

# Programming Tools for Interaction Design Soft-sketching

Alexis Morin

Umeå University

Umeå Municipality

Sweden SE-901 87

almo0021@student.umu.se

## ABSTRACT

Interaction designers who are not software engineers sometimes feel a need for software and hardware sketches in order to bring their ideas and designs to life. A number of tools open doors to these designers to create “quick-and-dirty” hardware and software sketches. Even with increasingly fast processors, many of these sketches are built in a way that wastes hardware resources. The number of abstracted, simplified and slow programming languages has created an ecosystem of sluggish runtimes with heavy hardware restrictions. In this paper, in this paper, I propose a return to production programming environments in order to allow hardware and software sketches to fulfill their original goal: deliver a partial look and feel of the idea being designed.

## Categories and Subject Descriptors

D.3.2 Design Languages

## General Terms

Languages

## Keywords

Software, sketching, programming, languages, performance, hardware, processing

## 1. INTRODUCTION

Whilst redesigning the graphic user interface (GUI) for the FixturLaser XA Pro laser industrial shaft alignment device<sup>1</sup>, it came to be relevant that a sketch be created in order to better illustrate some decisions that were made during the design process. For multiple reasons which are outside the scope of this paper, the proposed hardware that was to support the design proposal is a computer tablet, regardless of its operating system. The hardware on which the working version of the software sketch was executed on is an iPad[2] for the reason that it was the only tablet hardware available for use at the time.

The process for getting any interactive work natively onto an iPad is somewhat convoluted regardless of the technologies used. The options are somewhat limited and always include purchasing development licenses from the Apple developer program[3] and going through a costly and complex process of setting up developer certificates on the device: far from being the best choice for anyone.

On the contrary though, the process one has to go through in

<sup>1</sup> The GUI redesign of this device was the main component of a graphic design project at the Umeå Institute of Design of the Masters on Interaction Design program. The XA Pro is a touchscreen wireless unit used to assist technicians in the task of industrial shaft alignment. The XA Pro uses a design developed by Semcon AB, based in Gothenburg, Sweden.

order to obtain a basic grasp of the programming principles of Objective-C[4] on iOS-based[5] platforms is rather straightforward. This paper is about the advantages of going through that process and of using the appropriate programming paradigm for the appropriate task rather than banking on in-between prototyping and sketching languages.

## 2. Native code in action

The graphic user interface design proposal for the redesign of the FixturLaser XA Pro industrial shaft alignment device was completed along the assumptions that it would be presented on an iPad but designed for implementation on any tablet based operating system. It is in this case considered to be relevant to have an as-close-to-final software sketch running on an iPad device[7]. The programming language used to complete this sketch was Objective-C, which happens to be the native language for use on Apple iOS-based devices. The main advantage of working this way lies in the better response time with the device and being able to use the SDK in order to produce a native GUI. Using any other programming language or environment would require the designer to recreate most of the widgets being used from scratch. Using the native SDK therefore translates directly to improved touch interaction with native speeds and framerates. Such performance cannot be achieved by using Adobe Flash[1], JavaScript[28] or any other abstracted programming methods. Producing the content in HTML for tablet browsers also does not bring the native feel that users know from various tablet environments.

```
- (void)setDetailItem:(id)newDetailItem {
    if (detailItem != newDetailItem) {
        [detailItem release];
        detailItem = [newDetailItem retain];
    }
    // Update the view.
    [self configureView];
}

if (self.popoverController != nil) {
    [self.popoverController dismissPopoverAnimated:YES];
}

- (void)configureView {
    // Update the user interface for the detail item.
    detailDescriptionLabel.text = [detailItem description];
}

#pragma mark -
#pragma mark Split view support

- (void)splitViewController:(UISplitViewController*)svc willHideViewController:(UIViewController *)aView
    barButtonItem.title = @"Root List";
    NSMutableArray *items = [[toolbar items] mutableCopy];
    [items insertObject:barButtonItem atIndex:0];
    [toolbar setItems:items animated:YES];
    [items release];
    self.popoverController = pc;
}
```

Figure 1: A code snippet from the XCode programming environment.

It took 4 days of learning Objective-C<sup>2</sup> to acquire knowledge sufficient to program the sketch to the extent desired. It took another 2 days of programming work to achieve a worthwhile result. The architecture of the application was also far from ideal, but this being a UI redesign proposal, my software sketch did exactly what it needed to do. I was able to showcase a sample of

<sup>2</sup> The learning was completed through the Apple Computer sanctioned Stanford University Computer Sciences 193-P course.

the visuals running on the hardware that was chosen for the design.

The hardware and software combination used by FixturLaser in their current XA Pro device has remained virtually unchanged since 2005. The hardware runs an embedded Windows CE[18] on a 600MHz processor with a Flash interface and a Java backend, which is required to interface with the hardware. The overall experience is inconsistent and sluggish at best. Considering what other technologies have evolved since then, the mix of technologies used in the XA Pro would be considered obsolete by current day standards. The software sketch built on iPad in such a short amount of time also goes to show that the cost of changing the software or changing the hardware-software combination would be lesser than the manufacturer originally estimated, this assuming that most of the design specification is already established.

