provides a high level Java API which allows application authors to interconnect and reuse supported devices with the same interface regardless of whether they are running on a local or networked Kolo process. This same API is simple to extend allowing developers to easily incorporate new sensors and media elements in a consistent framework for control data transport. To promote quick adoption by authors, Kolo's API is designed with relatively few core abstractions –especially when compared to traditional distributed frameworks. Kolo only requires the understanding of six concepts: knobs, values, subscriptions, groups, relationships, and arbitrators.

Because Kolo is designed to incorporate platform-specific and hardware-specific device drivers, a C/C++ Java Native Interface (JNI) is provided. Using this JNI, a wrapper for the Kolo API is available in Macromedia's Director via an "xtra". The Java API can also be used in Processing and in Cycling '74 Max/MSP through a set of Max objects. In addition to the Java API, Kolo applications can be authored using Nebesko, a scripting language written specifically for creating ubiquitous computing applications using Kolo.

While Kolo addresses the difficult problems implementing and interconnecting distributed objects, Nebesko addresses the problems of monitoring, reacting to, and coordinating distributed state by providing a finite state machine that natively supports state transitions based on distributed object state, their interconnections, and time. Perhaps the most important feature of Nebesko is that knobs are treated as first class citizens. Getting or setting the value of a knob is no different than getting or setting the value of a local variable. Subscriptions between distributed objects are no harder to create than variable assignments and creating a relationship is no different than writing an expression.

6 Results

Kolo and Nebesko met the design requirements outline in section 4. These requirements include the quantitative requirement of "low latency networking" which implicitly includes reasonable network utilization. The design requirments also provides more qualitative conditions including appropriate abstractions for state management and synchronization, extensibility, integration with existing tools, and support for experimentation. To evaluate these measures we present a case study, Kolo's use in "Ecce Homology," Kolo's first full scale deployment in the arts.

7 Discussion

Kolo needs an active community of artists and developers. In creating artwork, artists drive the demand for new sensors and actuators. In extending Kolo to work with new devices, developers create new possibilities for expression. It is our hope that participants in the "12th Biennial Symposium on Arts and Technology" will help form a community of Kolo/Nebesko authors and developers.