Interview with Mark Hellar

> Mark Hellar is a consultant on technology initiatives at a number of cultural institutions throughout the Bay Area and beyond, and the owner of Hellar Studios LLC. Before opening his own studio in 2009, Mark has worked as a systems architect at the Tides Foundation, academic technology manager at the San Francisco Art Institute, and as a digital-media specialist at the Bay Area Video Coalition (BAVC). Current projects include working on new media conservation initiatives at SFMoMA and as a technology consultant to BAVC.

June 14, 2013

Interviewers: Crystal Sanchez and James Smith

Please describe your experience.

I'm an independent consultant. I've been working with SFMoMA[1] since 2009 on the conservation and care of digital artworks, with a focus on software-based art.

In 2008, the Museum acquired two works that had been commissioned in 2000 for an online exhibition called *e.space*[2]. The curator of media arts, Rudolph Frieling, decided he wanted to acquire them for the collection. One was a multimedia/website work called *Predictive Engineering II* by Julia Scher[3]. The other was *Agent Ruby*[4] by Lynn Hershman Leeson, which was a Java program that presented an artificial-intelligent website based on a character Tilda Swinton played in Lynn Hershman's 2002 movie *Technolust*. You can see that piece at AgentRuby.net; it's been running since 2001. Those were the first two pieces I worked on.

In addition, I'm been working with SFMoMA to launch its digital repository. We're looking at Archivematica, which is based on the Open Archival Information System (OAIS)[5], an ISO standard.

For seven years I worked in IT as a UNIX engineer and programmer in a corporate environment, so I had a technical background. Then in 2003-04, the city of San Francisco had these grants to retrain the workforce in new media; I got a scholarship from the Bay Area Video Coalition[6] and took 300 course hours of video and new media training. It was a lot of post-production stuff—things like Final Cut Pro, video engineering, Avid, and more esoteric software like Max/MSP and Jitter[7], which is digital signal processing software for audio and video.

So I learned all this multimedia software and after that I shifted careers. I went to the San Francisco Art Institute[8] as the academic technology manager, where I ran the digital media studio and all academic technology facilities. I was there for four years. At that time, I was still employing my technology background from my corporate IT years, but I was also integrating that with my new digital media skills in this fine arts academic environment.

At the time they had a lot of international artists-in-residence come, some of whom were working at the intersection of art and technology. I remember one time, we set up this wireless sensor with a bunch of custom wireless electronics, sensors and a microcontroller, so when someone walked across the main courtyard, it would play this big audio loop out of the bell tower on the campus and you could hear it throughout all of North Beach. That was one example. It was very non-standard and very interesting. In corporate technology, there are a lot of standards, but with fine arts requirements for technology, you have to re-invent and come up with new and interesting systems.

After the San Francisco Art Institute, I went to the Bay Area Video Coalition (BAVC) as a visual media specialist. The city had commissioned a 10 gigabit fiber connection to the Internet, and the Director of BAVC at the time wanted to know what he could do with that connection that would be innovative. So we were trying to think of projects that would allow us to really innovate with this lovely infrastructure. BAVC also has a preservation department where they digitize all sorts of old formats—1" open-reel, U-matic, Betacam, laser disc. They've been doing this for years, and had been working with SFMoMA on and off. I think they did the post-production on one of the *Cremaster* series by Matthew Barney also they digitized Bill Viola's work for an exhibition in 2001-02.

I was at BAVC around 2008-09, when Rudolph Frieling decided to acquire the software-based works that I mentioned earlier. Frieling and Jill Sterrett, the director of conservation and collections, came to me and said, "You guys have really helped us in our conservation efforts with video, but now we're collecting software. Can you help with that?" The executive director of BAVC at the time said, "Talk to Mark; he's our digital media specialist." So I had a long talk with Jill, and that's how the relationship with SFMoMA began.

How do your approaches to video-based work and software-based work differ?

When started working with SFMoMA, they were still receiving tapes, which were very standardized—it was DigiBeta at the time. That would be digitized—we're talking about standard definition works—to 10-bit uncompressed video in a QuickTime wrapper. So it was all very spelled out.

Ruby and *Predictive Engineering II* were both software works, but otherwise they were very different. These works consist of a number of components; for example, *Ruby* had a Java program that was a natural language interpreter, which communicated via a Web server to a Flash multimedia interface. You would enter user input and it would be would be analyzed by a Java program, then it would scan a database that Lynn Hershman's programmers created to return an artificially intelligent response. It was a very exotic set of components. Julia Scher's piece was also a network of components; it was a little less complex than *Agent Ruby*, but it was still about 11 HTML pages, each containing a Macromedia Flash object. Each Flash object contained hundreds of animation layers—images and sounds, then ActionScript code to make them interactive.