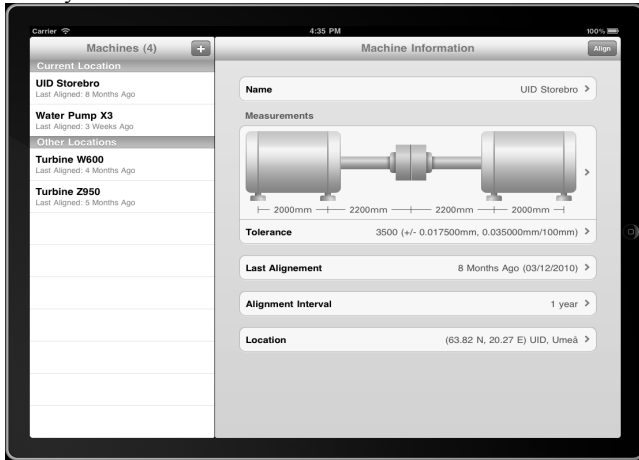


Figure 2 XAOne design proposal sketch on an iPad simulator

### 3. Programmers at work

#### 3.1 Computer engineers

People who program seek safety. Let me explain: they very seldom explore programming languages outside of what they learned in school and this to their detriment. The end result of such programming will be functional, if only in a “good enough” manner. For example, Java[23] is a language that students will learn heavily when undertaking a degree in Computer Engineering. It includes several desirable features for people who are learning how to program. Several computer engineers will go through an entire professional career and write anything that is asked of them in Java. If a required software library is inexistent, they will write it in Java, which always does work but is seldom the ideal solution.

There exists multiple programming languages because there exists different problems that one can solve in various ways. Erlang[9], developed by Ericsson, is made for solving large concurrency<sup>3</sup> problems as well as running without ever stopping. It is even possible to update a module in the code while it is running. That is why it runs telephone lines. Lisp[17] and its dialects or derivatives like SmallTalk[14] and Scheme[26] are performant when dealing with long lists of data. R[25] is widely used in the field of statistics applications. MATLAB[16] is used

<sup>3</sup> The simultaneous occurrence of events or circumstances [http://www.merriam-webster.com/dictionary/concurrence]

for advanced mathematical visualization as well as physics simulation. OpenCL[12] allows for the execution of kernel level instructions on GPUs (Graphical Processing Unit) and as such can only used in this manner. These are only some examples of specialized programming languages. By doing a minimum of research before committing to creating a software sketch, it is possible to examine the possibilities and make a decision as to which tool is best for the job.

#### 3.2 Designers

In the field of Interaction Design, designers have been blessed with a few platforms in recent years that attempt to make programming somewhat of an easier pill to swallow. Because sketching in software or hardware is often but a portion of the design work, tools and programming environments aimed at designers make away with the better part of the complex programming metaphors that designers seldom take care in learning. Most of these abstracted languages and environments are used to build sketches of applications. A software sketch usually has a limited set of features and serves the purpose of demonstrating only key features in a design.

One of these languages is called Processing. It is often used as an introduction to programming for many designers and artists. It is extensible using Java-based libraries but quickly drops in performance when creating graphic-heavy applications. LUA[15] is built on C and focuses on rapid construction of applications and a tight core. Max/MSP[8] is a node-based programming language often used by artists or designers in order to process audio and video signals for installations. When a project gets too complex, projects usually get shifted into production programming languages with real UI libraries and efficient backends.

#### 3.3 Artists

Many contemporary artists use the digital world as a canvas. The artists working in the field of Computatin Arts and Cyber Arts, for example, often create pieces that compel the viewer through the use of digital media, projections, ambient sound, and installations. Here is the story of NextText, which is a good example of the argument I support in this paper.

NextText is a software library for the manipulation of typographic symbols written for Processing by Elie Zananiri in 2007. It was featured in some typography-centered pieces by Jason Lewis, such as “What They Speak When They Speak to Me”. [13] Lewis of OBX Labs[21] quickly realized that the performance of NextText was lacking. Indeed, once 50 typographical shapes or more were displayed on screen, the application would suffer from a noticeable decrease in framerate. To remedy the situation, Lewis had Mr. Softie completely written by Bruno Nadeau. Mr. Softie is a standalone application that combines most if not all of what NextText can do, but this time in a suitable programming language. This time however it was made with a C++[27] core and QT[20] user interface. The rewrite of NextText allowed for improved framerates and the capability of an increasing number of simultaneous actions to a level that it would never have been able to achieve previously. The use of a lower-level language improved performance in the manipulation of typographic characters. It has permitted the creation of typographically complex art pieces such as “Muds” [11]. This is a perfect example of a design-related project that was initially built using the wrong tools for the job; its path was

then corrected due to necessity and it is as such a better-adapted tool today.



