Algorithmic Craft: Tools and Practices For Creating Useful and Decorative Objects With Code

by

Jennifer Jacobs

Submitted to the Department of Media Arts and Sciences
in partial fulfillment of the requirements for the degree of

Masters of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author .....
Department of Media Arts and Sciences
May 18, 2013

Certified by .....
Leah Buechley
Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Algorithmic Craft: Tools and Practices For Creating Useful and Decorative Objects With Code

by

Jennifer Jacobs

Submitted to the Department of Media Arts and Sciences
on May 18, 2013, in partial fulfillment of the
requirements for the degree of
Masters of Science

Abstract

The accessibility, diversity, and functionality of modern computer systems make computer programming (hereafter programming) useful in many realms of human study and advancement. Visual and physical art, craft, and design are interrelated domains that offer exciting possibilities when combined with programming. Unfortunately, use of programming is currently limited as a medium for art and design, especially by young adults and amateurs. Many potential users view programming as highly specialized, difficult, inaccessible, and only relevant as a career path in science, engineering or business fields, rather than as a mode of personal expression. Despite this perception, programming has the potential to correspond well with traditional, physical art-making practices. By forging a strong connection between programming and the design and fabrication of personally relevant physical objects, it may be possible to foster meaningful experiences in both programming and design for novice practitioners. The combination of digital fabrication technologies with computational design serves as one such connection.

Thesis Supervisor: Leah Buechley
Title: Associate Professor

"The type of work which modern technology is most successful in reducing or even eliminating is skillful, productive work of human hands, in touch with real materials of one kind or another. In an advanced industrial society, such work has become exceedingly rare. A great part of the modern neurosis may be due to this very fact; for the human being enjoys nothing more to be creatively, usefully, productively engaged with both his hands and his brains."

E.F. Schumacher, Small is Beautiful (1973)

Acknowledgments

This is the acknowledgements section. You should replace this with your own acknowledgements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computation is a driving force in our world. The power and ubiquity of modern computer systems have made the skill of computer programming (hereafter programming) relevant to a wide range of human studies and disciplines. Most commonly, programming is viewed as an essential component of science, engineering, and business related applications[**?**]. As a result, many nascent programmers view programming as highly specialized, difficult, inaccessible, and only relevant as a career path in those particular fields. In reality, computation is a broad discipline with many applications, ranging from professional to personal. Foremost programing can serve as a medium for personal expression, through applications in art and design. With the emergence of digital fabrication technology, In addition, programing provides the means to design and produce useful objects and devices, not only on at an industrial scale, but also on a personal and individual scale [**?**]. When programing is used create unique, functional physical objects, new possibilities emerge in the way people design, the types of objects people create, and role programing can play in peoples' lives. Computational design, the practice of programming to create form, structure and ornamentation, is a new way to design. When paired with digital fabrication technology, computational design allows people to make physical objects by writing code. My objective is to examine the combination of computational design, digital fabrication and traditional arts and crafts for the production of functional decorative objects. I define this domain with the term algorithmic craft. In this thesis,

I will define the affordances of algorithmic craft and describe the development and dissemination of three tools to support novice practitioners in this domain. The process of bridging the spaces between textual programming language, visual design, and physical construction however, is not self-evident and raises many practical and theoretical questions. What are the important design principles to consider when creating programming environments for physical design? How do we compellingly link textual code with visual designs and what are the appropriate intersection points between textual manipulation and visual manipulation? What support is required to help people move back and forth from programming to building real objects in a way that is comfortable, expressive and pleasurable? How can we remove the technical challenges in translating code into an object that can be successfully fabricated, while still supporting a wide variety of design styles, aesthetics and approaches? Finally, how can we interlink the often disparate processes of physical prototyping with digital design and programming in a way that creatively reinforces both physical and virtual modes of working?

Outline general thesis here...

Chapter 2

Motivation and Background

"The mathematicians patterns, like the painters or the poets must be beautiful; the ideas like the colours or the words, must fit together in a harmonious way. Beauty is the first test: there is no permanent place in the world for ugly mathematics."

G. H. Hardy [**?**]

2.1 Computational Design

The practice of computational design is fundamentally different from contemporary design. Due to the multitudes of approaches among different designers, it ultimately futile to attempt to describe a single standard design process. It is however useful to point out several key features of computational design that stand in contrast with the conventional design.

