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
June something, 2013

Certified by .....
Leah Buechley
Associate Professor of Media Arts and Sciences
Thesis Supervisor

Certified by .....
Mitchel Resnick
Academic Head, Program in Media Arts and Sciences
Thesis Supervisor

Certified by .....
Neri Oxman
Assistant Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by .....
...
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 June something, 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 of Media Arts and Sciences

Thesis Supervisor: Mitchel Resnick
Title: Academic Head, Program in Media Arts and Sciences

Thesis Supervisor: Neri Oxman
Title: Assistant Professor of Media Arts and Sciences

"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

To come....

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> *"Some people write poetry in the language we speak. Perhaps better poetry will be written in the language of digital computers of the future than has ever been written in English."* -J.C.R Licklider

Computation is a driving force in our world. The power and ubiquity of modern computer systems have made the skill of computer programming relevant to a wide range of human studies and disciplines. Commonly, programming is viewed as an essential component of science, engineering and business. Despite this view, computation is also a powerful tool for creative disciplines. When applied to the arts, programming offers different paradigms for creation and new methods of problem solving. My personal interest in this respect lies in computational design: the use of programming to generate visual form. Frequently used to create screen-based work, computational design is also applicable to the design of physical objects. The advent of digital-fabrication technology allows for a transition from digital forms produced by computation to physical artifacts that live in the world. The joint use of computational design and digital fabrication is becoming an integral component of professional art and design, both as a method of prototyping and as a means for producing finished work. Although these technologies primarily serve professionals, the combination of computational design and digital fabrication also provides the opportunity for amateur creative expression. The growing diversity in programming environments, combined with the emergence of the personal fabrication movement suggest new ways to support non-professionals in a combined practice of computation, design and construction. In addition, many forms of digital fabrication are compatible with traditional art and craft techniques. The combination of computational design and digital fabrication provides a way of making programming relevant to people with an interest in craft. The development of accessible computational-design and digital-fabrication tools therefore presents new opportunities for broadening participation in programming, expanding the ways in which people create, and changing the role programming can play in people's lives.