Figure 3: A piece titled “Feel” from David Jhave Johnston's “Muds”

## 4. Using the right tools

The aim of this paper is not to affirm that all tools made available to designers are irrelevant. It rather wishes to sensitize designers about the use of appropriate tools for the best practice of Interaction Design. [7] If there is anything I learned from programming it is that there are oftentimes more solutions that one would like. The skill of a good programmer lies in choosing one of these solutions for justified reasons, sticking to the design specification and implementing the solution.

### 4.1 Tools for sketching in hardware

This paper is focused around using systems to perform computational tasks in a design context and doesn't only focus on the use of software within a desktop operating system. The choice of tools used to program and interface with hardware is equally important. When interfacing with hardware sketching platforms such as Arduino[6] or Phidgets[24], most programming languages achieve optimal performance but some languages should simply be off limits[19]. ActionScript and the Adobe Flash Player for example do not have native serial hardware access for security reasons. This means that any sketch that solely uses Flash is incapable of communicating to devices connected to the computer. Therefore, in order to achieve communication between Phidgets or Arduino and ActionScript, an in-between layer of socket servers is required. This completely detracts from the low-level programming feeling. The idea behind sketching in hardware is after all to achieve the partial result of a design in order to visualize its founding concepts simultaneously and in the real world. One of the reasons why Arduino is so popular is because of its easily extensible C API and how easy it is to put programs onto the hardware. Phidgets on the other hand are programmable with no less than 20 programming languages. They offer flexibility in how the hardware is accessed and make the assumption that designers have done the research as to which tool is appropriate for use in the context of their work. Both Arduino and Phidgets are valid options for hardware sketching because they extend quality tools to the designer with an emphasis on freedom of choice and performance.

## 5. CONCLUSION

In the spirit of striving towards great design, I suggest that interaction designers who are serious about their software and

hardware sketches take extra care into creating their software sketches by learning about the programming languages that could potentially drive those sketches. Just like how graphics designers will strive to achieve a pixel perfect design, interaction designers should share the same attention to detail. It is oftentimes worthwhile and not as difficult as perceived to write software sketches by using real-world programming languages and environments. The goal here is not perfect programming form, but optimal performance, a good sketch and great design.

## 6. REFERENCES

- [1] ADOBE, <http://www.adobe.com/products/flashplayer/>
- [2] APPLE, <http://www.apple.com/ipad>
- [3] APPLE, <http://developer.apple.com/>
- [4] APPLE, Introduction to The Objective-C Programming Language, <http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html>
- [5] APPLE, iOS 4.2, <http://www.apple.com/ios/>
- [6] ARDUINO TEAM, <http://www.arduino.cc/>
- [7] Buxton, Bill, (2007). Sketching User Experiences: Getting the Design Right and the Right Design, Morgan Kaufmann
- [8] CYCLING 74, <http://cycling74.com/products/maxmsp/jitter/>
- [9] Ericsson Computer Science Laboratory, <http://www.erlang.org/>
- [10] Fallman, Daniel, (2003). Design-Oriented Human — Computer Interaction, Fort Lauderdale, Florida, USA
- [11] Johnston, David, (2009). Muds, [http://glia.ca/conu/muds\\_09](http://glia.ca/conu/muds_09), accessed 27 February 2011
- [12] KHRONOS, OpenCL – The open standard for parallel programming of heterogeneous systems, <http://www.khronos.org/opencl/>
- [13] Lewis E. J., Nadeau B., Zananiri E., (2007). What They Speak When They Speak to Me, Gallery Oboro, Montreal
- [14] Lount, Peter William, (2004). What is Smalltalk?, <http://www.smalltalk.org/smalltalk/whatis-smalltalk.html>
- [15] LUA, Lua: about, <http://www.lua.org/about.html>
- [16] THE MATHWORKS INC, MATLAB The Language of Technical Computing, <http://www.mathworks.com/products/matlab/>
- [17] McCarthy, John, (1979). History of Lisp, Stanford University, USA
- [18] MICROSOFT, <http://www.microsoft.com/windowseembedded/en-us/evaluate/windows-embedded-ce-6.aspx>
- [19] Moussette C. and Dore F., (2010). Sketching in Hardware and Building Interaction Design: Tools, Toolkits and an Attitude for Interaction Designers, Proc. Of Design Research Society 2010, Montreal (Canada)
- [20] NOKIA, <http://qt.nokia.com/products/>
- [21] OBX LABS, <http://www.obxlabs.net/>
- [22] OBX LABS, Mr. Softie, <http://www.mrssoftie.net/>
- [23] ORACLE, <http://www.oracle.com/us/technologies/java/index.html>

[24] PHIDGETS, <http://www.phidgets.com/>

[25] R, What Is R?, <http://www.r-project.org/about.html>

[26] SCHEMERS, <http://community.schemewiki.org/?scheme-faq-general>

[27] Soulie, Juan (2009), A brief description, <http://cplusplus.com/info/description/>

[28] W3Schools, <http://www.w3schools.com/js/default.asp>