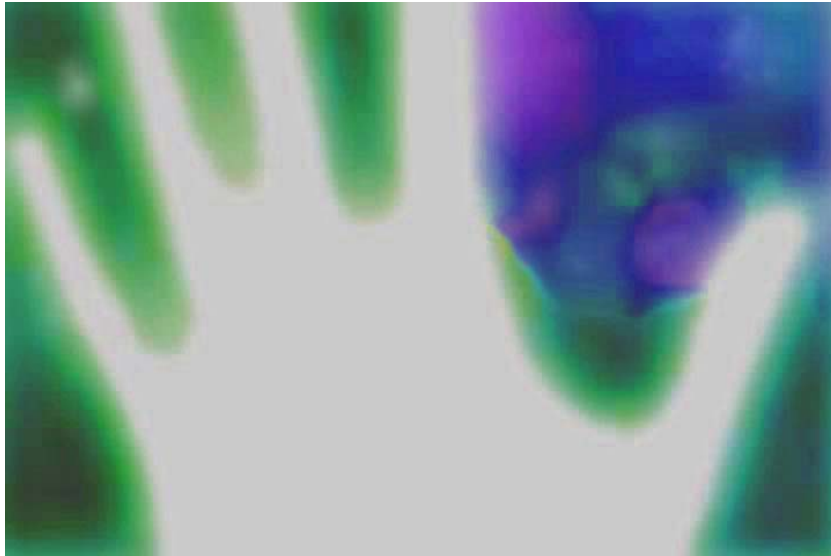


Luminance

Interactive Art Experience



A Fall Quarter Update for
CSU Hayward Multimedia Graduate Program

Dec.7, 2004

shortattentionspan

▼ | ●
J.J. Scott Lee

<http://sasweb.csuhayward.edu>

The Research Question

Can a new methodology be developed to create complex, playful content for multimedia installations whose interactivity depends solely on physical body movements?

Project Goals for Fall Quarter 2004

Establish a complete working prototype of the Luminance installation. This prototype should use all project components, including live video input and motion tracking. Simple interactivity between shadows and digital objects in Flash achieved.

To explore the nature of playfulness and what might constitute content complexity in the context of moving, responsive images.

Major Project Events – Fall Quarter 2004

10.13.04 – Achieved Success connecting EyesWeb to DV Camera as input

10.16.04 – FLOSC setup and connectivity between Flash and FLOSC established

11.01.04 – After a month of technical investigations, made decision to proceed with EyesWeb – FLOSC – Flash trio as development environment and production platform for Luminance.

11.12.04 – Attended opening of Reactive exhibit at RX Gallery including panel discussion with Camille Utterback, Brian Knep, and Scot Snibbe.

11.17.04 – Successful connectivity between all components of Luminance (camera, EyesWeb, FLOSC, Flash)

11.20.04 – Full scale setup in lab. First test participants. Observed users' delighted responses and immediate inclination to characterize the natures of ... dots.

11.23.04 – Project web site launched at <http://sasweb.csu Hayward.edu>

Complex Playful Content

Concentrating on having a functioning prototype this quarter was not just a decision to tackle “easier” mechanics rather than the trickier subject matter concerns. Because the interactive art installation arena is one where, for the most part, creating only one effect has so far been sufficient, the parameters of content development are not as well developed as in more mature media. The “language” has yet to be established.

Secondly, the effectiveness of interactive art depends mightily on users’ activities as well as the creators’ ideas. We intend to utilize user feedback to various trial content as a strong component in crafting the ultimate subject matter(s).

Therefore, as much as we looked to current examples for inspiration, the true source of evocative and lively content would be from pursuing ideas evoked by actual tests with our own prototype. By playing around with rudimentary content, we would recognize, imagine, and plan new content.

Prototype Play

Information about user interactivity was gained from just the few times our working prototype was up and running. When we had the full scale setup working in the lab, non-team users began fully characterizing simple shapes, assigning them identities and responding to the varying latency of the moving dots and circles as though the images were alive.

This was especially true when the real-time Flash objects became a tiny bit more visually complex. Once users were interacting with translucent circles of varying sizes, they “identified” them as bubbles and became more playful in their physical movements. This is encouraging for the evocative nature of abstracted images rather than literal ones as an avenue to pursue for complex subject matter. Users do (as with other art forms) bring their own experience and imagination to the art installation.