So we have these two software-based works. They were written in different languages and have different components. We realized that the technical complexity and variety of components and behaviors in these kinds of works created the need for new forms of documentation. We came up with a form of documentation called the "technical narrative," which is now a standardized record for documentation of any digital artwork that SFMoMA acquires. It consists of four parts:

1. A very high-level, functional summary description of how the work operates as a whole. This part of the narrative provides a platform-neutral description of the work. What does it do? How does it operate?
2. A modular examination of components, what they do, and their related functions. This section looks at each component individually, such as the Java natural language processor in *Ruby* or the Flash files in *Predictive Engineering II*. It defines what its function is. It also gives a high-level examination of how all these components work together as a complete system.
3. A detailed description of the artwork as it exists upon acquisition. This gets into specifics about the hardware, software, operating system, environment, languages, code, versions, etc. We examine the components and their specific technologies to get an understanding of how they serve the operational requirements of the work.
4. An analysis of current technology and an evaluation of its longevity, so we can think about obsolescence. The last part considers the long-term stability of the piece with possible strategies for preserving the work over the long term. For example, both *Ruby* and *Predictive Engineering* have Flash components, and we know that Flash is going away. It doesn't play on Apple devices, and there's no longer support in Android. So we looked for alternatives (like HTML 5, for example) and a strategy to migrate those Flash components to them.

Is that kind of migration—say from Flash to HTML 5—something you would do independently of exhibition? Is this the kind of issue that is revisited regularly, or is it exhibition-driven?

The tricky part with these two works is that they are *always* on exhibition; they have both been running from 2001 until today. You know if they have become obsolescent, because they are always on.

One thing about *Agent Ruby* that we did not know when we acquired it is that it had been recording all of the conversations it had with participants, starting in 2001. We had that in the technical narrative, even though it was not originally considered a critical component of the work, and Rudolph Frieling got very interested in it. There was an 80-gig log file of these conversations, and we wrote Python scripts to mine it and pull out certain topics defined by the curator. These conversations were displayed along with the work as "The Agent Ruby Files."[9] They are in binders next to the kiosk in the Museum, and we also have a huge vinyl printout of excerpts along the walls. In that case, it was interesting because the technical narrative we wrote created a re-interpretation of how the work was exhibited.

Originally, *Ruby* was on this old server, and we found some other things on it that were unexpected and considered non-critical: a different Flash interface that had never been used, a 3D model, and some code that was a kind of text-to-speech program. It looked like they had been trying to create a 3D model that would convert text from the artificial intelligence program into speech, but that never happened. So we found efforts that were abandoned, and we documented them in the technical narrative. But at the time we wrote it, they were not critical components, and we noted that.

Could this technical narrative you have developed be used by other organizations?

I think so, I just presented on this at the American Institute of Conservation (AIC), and we have this *Matters in Media Art*[10] partnership between MoMA, SF MoMA, and the Tate, we have talked about it there. The Tate is interested in the technical narrative, and they are employing it on their software-based works. So it's not an ISO standard or anything like that, but I think there is interest in having some best practices about how to approach these software-based works.

A topic that comes up in the OAIS model is this concept of representation information[11]. The general idea is that you have this digital object—the bits—and you need to represent it in the way it was originally intended. So what documentation do you need? In these cases we are discussing, how do you maintain the artistic integrity or the intent of the artist? When I think of the technical narrative and then read the OAIS model, I can see a relationship between what we have done and its concept of representation information.

Did you use the OAIS standard to guide you in designing the technical narrative?

No. We came up with the technical narrative in 2009, and we only started looking at OAIS in more depth around the beginning of 2012 or the end of 2011. But we do find parallels, at least in that area of representation information, which justified to us that there is a need for this type of documentation.

Are there other standards out there that you look at to guide you in this work?

On one project, we had a software developer from Adobe, and she shared their software template documentation. So we're looking at things from the field of software development to guide us as well, especially with the idea of managing change. If you put a videotape in a physical archive, you don't want it to change. But it's in the nature of these software-based artworks that they require constant change, because that's just how the software industry works. It's a landscape of obsolescence that is always changing. There's new hardware, new devices, new features; companies come and go. So with the software-based works, you have to manage that change. We're looking at internal systems in the software industry for managing change.

Could you give an example of how you manage change?