Both computational and conventional designers begin with a design problem. Conventional designers often proceed by roughing out a number of specific solutions to this problem. These early solutions are evaluated against one another for their successes and drawbacks, and from this evaluation a smaller set of more refined solutions are produced. This iterative process may continue, often through the incorporation of outside feedback, until a single solution is reached that is sufficiently refined and successful in addressing the initial problem. This highly simplified summary approximately describes the conventional iterative design process. Computational design

incorporates many of the iterative principles of conventional design, but differs significantly in its approach. Rather than begin by developing a concrete initial solution to the initial design problem problem, the computational designer must first formalize the elements of the problem into a set of rules. The designer then creates a system based on these rules that capable of producing a variety of solutions, depending on the input criteria it is given. In its simplest form, this system may consist of a single algorithm with static input and limited output solutions. More frequently however, the computational design process produces complex systems that act on upon a wide range of input criteria and parameters, and can produce nearly infinite number of design solutions. Iteration in computational design then takes the form of incremental adjustments to the system. Naturally, many of the solutions produced by an initial system fail to address original design problem. By sampling a number of outputs from a system, the designer can "tweak" or make adjustments to the rules that govern the system, eventually resulting in more and more desirable output solutions. This process of sampling and tweaking is continued until the designer is satisfied with a given range of outputs. The designer can then vary the input to the system and can select among the resulting solutions.

While the process of computational design can be distinguished from conventional design, the two fields are compatible with one another. Aside from a wholly different approach, computational design can also be considered as a means of extending traditional design practice through several key affordances: These include the following:

- **Precision:** Computation supports high levels of numerical precision with relatively little effort on the part of the designer.

- **Automation:** Computation allows for rapid automation of repetitive tasks. Automation often plays a key role in enabling the development and transformation of complex patterns and structures, through the combination of large numbers of simple elements in an ordered and structured manner.

- **Generativity and randomness:** Computation allows for the programmer to create algorithms which when run, allow for the computer to autonomously

produce unique and often unexpected designs.

- **Parameterization:** Computation allows users to specify a set of degrees of freedom and constraints of a model and then adjust the values of the degrees of freedom while maintaining the constraints of the original model [**?**].

- **Documentation and remixing:** Computationally generated designs are generated by a program, which can be shared with and modified by other designers. Because these programs are often text-based, they also serve as a form of documentation of the design process.

In combination with these affordances however, computational design also incorporates a number of challenges in the design process that are not present in traditional design:

- **Formalizing complex problems** As design problems grow in complexity, formalizing the problem in a manner that can be expressed programmatically becomes increasingly challenging. Writing an algorithm to generate a visual pattern is relatively simple, however writing a program to incorporate that pattern into the design of an entire garment is non-trivial.

- **Creating singularities:** A designer will often choose to deviate from a set pattern or structure at specific points in order to create a special emphasis in that area. Because computational design is governed by a systematized ruleset, the methods of breaking these rules at arbitrary points is are often unclear and tedious to implement.

- **Selecting a final design:** The systematic approach to computational design gives the designer the ability to produce extremely large numbers of solutions to a single design problem. While this is useful in situations where multiple solutions are required, when a single design must be chosen, the process of deciding on a solution is often difficult and sometimes arbitrary, especially if the decision is based on aesthetic criteria.

2.2 Digital Fabrication

Although computational design must be conducted on a computer to some degree, the artifacts generated by computational design are not restricted to the screen. Digital fabrication technology provides the opportunity to translate programmatically generated designs to physical form. Digital Fabrication is the process of using computer-controlled machines to fabricate objects specified by a digital design file or tool path. The machines that encompass digital fabrication range from 3D printers, laser cutters, and computer numerically controlled (CNC) milling machines, to vinyl cutters, CNC embroidery machines and knitting machines, and even inkjet printers. Digital fabrication shares many of the affordances of computational design. In particular, it allows for the creation of physical objects of great complexity without formal skill in craft or extensive manual labor. Digital fabrication also allows for the rapid production of small volumes of similar or identical objects. Lastly, because the artifacts produced through digital fabrication are derived from digital files, anyone with access to the file, and a similar fabrication machine can create a copy of the object, or incorporate elements of it into a new design.

Digital fabrication is also compelling for its own reasons. Currently, digital fabrication machines are rapidly decreasing in price and increasing in availability [**?**]. As a result, we are seeing the emergence of personal fabrication, wherein sophisticated manufacturing technologies are becoming available to regular people [**?**]. Excluding personal 3D printers which are generally limited to a few varieties of ABS plastic, most personal fabrication machines can work with a wide range of materials. Laser cutters work well with traditional materials such as wood, paper and cloth. Vinyl cutters can also be used on cloth and paper, as well as cut vinyl patterns which can be used for screen printing. The current stage of personal fabrication is estimated to be at the same place as personal computing in the 1970s [**?**] add section on computer aided design