Nature of Play

Our aim is to tap into users’ natural impulses so that the participants may exercise their bodies and their imaginations. Investigations into the nature of play directly reinforce our initial intentions: to provide an opportunity for voluntary physical activity that can evoke feelings – of amusement, enjoyment, pleasure, surprise. But we’re especially alert now to the dual purpose often ascribed to some play forms, that experiencing an activity in relaxed ease can engender more complex realizations.

After observing our initial users, the role of tension is of interest: the expectation that comes from being called to a situation where alertness is needed to participate. Not knowing exactly what happens next automatically quickens users’ interest.

Emotional Design

A growing area of product design theory seems germane to our investigations. Don Norman, author of The Design of Everyday Things, has developed a way of categorizing design by how people process aspects of products. In his new book, Emotional Design, Norman describes three levels of perception in ways that parallel interactive art concerns.

Norman's levels of engagement are:

- **Visceral** : biologically set preferences, emotional, unreasoning, attracted to bright colors, warm & lit places, appealing smells
- **Behavioral** : ease of use and performance based on practicalities, how well something functions for whatever it does
- **Reflective** : messages, culture-based, analytical, relating to self and/or self image

An effective, playful activity would have visceral appeal, engage the user on a behavioral level of mental processing (be easy to operate), and should have evocative elements to be reflective. His emphasis is on moving beyond functionality to playfulness and fun, and speaks to Luminance's aims in creating engaging content.

"These levels where hope and fear, and satisfaction and anger reside. Deliver on positive expectations and people experience pleasure. Deliver something different than expected, but equally satisfying, and people have fun."

So far, aiming to create content based on manipulatable facial shapes still seems to offer the best opportunities for Luminance's first content design.

Live Research

An evening spent at a gallery panel discussion event for visiting interactive artists was quite valuable. Here were Camille Utterback, Brian Knap, and Scot Snibbe, some of the artists whose work comes nearest to what we intend. This not only gave us a chance to observe users relating to the works – and to experience them ourselves – but also provided personal discussion opportunities with the artists not usually afforded at traditional openings. The experience also highlighted the contrasts between what creators intend (what they see in their work) and what a user experiences.

Of special interest was the expressed feeling of many attendees (out of the earshot of the artists) that one of the installations was "unsatisfying". This seemed to relate to the contrast between the expectations of the users – that they would dance around and soon see their efforts appear on the screen – and the actual timing of their appearances on screen 15 or so minutes later. Immediate feedback is king.

Utterback's "Untitled Five" installation was the most complex and users spent more time playing there than with the other works. However, much of the time spent was in figuring out exactly what they were supposed to do in order to affect the art. Fortunately, the visual beauty and lively, constantly moving shapes were engaging enough to keep users curious.