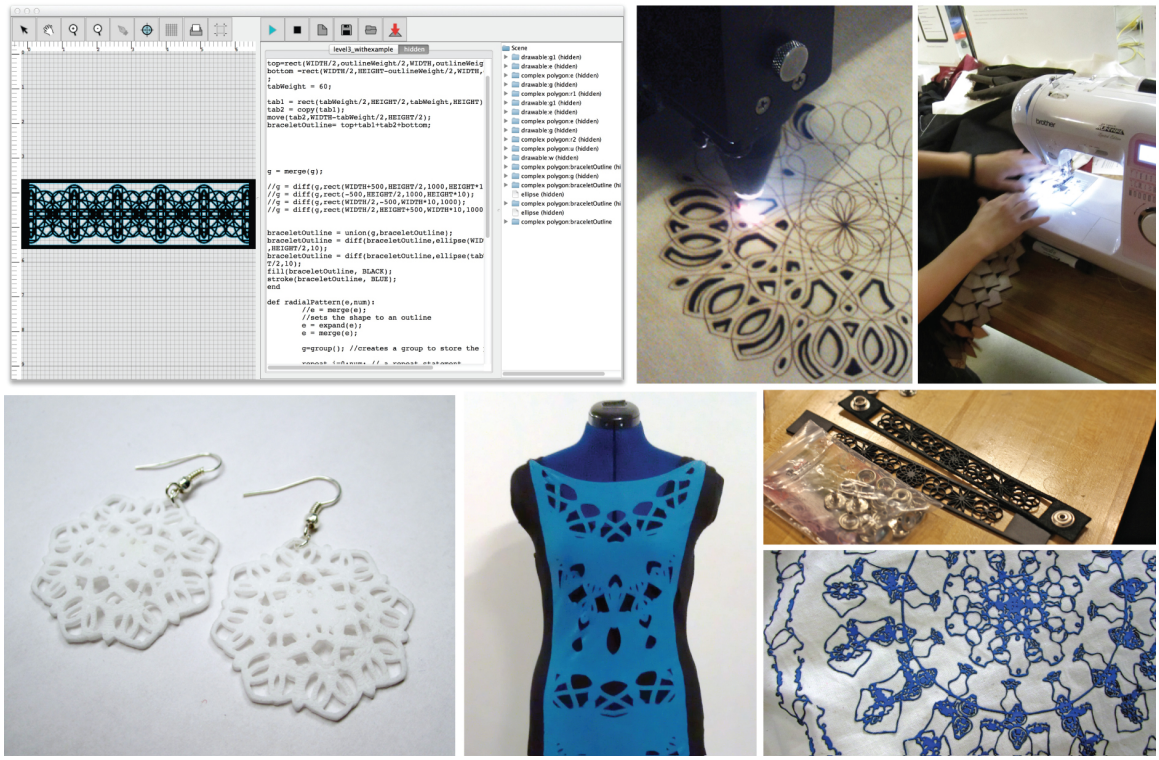


Figure 1-1: Algorithmic craft practices, tools and artifacts.

In this thesis, I examine methods that facilitate participation in the intersection of computational design, digital fabrication and traditional arts and crafts for the production of functional and decorative physical artifacts. I use the term algorithmic craft as a way of describing these collective practices. My primary aim is to support people without formal education or experience in computer science in the practice of algorithmic craft, although I also seek to develop methods that are accessible to individuals without formal training in art and design. My goal is to work with people in ways that emphasize pleasure, recreation and non-professional utility in the service of personal expression.

My study of this space is motivated by the practical and theoretical questions that arise when bridging the spaces between textual programming language, visual design, and machine and hand-based physical construction. 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?

In order to address these questions, I developed three different software tools to support practitioners in this domain. Codeable Objects is a programming library that helps people design and fabricate laser-cut lamps. Soft Objects is an extension of Codeable Objects that allows people to use programming to create forms and patterns for fashion and garment design. DressCode is a combined graphic-design and programming environment with a custom programming language, developed to support open-ended computational design for digital fabrication. Each of these tools were evaluated in workshops where people used them in combination with digital fabrication machines to produce physical artifacts. Through the workshops, I examined several different approaches for introducing novices to computational design. In addition, I evaluated how a variety of crafting techniques could be combined with Computer Aided Design (CAD). The lessons learned in developing and testing Codeable Objects and Soft Objects were used to inform the development of DressCode. This thesis presents the evolution of these three tools, and describes the rationale behind their design. I conclude with a discussion of some of the unique affordances of algorithmic craft, and the possibilities it offers for creative expression in the future. this may change.. be sure to check

Chapter 2

Background and Motivation

"The mathematician's patterns, like the painter's or the poet's 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 [4]

In order to illustrate the creative potential of algorithmic craft, it is useful to examine its contents: computational design, digital fabrication and craft. Each of these disciplines have distinct strengths and limitations. Through their convergence, new, compelling forms of creation become possible. In computational design, the abstract qualities of computation provide a powerful way of thinking about design. With appropriate programming, a computer can embody any conceivable process [**?**]. Similarly, computational design can be applied to any design domain, be it physical, visual, sonic, or interactive. Although individual applications of computational design often consider material properties that are relevant to a specific domain, computational design does posses an inherent connection to the material world. Craft on the other hand is closely linked to materiality. Craftspeople work in close contact with the physical world and craft artifacts and processes are directly informed by material properties. Digital fabrication provides a connection between the abstraction of computational design and the material domain of craft, by converting digital concepts to physical forms. In the following section I discuss the properties of computational design, digital fabrication and craft in greater detail and describe connection points between each discipline.