2.3 Algorithmic Craft

The conjunction of computational design and digital fabrication has the potential to allow individuals to use programming to express their aesthetic concerns in the creation of objects. This is important because aesthetic expression through design is a substantial part of intellectual development [4] and an important part of peoples lives. FaThese machines offer the potential to extend and innovate traditional forms of design, constructing and crafting by allowing for greater levels of automated complexity and precision in physical objects, and correspond well with the practice of programming. citation from rosner arcticle craft vs design: A central element of these and other visions of the future is that craft is done for us: Kitchens tell us what and how to cook, eliminating the creativity and pleasure of cooking from scratch with whats on hand; object printers create flawless prototypes, eliminating messily glued-together chipboard and toothpicks. In this new world, craft becomes fetishthe proudly displayed collection of vinyl records shelved alongside an iPod and digital files [1].

importance of materials- craft as domain of materiality, design and programing as abstractions in many cases, but deal very directly with materiality when applied to the real world- craft offers direct connection to this - citation from rosner article craft vs design

2.4 Barriers

rename section heading Insert section about mythology of fabrication as personal replication contrasting with craft aspects maker culture (fab-Gershenfeld pg 5)- and views as purely hobbyist practice (Tanenbaum) perhaps expand into section on barriers and perceptions?

Chapter 3

Related Tools and Research

There are numerous forms of CAD software and programing environments. Within the realm of computational design and digital fabrication, there are 4 primary categories of existing tools that directly relate to my study of computational design and digital fabrication: professional computational-design tools, entry-level programing environments, and novice-oriented computer-aided-design (CAD) tools. While certain qualities are shared between these categories, several key distinctions exist between each group of tools.

3.1 Professional Computational Design Tools

A couple of forms of professional computational-design tools exist. Foremost, many popular graphic-user-interface (GUI) CAD applications include a feature that allows the user to automate certain elements of the program through scripting or programing. For example, in Adobe software like Photoshop and Illustrator, it is possible to write JavaScript-based programs to automate various application procedures. Similarly, 3D modeling tools such as Maya and Blender feature the ability to script behaviors in languages that are syntactically similar to Perl and Python respectively. This scripting is usually omitted from the primary menus and interfaces of the applications that feature it. There are also professional tools that are explicitly developed for computational design. The most prominent example is Grasshopper, a third-party

add-on for the Rhinoceros 3D modeling tool. Grasshopper is a data-flow programing environment that lets users combine a variety of modules and blocks to create and adjust 3D models in Rhino. A textual coding module is also available and allows users to integrate C# scripts using the Rhino API into their patch, although the user must have an understanding of the basic principles of programing in order to effectively use this functionality.

DesignScript, a more recent computational design tool, developed by Autodesk, is a domain specific text-based programing environment and language that contains methods to generate and manipulate geometric models that are compatible with existing Autodesk applications. DesignScript itself functions as an add on to the Autodesk AutoCad software. DesignScript is intended for use by experienced designers and 3d modelers who posses a range of programming expertise. The language syntax is based on C#, however it features the ability to operate in both associative and imperative paradigms, in an effort to support a pedagogical transition between basic and more complex forms of computational design [**?**].

Lastly, OpenSCAD is a script-based constructive solid geometry modeling tool developed specifically for CAD applications. OpenSCAD contains a custom programing language in which the user can create descriptions of 3d models in a textual format, and display them by compiling the script. This scripting behavior provides the user with precise control over the modeling process and enables the creation of designs that are defined by configurable parameters, however this control comes at the cost of requiring the user to be familiar with textual programing an scripting. In fact, OpenSCAD is explicitly developed for programmers and relies on textual input exclusively as the mechanism for design, In addition, unlike the prior tools mentioned, OpenSCAD is both free and open source, and many variations and derivatives of it exist [**?**].

In the context of digital fabrication, one of the most important elements of these professional tools is their ability to import and export a wide variety of file formats, thus facilitating the transitions between a digital design and the required file type for a specific fabrication tool. Despite their power, and due to their high cost and complex

feature set, these professional tools are extremely difficult for amateurs to access and use. It is also important to note that with the exception of OpenSCAD, the examples listed are only available as plugins or add-ons or are developed to supplement an existing graphical tool, rather than serve as the primary method of design. In some cases this status as a form secondary functionality adds a set of practical barriers to independent use. The scripting tools in illustrator and photoshop are difficult to locate, Grasshopper only functions on Windows versions of Rhino, and Design Script requires the prior purchase of AutoCAD to operate. Although these practical barriers can be overcome, their existence often prevents less experienced users from gaining access. In addition, the positioning of computational functionality as secondary to the primary method of design points to a larger ideological classification of these forms of design as a specialized and exclusive, rather than a primary method of design.