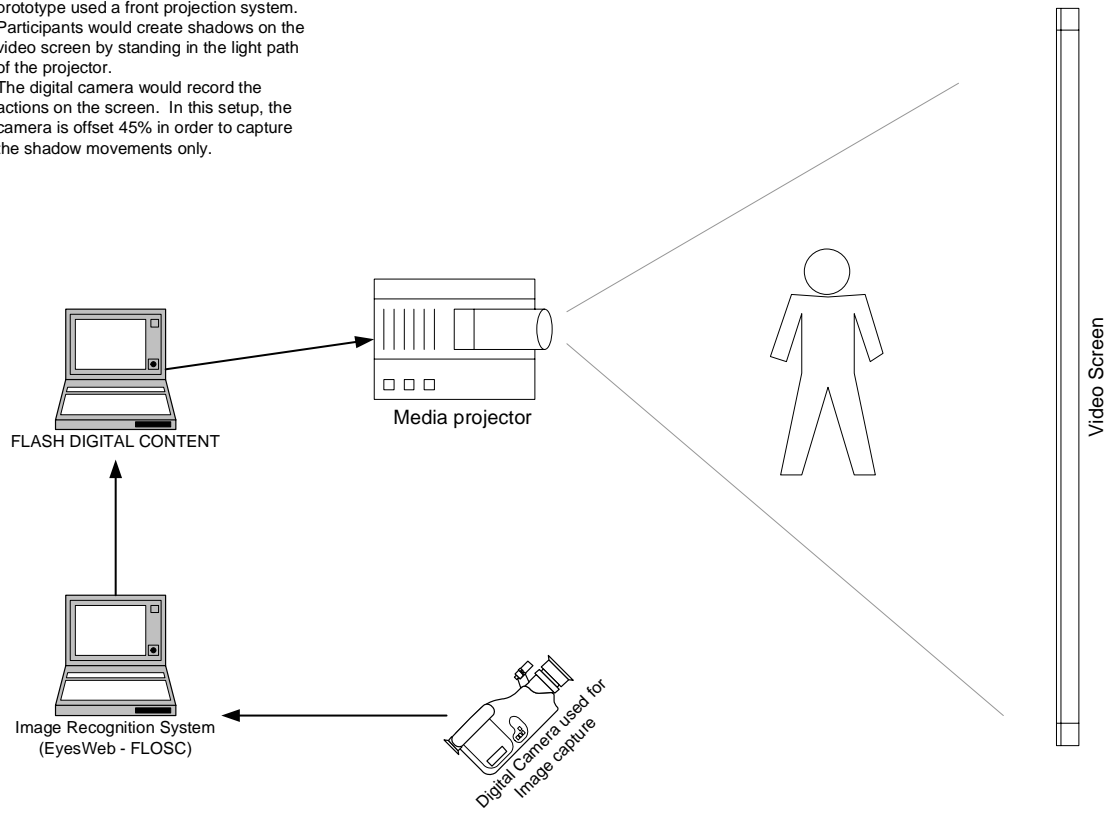
Description of Overall Technical Scheme

The Luminance installation consists of the following components:

- **Video Camera:** Used for capturing shadow movements against the video screen
- **Motion Recognition System:** Consists of a PC running Eyes Web and FLOSC. A video camera is connected to this system using Firewire. A custom patch configuration using Eyes Web has been developed by the team to capture the video input from the camera, separate the shadow movements from the rest of the image, and assign moving points along the shadow. These points are converted to X,Y coordinate positioning data formatted as Open Sound Control data and transmitted through the FLOSC gateway.
- **FLOSC (Flash Open Sound Control):** FLOSC is custom java gateway which converts Open Sound Control formatted data (OSC) into XML data which is transmitted to Flash SWF file connected to the gateway
- **Macromedia Flash:** Receives XML formatted data from FLOSC. Flash parses the XML and converts the coordinates into individual movie clips. The movie clips can interact with other movie clip objects on the Flash stage.

Luminance Prototype Setup | November 2004

The initial setup of the Luminance prototype used a front projection system. Participants would create shadows on the video screen by standing in the light path of the projector. The digital camera would record the actions on the screen. In this setup, the camera is offset 45% in order to capture the shadow movements only.



Capture Development

Choosing the Technology

We were initially turned on to the EyesWeb Open platform through one of J.J.'s contacts. As freeware, it was highly appealing to us, especially compared with similar commercial products such as Max/MSP or PureData. However, we quickly found that the EyesWeb documentation is quite sparse, meaning that we had to rely on experimentation and posts by the EyesWeb user community to puzzle out how to develop our own patches for this system.

Our first goal was finding a method of linking EyesWeb to Flash in real time. Many EyesWeb users had posted favorably of a simple freeware server application called FLOSC, written by Ben Chun. We downloaded the package and J.J. was able to initialize communication between Flash and the flosc server early in our development, meaning that the far end of the bridge was ready.

Video Tracking

We were stuck for some time on how to accurately and reliably convey shadow data captured in EyesWeb across the link. Example patches provided with EyesWeb demonstrated background subtraction from a video feed, giving us a starting point for further development.

We had assumed that we would be sending data about the entire shadow as a sort of 'mask' across to Flash. After much consideration, we decided that this approach would involve too much computational overhead on both ends of the system to be feasible, and began searching for alternatives.

Road Not Taken

One method we considered briefly was in trying to isolate uniform objects inside the shadow mask in EyesWeb, then relaying the positions of these objects across the link to Flash. This method was to approximate the shape of the shadows with accuracy depending on the granularity of these objects. Had we not discovered a different, simpler technique, we would probably still be doing development on this method, as none of us have the mathematical or programming background to create a tiling algorithm of this sort in a reasonable amount of time.

Captured!

The heart of the capture side of Luminance is the EyesWeb implementation of the Lucas-Kanade Feature Tracking Algorithm. Details on this algorithm can be found in their initial publication: [An Iterative Image Registration Technique with an Application to Stereo Vision](http://www.ri.cmu.edu/pub_files/pub3/lucas_bruce_d_1981_1/lucas_bruce_d_1981_1.pdf), Bruce D. Lucas and Takeo Kanade, the 7th International Joint Conference on Artificial Intelligence, 1981.
http://www.ri.cmu.edu/pub_files/pub3/lucas_bruce_d_1981_1/lucas_bruce_d_1981_1.pdf

Effectively, the Lucas-Kanade algorithm as applied in EyesWeb assigns a number of 'tracking points' to different distinguishing features on an image, and attempts to track their positions across multiple

frames. When tracking of a feature fails, for whatever reason, the algorithm attempts to assign that point to a new feature on that image.