2.1 Computational Design

As previously mentioned, computational design can refer to a range of design practices that incorporate computational processes. Computational processes are explicit patterns of rules that can manipulate data [**?**]. Computer programs are description of computational processes in a format

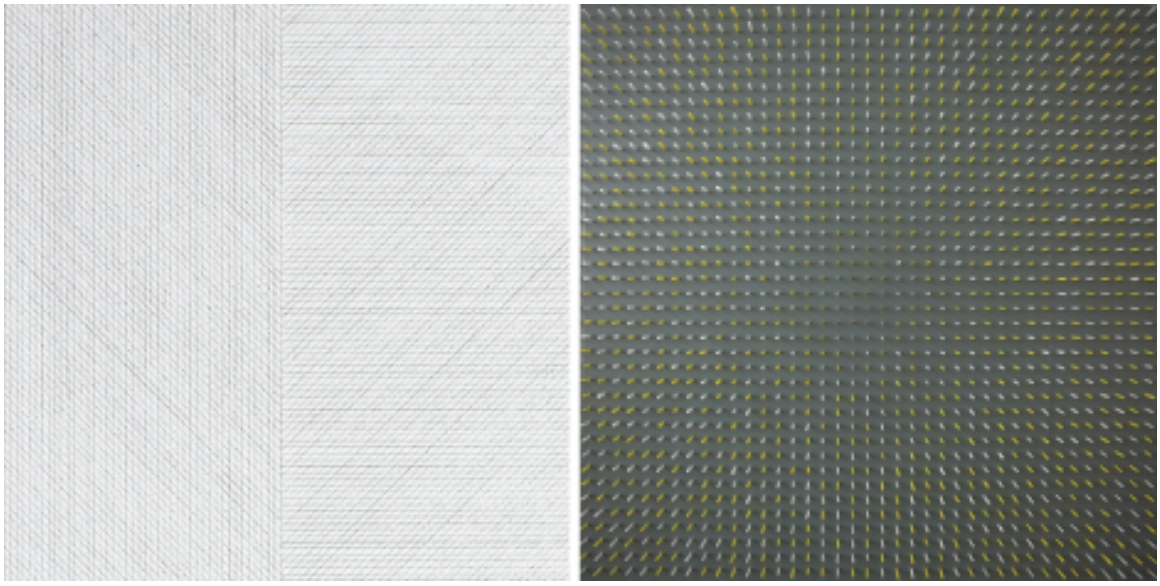


Figure 2-1: Wall drawings by Sol Lewitt. From left to right: Wall Drawing 19: A wall divided vertically into six equal parts, with two of the four kinds of line directions superimposed in each part (1969), Wall Drawing 38 (Detail): Tissue paper cut into 1.5-inch (4 cm) squares and inserted into holes in the gray pegboard walls. All holes in the walls are filled randomly (1970).

that is readable by a computer. My focus in computational design is the process of using computation processes to create visual forms and patterns. Because computational design usually requires the designer to write programs, it is possible to mistake the practice of computational design as a technical skill rather than a way of thinking [6]. It is true that computational design does require a degree of technical expertise; practitioners must be able to write and read code. Despite this requirement, computational design is better represented as a way of applying procedural thinking to a design task. Rather than producing specific design representations, the designer authors a set of rules that define a system capable of producing many outcomes. Designs are presented in abstract terms, resulting in the potential to create multiple variations that share a set of constraints. One of the challenges of computational design is in authoring guidelines that effectively produce solutions within the desired design space. Presented as an ordered set of instructions, these guidelines constitute an algorithm. Arguably, it is possible to engage in computational design without using a computer. The artist Sol Lewitt created a number of artworks consisting of a set of instructions that dictated the constraints of an artwork:

> WORK FROM INSTRUCTIONS (1971):
> USING A BLACK, HARD CRAYON DRAW A TWENTY INCH SQUARE.
> DIVIDE THIS SQUARE INTO ONE INCH SQUARES.
> WITHIN EACH ONE INCH SQUARE, DRAW NOTHING, OR DRAW A DIAGONAL STRAIGHT LINE FROM CORNER TO CORNER OR TWO CROSSING STRAIGHT

LINES DIAGONALLY FROM CORNER TO CORNER.