In the case of *Ruby*, it was running on Java 1.4, and that was reaching the end of its life several years ago. So we went to Sun Microsystems' website, looked at the latest version, and were able to recompile the code in Java 1.6, which was the current version at that time. We also looked at things like the operating system, which was Red Hat Linux 7, I believe—a very old version. So we upgraded that to Ubuntu Linux 10.4 long-term support; Ubuntu Linux is very good, because they commit to supporting a product through an end date, which is usually five years out. *Ruby* was running on an actual physical machine—one of those old beige PCs, circa 2002, with a parallel port and a floppy disk. So we decided we should move it onto a virtual server, so we no longer have to worry about the hard drive crashing on a 10-year-old machine.

To summarize, we looked at every specific component in the technical narrative, then asked, "Where is this at in its life cycle? And what options do we have to upgrade it?" We came up with a plan, then went to Lynn Hershman Leeson and reviewed it with her. We said, "These are the upgrades we need to make sure she will continue to run and for security, so we can at least guarantee the next five or ten years." We reviewed the plan to make sure it stayed true to her vision of the work. That's our approach to anti-obsolescence.

There are some file format databases out there such as FITS[12] and PRONOM[13]. Those are very good for things like PDFs or QuickTime wrappers or things like that; but at the time we were working on Ruby we couldn't find more exotic components like Macromedia Flash or MAX/MSP source code there. That's not to say those databases can't serve that purpose. In fact, we should be submitting entries as we do this work.

That touches on the question of where more research is needed in the field—maybe one idea is more categorization of some of these more complex file formats and systems and adding those to databases that are already out there and being used for things like wrappers. Do you think that would be an appropriate space—someplace like PRONOM or the Universal Formats Registry? Or are those components just too different, and you have to deal with them individually?

I don't think it's a question of either/or. Look at Flash. You can talk about the longevity or obsolescence of it as a whole. But then when you look at individual Flash projects and how they are individually designed and have ActionScript code or things like that, a registry is not going to help you port the code. It might say something like, "For migrating Flash, you might want to consider HTML 5." But it's up to the conservator of a particular work to look at the source code, understand it and document how it works, so that is can be migrated to the format recommended by these registries. I know Glenn Wharton and Deena Engel at NYU just presented a paper on documentation of source code as a conservation strategy for software-based art[14].

How do you work with art conservators to make these decisions? It seems like a very complex strategy to port the work and recompile it to make it work on new platforms.

SFMoMA has a group called Team Media that meets once a month. It includes conservation, curatorial, exhibition installation, and registration staff. We talk about these works and put them on the table. I come from a very technical place, and I'll present that perspective. But then everyone else presents his or her ideas, perspectives, and concerns. So we have this collaborative system that allows us to reach consensus through an open dialogue about how to move forward.

Do you always talk in terms of specific works, or do you ever discuss classes of similar works that can be treated in similar ways?

I could talk about video as a class. The museum is not getting DigiBeta tapes so much anymore; they're getting hard drives. And they're not getting just works in standard definition; they're also getting high definition. With the tapes it was DigiBeta, 10-bit

uncompressed, with a QuickTime wrapper. But now, when we get a digital file, it could be a DV file for standard definition, which is compressed. It could be h.264 with varying bit rates. It could be ProRes—we get a lot of ProRes. So we came to the decision that we need to have a conversation with the artist during the acquisition to advocate for the least-compressed format possible. When you ask about classes of works, that's what comes to mind.

When we talk about web-based works, there is a lot of variability, but we do have some points of agreement, like that they are always on. We're working on another piece now, *Learning to Love You More*[15] by Miranda July and Harrell Fletcher, and we're coming up with standards about the domain name. The registrar is going to handle it, so we have a standard protocol for that. It doesn't sound like a big deal, but the domain name for these web works is how they are viewed and accessed.

For software-based works, there are a lot of commonalities, but there is a lot of variability too. At the Smithsonian workshop, I was asked, "What will you do when you have to deal with 1,000 works?" My answer was, "Well, we have about eight right now; so I don't think we're going to have to deal with 1,000 works any time soon." We're giving individual attention to all of these works because things are just emerging. In a way we're lucky because the collection is quite small and we can pay a lot of attention to each work and define standards where they feel appropriate. So hopefully when 1,000 works in a collection is the norm, we will have some kind of structure. We're exploring, discovering and defining that now.

Can you see any commonalities among these works yet that may be amenable to guidelines that apply across works?

There are some things like that. For example, we want the source code at acquisition; that's something like a standard.

But we recently met with the architecture/design department because they were interested in what we were doing in the media department and how it might help them to preserve their works. But the works they brought to the table were very different; they were things like Google Glass, the Pebble watch (an e-watch that was a big Kickstarter success), a fitness watch with sensors, and things like that. I realized they would have to approach these things in a totally different way, because we're never going to get the hardware schematics and source code for Google Glass—it's a trade secret. Things like that in the architecture and design collection are also software-based works, but because of their nature, we can't treat them the same way as we treat the media artworks, where we can get the source code and have access to the artists and their engineers.