In Luminance, the image fed to this algorithm is the shape of the shadow as extracted from the background on the screen, which is based on (haha) the lack of luminosity in the shadow regions. If we clip a display into the patch at this point in the data flow, we see that the source image has been processed into an extracted silhouette; white pixels compose the shadow, black pixels compose the background, and there are no features of note inside either of these regions.

This means that all of the interesting features assigned by the Lucas-Kanade algorithm are clustered along the edges of the shadow, which serves our purpose nicely.

Trial, Error and Digging

After some investigation (documentation for this implementation of the algorithm is nearly non-existent), we found that the assigned points were kept in a list of matrices in the EyesWeb. Each matrix contains the name of the point, and the X and Y coordinates of that point. After some further investigation and experimentation, we found a method to extract data about any given point from this list and pass it to the floc server.

Our first major success this quarter passing the coordinates of a single point to Flash, and having Flash draw that point on the screen. We were later able to expand the Actionscript to draw objects for multiple points simultaneously, using a method that will scale up or down to accommodate however many points are sent. Our prototype draws small circles that match the perceived locations of these points.

Next Considerations

With our prototypical front-capture setup, the camera views the screen at a moderate distance and angle, a configuration that has several limitations:

1. There is a triangular zone between the camera's lens and the screen that is effectively forbidden to the user. If the user(s) enter this zone, their images will tend to be picked up as shadow data, and wildly offset points will be assigned and drawn on the screen.
2. There is no translational component in place to correct for the viewing angle of the camera. This means that the points drawn on the screen are offset from the user's shadow by an amount that varies depending on their proximity to the screen, and a few other factors.

Our plan to use a rear-capture configuration will eliminate both of these issues. From the rear, the camera will always have an unobstructed, accurate view of the screen. In future versions, **the users will be able to stand close to the screen, and the point positions will correspond very closely with their shadows.** Allow me to repeat that again: **the users will be able to stand close to screen, and the point positions will correspond very closely with their shadows.**

Gateway

Open Sound Control

OpenSound Control ("OSC") is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology and has been used in numerous application areas, including midi, audio, and video.

OSC Features:

- Open-ended, dynamic, URL-style symbolic naming scheme
- Numeric and symbolic arguments to messages
- Pattern matching language to specify multiple targets of a single message
- High resolution time tags
- "Bundles" of messages whose effects must occur simultaneously
- Query system to dynamically find out the capabilities of an OSC server and get documentation

Sockets

The key to connecting the Eyes Web Application to Flash through FLOSC is the idea of sockets. The following general description of a Socket was found on webmonkey.com.

"Every computer on the Internet talks to other computers on the Internet using sockets. When you open a connection to another computer, you use a socket. Email servers open a socket for every user checking his or her email and every user sending an email. Web servers open (and close) a connection for every file they transmit. (Note: Some web servers are smarter than this, but handwave it for the moment — we'll come back to it.) Webcams, Napster, and ICQ all run on sockets. Every conversation consists of at least two: one on each end. If you're going to write applications which communicate over sockets, you'll have to deal with each of those ends. Even if you're only planning to deal with one side, it's still a socket."

Key Flash Classes and Methods

This section introduces some of the main Flash classes, methods, and event handler used in connecting Flash to the FLOSC gateway

XMLSocket Class

ActionScript provides a built-in XMLSocket class, which lets you open a continuous connection with a server. A socket connection lets the server publish, or *push*, information to the client as soon as that information is available. Without a continuous connection, the server must wait for an HTTP request. This open connection removes latency issues and is commonly used for real-time applications such as chats. The data is sent over the socket connection as one string and should be formatted as XML. You can use the XML class to structure the data. To create a socket connection, you must create a server-side application to wait for the socket connection request and send a response to the SWF file. This type of server-side application can be written in a programming language such as Java.