The resulting piece would vary depending on how the instructions were interpreted by the people executing them. Many of his other artworks contained the instructions for producing them in the title of the piece (figure:2-1.) Using a computer to execute computational design algorithms as opposed to human beings greatly accelerates the process of execution, and supports a non-linear design process. The specific affordances of computational design are as follows:

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

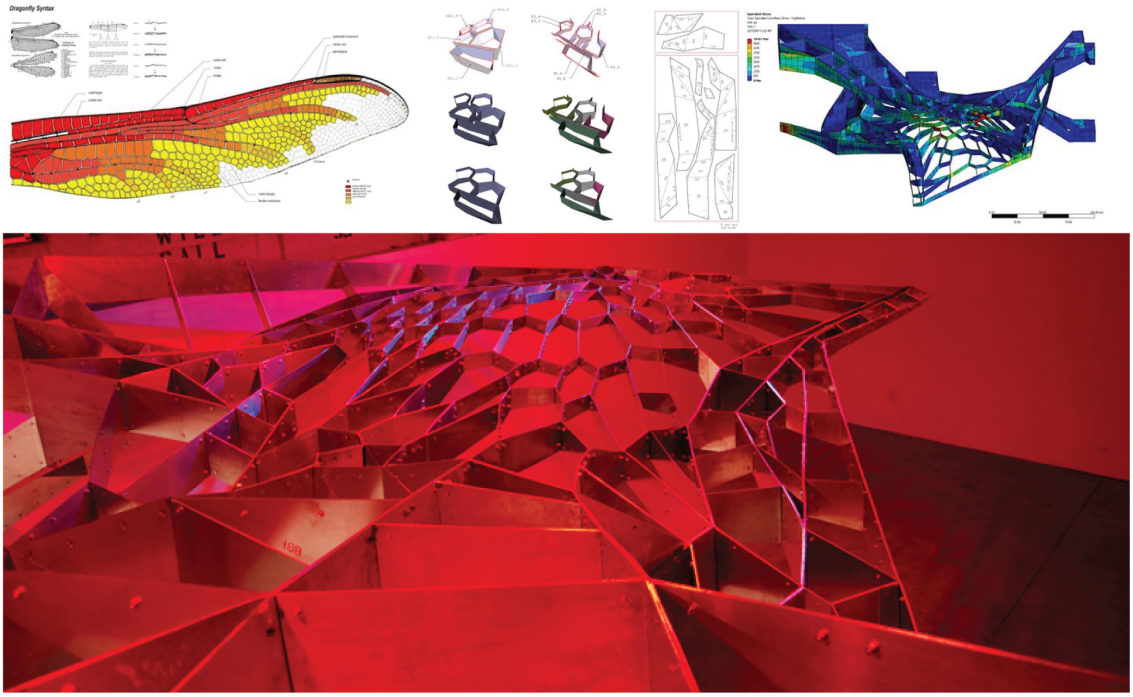


Figure 2-2: Dragonfly by EMERGENT/ Tom Wiscombe (2007). Installation. Inspired by the structure of a dragonfly wing, this installation was shaped in by a set of parameters including gravity, specific support points and flat material properties. The structure was generated based on support and loading data, which was run through a feedback loop, producing a design which supported both the performance criteria and a novel variation of form [**?**].

- **Visual Complexity:** Computational design supports the creation and transformation of complex patterns and structures through rapid automation and iteration which allows for the combination and manipulation of large numbers of simple elements in a structured manner.