I heard you mention Max MSP, which I'm familiar with, so I'd like to target a question there. If you have a Max work and you have acquired the Max patch file and third-party library objects, do you think you could take that knowledge and apply it to another work built in Max? At what point do you have to focus on the specifics of each work?

We do have a Max work: *Sonic Shadows* by Bill Fontana[16]. It's an installation that pipes data from accelerometers in the steam room to ultrasonic speakers on pan-tilt heads. The Max patch processes the audio and controls the pan-tilt heads via the DMX protocol. Miller Puckette originally created Max; it went through multiple companies and it is now owned by Cycling '74 in San Francisco; it is proprietary software, although it's very well documented.

Miller Puckette later went to UCLA and re-wrote it as open source—but not completely—in a language called Pure Data. So we talked to the engineers who built the Fontana work and asked if we could recreate it in Pure Data. I still think that might be something to look at, because Pure Data is probably one of the closest open-source analogs to Max. If I was putting Max in the file format obsolescence database, I would say, "Consider migrating to Pure Data." If I had a bunch of Max objects and Cycling '74 went under, I would initially look at that as a migration strategy.

But there are other things to consider there. One of the DMX control components in the Fontana piece was a custom object that was built by some programmer with a private license; does Pure Data[17] have this third-party component that handles the DMX protocol? Would I have to look at every object in Max MSP and how they work together as described in the technical narrative to see if I could create an analog in a different language such as Java?

It sounds like you have standardized the documentation of the risks and the components, but it's hard to standard accession formats or preservation plans for classes of works.

I have worked with Lynn Hershman Leeson recently on developing some new works. We had this piece two months ago that would pull images using the Flickr Application Programming Interface (API), and I asked, "Lynn, could you collect this?" Because it's calling on a third-party API that's not even hosted locally— it's out in the cloud; she's into these network-based software works. I think I'm seeing more of that.

There's a piece at MoMA called *I Want You To Want Me*[18] by Jonathan Harris and Sep Kamvar, which works by pulling from a bunch of data sites for their RSS feeds and things like that, then presenting them on a graphical display with a touchscreen. I don't think this

was driven by conservation, but it is now displayed with a subset of data on a SQL database that was gathered over a period of years.

Third-party APIs are a challenge. A lot of the software components in your works use third-party APIs that may be patented. And now you are getting all these cloud APIs—Flickr, Google Maps, or whatnot. I think as we see more network-based software works- that will become more of a challenge and a question.

If you were to train someone to do your job—working in an art museum on the technical side of software-based works—what would that training look like?

No one can know everything. I would say the most important thing is to be a technology generalist who can look at a work and understand the technology that's built in—be it Python or Max or Flash or whatever—and who can do the research to identify the current standard technology and identify outside experts who can help with migration or emulation.

I have a list here that comes from a document we did for the Matters in Media Art project calling out the skills needed for the modern conservator of time-based media art. It lists computer science, computer engineering, programming, code documentation, code recompiling and porting, software emulation, digital video technologies, and digital repository management. There's this huge chart, and I was thinking, "Well, that's like the Über-Expert." That's why I go back to saying that the most important is to be a generalist who understands the landscape of technology and to know who to reach out to and define what you need.

1.See: http://www.sfmoma.org/

2.See: http://www.sfmoma.org/exhib_events/exhibitions/espace

3.See: http://www.sfmoma.org/media/exhibitions/espace/scher/jsindex.html

4.See: http://agentruby.sfmoma.org/

5.See: http://en.wikipedia.org/wiki/Open_Archival_Information_System

6.See: http://bavc.org

7.See: http://cycling74.com/products/max/

8.See: http://www.sfai.edu/

9.See: http://www.sfmoma.org/exhib_events/exhibitions/512

10.See http://www.tate.org.uk/about/projects/matters-media-art

11.See: http://www.dcc.ac.uk/node/9558

12.See: https://code.google.com/p/fits/

13.See: http://www.nationalarchives.gov.uk/PRONOM/Default.aspx

14.See: http://www.conservators-converse.org/2013/06/41st-annual-meeting-electronic-media-session-may-31-technical-documentation-of-source-code-at-the-museum-of-modern-art-by-deena-engel-and-glenn-wharton/

15. See: http://www.learningtoloveyoumore.com/

16. See: http://www.sfmoma.org/exhib_events/exhibitions/416

17: See: http://puredata.info/

18: See: http://iwantyoutowantme.org/