The XMLSocket class implements client sockets that let the computer running Flash Player communicate with a server computer identified by an IP address or domain name. The XMLSocket class is useful for client-server applications that require low latency, such as real-time chat systems. A traditional HTTP-based chat solution frequently polls the server and downloads new messages using an HTTP request. In contrast, an XMLSocket chat solution maintains an open connection to the server, which lets the server immediately send incoming messages without a request from the client.

To use the XMLSocket class, the server computer must run a daemon that understands the protocol used by the XMLSocket class. The protocol is described in the following list:

- XML messages are sent over a full-duplex TCP/IP stream socket connection.
- Each XML message is a complete XML document, terminated by a zero (0) byte.
- An unlimited number of XML messages can be sent and received over a single XMLSocket connection.

XMLSocket.onConnect Method

Example usage: myXMLSocket.connect(host:String, port:Number) : Boolean

Parameters

Host: String; a fully qualified DNS domain name or an IP address in the form *aaa.bbb.ccc.ddd*. You can also specify null to connect to the host server on which the SWF file resides. If the SWF file issuing this call is running in a web browser, *host* must be in the same domain as the SWF file; for details, see [Description](#).

Port: A number; the TCP port number on the host used to establish a connection. The port number must be 1024 or greater.

Establishes a connection to the specified Internet host using the specified TCP port (must be 1024 or higher), and returns true or false, depending on whether a connection is successfully established. If you don't know the port number of your Internet host computer, contact your network administrator.

XMLSocket.OnXML Event Handler

```
myXMLSocket.onXML = function(object:XML) {  
    // your statements here  
}
```

object An XML object that contains a parsed XML document received from a server.

Invoked by Flash Player when the specified XML object containing an XML document arrives over an open XMLSocket connection. An XMLSocket connection can be used to transfer an unlimited number of XML documents between the client and the server. Each document is terminated with a zero (0) byte. When Flash Player receives the zero byte, it parses all the XML received since the previous zero byte or since the connection was established if this is the first message received. Each batch of parsed XML is treated as a single XML document and passed to the onXML method.

FLOSC XML Data Type Definition (DTD)

```

<!ELEMENT OSCPACKET (MESSAGE+)>
<!ATTLIST OSCPACKET
    ADDRESS CDATA #REQUIRED
    PORT CDATA #REQUIRED
    TIME CDATA #REQUIRED
>

<!ELEMENT MESSAGE (ARRAY | ARGUMENT)* >
<!ATTLIST MESSAGE
    NAME CDATA #IMPLIED
>

<!ELEMENT ARRAY (ARRAY | ARGUMENT)*>
<!ELEMENT ARGUMENT EMPTY>
<!ATTLIST ARGUMENT
    TYPE (i|f|h|d|s|T|F|N|I) "i"
    VALUE CDATA #IMPLIED
>

```

Sample FLOSC Packet

```

<OSCPACKET ADDRESS="127.0.0.1" PORT="1080" TIME="0">
  <MESSAGE NAME="point_0">
    <ARGUMENT TYPE="f" VALUE="321.44025" />
    <ARGUMENT TYPE="f" VALUE="222.42276" />
  </MESSAGE>
</OSCPACKET>

```

Luminance ActionScript Pseudocode**Version 1**

Define an XML Socket Connection

Create Hanlder for Connecting

Create Handler for failed Connection

Create Handler for closing Connection

Create Handler for using incoming XML data through socket connection

Current Version

- XML Socket Definition
- Handler function for Connection Success, Connection Failure, Connection Closure
- If Incoming XML, then Parse XML into X,Y,label “siblings”
 - Create variables for x coordinate and y coordinate
 - Create empty movie clip on new level defined by incoming label data
 - Move movie clip to X,Y values
 - Determine random value for object radius and transparency
 - Pass radius, and transparency values to draw circle function
- Socket listener is continuously waiting for incoming XML data

Project Goals for Winter Quarter 2004/5

A working installation that can be left unattended for users to play with.

- Adapt system for usage with rear-capture configuration, acquire parts/materials for same.
- Address minor issues of system latency, accuracy. Fine-tune threshold values for use under different lighting conditions.
- Research a more elegant means of extracting point data from EyesWeb and LK tracker. Look into increasing the number of tracking points.

Expand the possibilities of visual results/movement to varying subject matter, increasing in complexity. Technically, this will involve establishing pre-existing stage objects which can interact with realtime objects manipulated by the user.

Test the prototype with users in a controlled circumstance with evaluations and video-documented responses.

Resources

Can be found on the Short Attention Span site.

<http://sasweb.csu Hayward.edu>