3.2 entry-level CAD Tools

A subset of CAD tools have also been created that are designed to be more accessible to a wider range of people. These tools provide an option for individuals who lack the experience and access to professional level tools, however they also provide an opportunity for more casual participation in CAD. SketchUp is a 3d modeling tool developed by Google to enable easier forms of 3D modeling. Although SketchUp was not explicitly created to allow people to design for CNC and digital fabrication, several 3rd-party add ons exist that allow users to export designs to file formats that are compatible with a variety of fabrication machine [**?**]. TinkerCad is another 3d modeling tool designed for entry level users. As opposed to SketchUp, TinkerCad is explicitly developed to assist in designing for 3d printers and has built in functionality to allow users to export their designs to the .stl format which is compatible with 3D printing [**?**]. AutoDesk has also produced several entry level 3d-modeling applications as a part of their 123D series. Many of these applications are designed to interface with digital fabrication, including 123D Make which allows users to convert stock or uploaded 3d models into a series of flat parts which can be fabricated on 2-axis

machines like laser cutters, and 123D Creature, which enables users to design a variety of creatures from a set of basic parts and then order a 3d printed model of their finished creature [**?**]. AutoDesk Research has also developed MeshMixer, an application for the intuitive merging and manipulating high resolution triangle meshes. MeshMixer was released to the public and has since become a popular 3d design tool for hobbyist 3D printer users. All of the entry level tools listed above vary in their specific approach to creating more accessible forms of CAD. In general they feature a trade off between limited functionality and power, in favor of a simplified tool set and an easier learning curve. Despite these restrictions, it is possible to use these entry level tools to develop highly complex and sophisticated models show example image of mesh mixer model. A more serious limitation of these tools is their ephemerality. Because entry level CAD tools are often free, and more frequently web based applications, it is common for them to suddenly become unavailable or no longer supported by the company that produces them. Tinkercad serves as a recent example of this wherein the parent company decided to transition to focusing on professional-level CAD tools and as a result, closed down the Tinkercad website and cut off access to the application footnote about tinkercad recently being acquired by autodesk. Several of these entry-level tools feature some form of scripting or programing functionality. A plugin for Sketchup allows users to automate certain actions by using the Ruby-based SketchUp API. TinkerCad allows users to create Shape Scripts, which are parametric models defined by javascript code. MeshMixer has an C++ API which is not yet publicly available, but is provided to interested parties upon request. While these computational tools suggest compelling possibilities, similar to the professional level tools listed above, they are positioned as secondary ways of interacting, and are much less deliberate than the primary features of the application.

3.3 Learning-Oriented programming tools

In addition to entry level CAD tools, a number of tools and applications have been created to introduce inexperienced programmers to the realm of computer science.

Logo, a computational drawing program, serves as the seminal novice programming language founded on principles of constructionism and embodiment [9]. The Scratch visual programming language is a notable successor to Logo, and allows users to create interactive projects by combining command blocks rather than writing textual code [11]. Alice is another programming environment that relies on visual programming, but is targeted towards an older user group than Scratch [3]. Turtle Art [16] and Design Blocks [2] are two visual programming languages inspired by Logo that are designed specifically for visual composition. Processing is a text-based programming environment designed for easy learning, and directed toward artists, designers, and inexperienced programmers [10]. Logo, Turtle Art, Design Blocks, and Processing facilitate computational drawing and, therefore, can be viewed as computational-design environments. There remains a gap, however, between novice-oriented programing environments and the novice-oriented CAD tools. With the exception of FlatCAD, none of the examples listed bridges the space between computational design and CAD for digital fabrication. Processing Scratch Alice Logo

3.4 Novel Fabrication and CAD tools

In addition to these tools, there are a number of research projects involving novel forms of fabrication and software tools. Most of these tools are highly domain specific, however they provide insight into the development of more general approaches for computational design and Digital Fabrication Laser Origami Midas Free-D Sketch Chair KidCad FlatCad

/Garment-Creation CAD and Fabrication tools Sensitive Couture parsing patterns into 3d garments Sketch Based Garment Design

Chapter 4

Design Objectives

Practical objects Open ended design options Low Barrier to entry Targeted audience

Chapter 5

Design Tools

Codeable Objects Motivation and design principles Tool Description Workflow description Workshop Workshop results

Soft Objects Motivation and design principles Tool Description Workflow description Workshop Workshop results

DressCode Motivation and design principles Tool Description Workflow description Workshop Workshop results Curriculum building Curriculum results

Chapter 6

Discussion (rename)

Processes- planning vs experimentation Prototyping The role of craft The affordances of algorithmic fabrication The aesthetics of computational design Critique in computational design

Chapter 7

Future Directions

Version Control (The loss of design) Better selection mechanisims Longer-term studies Targeted audience (revised)

Chapter 8

Conclusion

Appendix A

Tables

Table A.1: Armadillos

Armadillos	are
our	friends

Appendix B

Figures

Figure B-1: Armadillo slaying lawyer.

Figure B-2: Armadillo eradicating national debt.