MEEMOO: HACKABLE CREATIVE WEB APPS

FORREST OLIPHANT

Meemoo is a framework for creating hackable web apps. An app is a graph of modules and the wires that connect them. A module is a web page that can live anywhere online, and use any web technology. This web page includes JavaScript that defines the module's inputs and outputs: what data is accepted and what kind of data will be sent. The wires define where each module sends data. The graph that defines an app's layout, routing, and state can be saved and shared with a small amount of text.

I have focused module development on realtime animation tools, as this makes it simple to explain and engage creatively with the concept. It is not limited to animation though; any app or system that can be described by a dataflow graph can be made into a Meemoo app.

It has been designed with a few groups of people in mind: creators, hackers, and modders. Creators will use Meemoo apps to make audio-visual media and share them online. Hackers will explore how the apps work, and rewire them to work differently. Modders will use web technologies to create new modules which will allow different kinds of apps. It is designed in a way that each of these levels leads to the next, encouraging people "down the rabbit hole" towards learning coding.

In this thesis I describe...

CONTENTS

1 INTRO

Learning by example. How I learned HTML. Social constructionism?

1.1 *Hackability*

Designing for hackability implies respect. The designer acknowledges that they can’t imagine every use of their thing, so they enable people to hack, bend, mod, or fork it to their will.

Meemoo is designed for hackability on all levels. On the highest level, people can add and remove modules and wires without coding knowledge. On the lowest level, the entire project is Free software. https://github.com/meemoo

1.2 *Metamedia*

Kay: computer can be all other media, and is also active. Internet makes all media available everywhere. Web 2.0 made it participatory for publishing text, photos, videos. Twitter, Flickr, Youtube: one solution per media, no coding needed.

Web 2.0 makes media distribution easier by abstracting away FTP, HTML, etc.

Maybe Web 2.5 is creative apps online, like Picnik photo editor and TinkerCAD 3D design software. Media creation without desktop software.

Meemoo makes the modes of production also participatory. This makes it possible for people to invent and share new media. Web 3.0: web software creation without desktop software (or code)?

If, as McLuhan proposed, “the medium is the message,” then what kinds of messages are implied by the metamedium?

D. Rushkoff argues (Program or Be Programmed) that learning to code opens people’s eyes to the design of all systems. They see that bad design can be fixed.

1.3 *Tools*

From making tools to making toolmakers. Tool designer thinks: “How will people use this tool?” Toolmaker designer thinks: how will tool designers use this tool (to think about how people will use their tool)? Thinking about thinking about thinking (about thinking?). Metaconstructionism?

Meemoo will enable people to become tool designers before learning coding skills.

Programming skills distinct from coding skills. Visual programming makes the dataflow logic visual. With text-based coding you need to keep track of these relationships in another way.

2 CONTEXT

2.1 *Computers as abstraction*

2.1.1 *Mainframe to PC: Dynabook*

Kay: Promethian effort to bring computation from "priesthood" of mainframe admins to all people, especially children.

2.1.2 *Programming for kids: Smalltalk, Logo, Scratch, Alice*

Constructionism: getting kids programming gets them to think about thinking. The metamedium gives them control.

2.2 *Visual programming languages*

Meemoo is a kind of dataflow visual programming environment. This means that on a programming

2.3 *Free Software movement*

Can't own ideas. Why do people give away their work?

2.4 *Open hardware and maker movement*

Arduino modules: Gameduino, heart rate sensor, Lilypad... abstracting some of the electronics intricacies into modular components.

"If you can't open it you don't own it."

2.5 *JQuery Plugins*

JQuery : plugin :: Meemoo : module

Large community sharing plugins. I made this thing and it is useful to me.

3 DEVELOPMENT

Development on Meemoo's ancestor project began in January 2011. In October 2011 Meemoo became a Mozilla WebFWD fellow project.

3.1 *Previous work*

Meemoo is an abstraction of all of my earlier creative web app experiments, and I plan on rebuilding them in Meemoo to make it easier for me (and others) to modify how they work. I call it my *opus sink*.

3.1.1 *Media Bitch (2002), Flash*

http://forresto.com/oldsite/interactive/mbx/mediabitch.html

3.1.2 *Kaleidocam (2007), Quartz Composer*

https://vimeo.com/387429
Learning QC and dataflow programming.

3.1.3 *Megacam (2010), Flash*

http://sembiki.com/megacam
Webcam apps inspired in part by Lomo cameras. I chose presets for each toy to make it simpler, but that also removed the possibility to experiment with the variables.

3.1.4 *Looplab (2010), Pure Data*

https://vimeo.com/16956269 http://www.flickr.com/photos/forresto/5125930908/
Learning Pure Data. Network communcation of identical apps, each passing data to the next.

3.1.5 *Opera stage projection mapping (2011), Quartz Composer*

Last year I was working on a multi-screen video projection system for the set design of an Opera. I found Quartz Composer modules for midi control, video playback, and projection mapping. I patched them together to create a system that controlled video on four projection-mapped screens from one projector. These modules were all shared online by their authors in the open-source spirit. I needed to add a feature to one of them, and was able to do so in XCode.

Meemoo will make it possible for people to not only share such modules online, but also wire them together, experiement, and save output instantly online. This will lower the barrier to entry and increase collaboration potential.

3.1.6 *Web Video Remixer (2011), HTML*

This is the direct parent project of Meemoo, where I figured out how to communicate between web pages in iframes.

3.2 *Software design for hackability*

One of the goals for the project is that it is hackable on all levels. On the lowest level, this means that the code is open source.

3.2.1 *Common communication library for modules*

Each Meemoo module needs to include meemoo.js, which handles message routing. The inputs and outputs are then specified as in Algorithm 1 on the following page.

3.2.2 *Sharable app "source code"*

The graph format is JSON, like graph={nodes:[],edges:[]}. This "text blob" stores the position, connections, and state of all of the modules in the graph, which can be considered the source code of the app. Because it is a small amount of text, it is easy to share the app source code in forums, image descriptions, comments...

3.3 *User experience design for hackability*

3.3.1 *Direct manipulation*

Ben Shneiderman

Algorithm 1

```
Meemoo
  .setInfo({
    title: "example",
    author: "forresto",
    description: "this script defines a Meemoo module"
  })
  .addInputs({
    square: {
      action: function (n) {
        Meemoo.send("squared", n*n);
      },
      type: "number"
    },
    reverse: {
      action: function (s) {
        var reversed = s.split("").reverse().join("");
        Meemoo.send("reversed", reversed);
      },
      type: "string"
    }
  })
  .addOutputs({
    squared: {
      type: "number"
    },
    reversed: {
      type: "string"
    }
  });
```

Visual indication of what is happening in each module. (Like TouchDesigner). Dragging to change variables.

3.3.2 *Visual programming "patching" metaphor*

"The use of flexible cords with plugs at their ends and sockets (jacks) to make temporary connections dates back to cord-type manually operated telephone switchboards (if not even earlier, possibly for telegraph circuits). Cords with plugs at both ends had been used for many decades before the advent of Dr. Moog's synthesizers to make temporary connections ("patches") in such places as radio and recording studios. These came to be known as "patch cords", and that term was also used for Moog modular systems. As familiarity developed, a given setup of the synthesizer (both cord connections and knob settings) came to be referred to as a "patch", and the term has persisted, applying to systems that do not use patch cords." - Wikipedia on Moog

3.4 *What is abstracted*

As a programmer, working in Puredata and Quartz Composer can be frustrating. Certain logical constructions that would be easy to describe in code become a jumbled mess of boxes and wires.

4 TESTS/RESULTS

4.1 *User testing and feedback*

Aino - Camdoodle
Ginger - "You should add an onionskin"
Teemu - Metronome animation
Facebook Beta group

4.2 *Economic model illustrated with Meemoo*

http://meemoo.org/blog/2012-01-24-friction-free-post-scarcity-creative-economies/

4.3 *Live animation visuals for dance party*

http://www.youtube.com/watch?v=x1L-2jRVyhk

5 FUTURE DEVELOPMENT

This idea is bigger than one developer and one master's thesis. I plan on finding resources to continue work, and to bring more people with varied talents into the project.

5.1 *Community for sharing apps*

Meemoo was designed for sharing.

5.2 *Socket communication*

UX and server for sending arbitrary data from Meemoo on my smartphone to my laptop to your tablet (and back).

5.3 *Meemoo hardware*

Cheap computers (Raspberry Pi) + knobs + sliders + physical patch cables for performative interaction.

5.4 *Twenty Apps to Build With Meemoo*

In the spirit of Seymour Papert and Cynthia Solomon's 1971 memo, Twenty Things to Do With a Computer, I present this list of potential Meemoo apps:

1. Kaleidoscope with reconfigurable mirrors
2. Experiment with video feedback with webcams pointed at screens
3. Text-to-song generator with computer generated voices singing in harmony
4. Artistic visualization of data from bio-sensors
5. Beatbox control of video mashup (sCrAmBlEd?HaCkZ!)
6. ...

6 CONCLUSIONS

I contacted Ze Frank to ask if he would be a project advisor. He gave me some good things to think about:

"Creating 'possibility spaces' can be exciting for a number of reasons... but also can be a false God. It can be an excuse to never to actually grapple with whether there is value in the output itself, whether beauty is enough, whether people actually want what you are making, etc..."

Making a creative tool maker is pointless if, in the end, nothing creative is made. My dream is that somebody will make something beautiful with it. Shouldn't that somebody be me? If I don't do it, why would anybody else?

REFERENCES