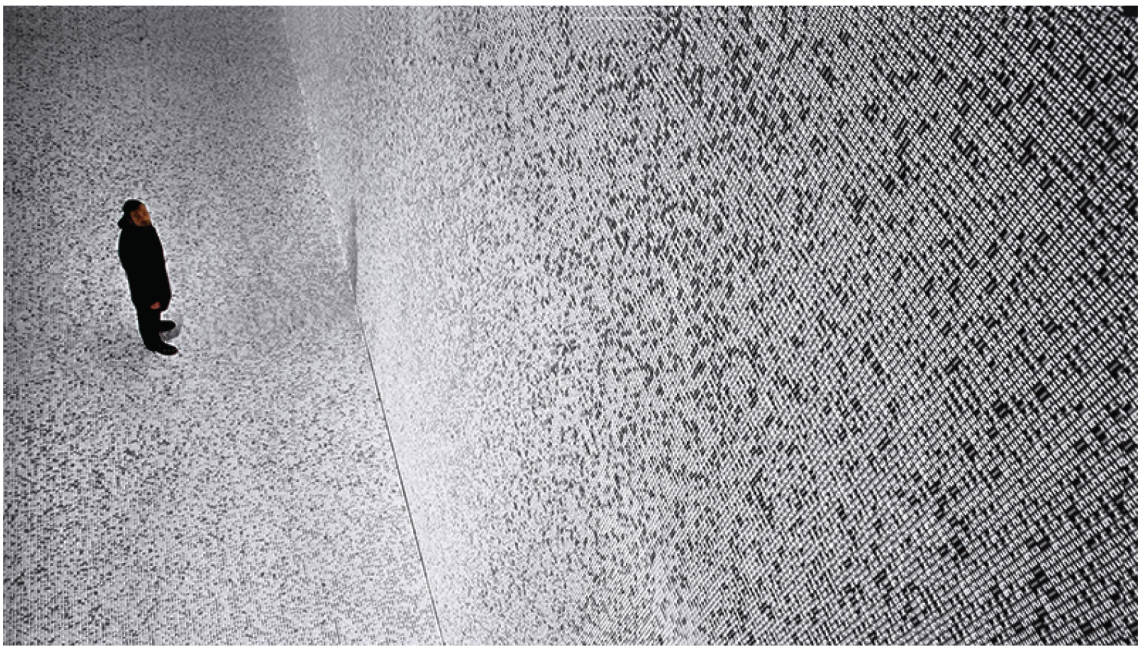


Figure 2-3: data.tron by Ryoji Ikeda (2008). Dynamic computer visual and sound installation. Each pixel of the visual image is dynamically calculated as a visualization of data from hard drive errors and software code. [**?**].

- **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.

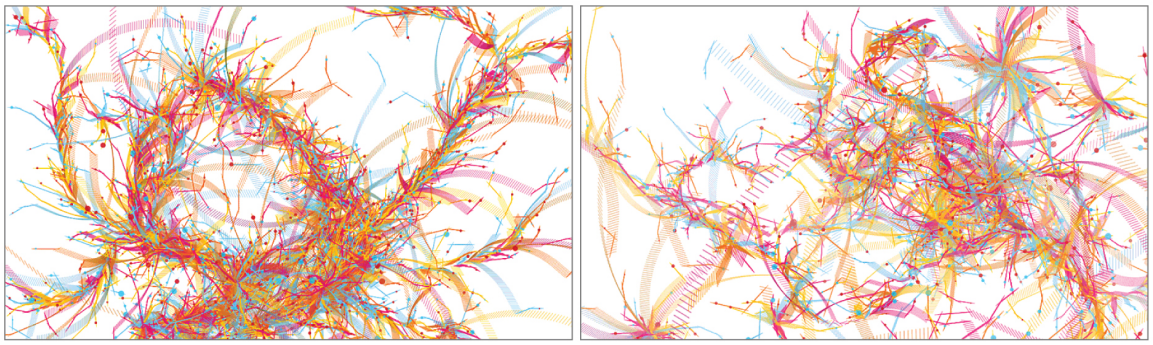


Figure 2-4: Abstract01.js by Marius Watz (2010). Interactive. The piece is comprised of a java script program that autonomously creates unique abstract compositions based on a set of visual guidelines specified by the artist [**?**].

- **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.

Figure 2-5: ArachnÈArmor / Corset by Neri Oxman (2012), Digital Materials Centre Pompidou, Paris, France. The form of the web on the piece is determined by the anatomical location of a person's rib cage [**?**].

- **Documentation and remixing:** Computationally generated designs are generated by a program, a document which can be shared with and modified by other designers. These programs also serve as a form of documentation of the design process itself.

In combination with these affordances however, computational design also contains a number of unique challenges:

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

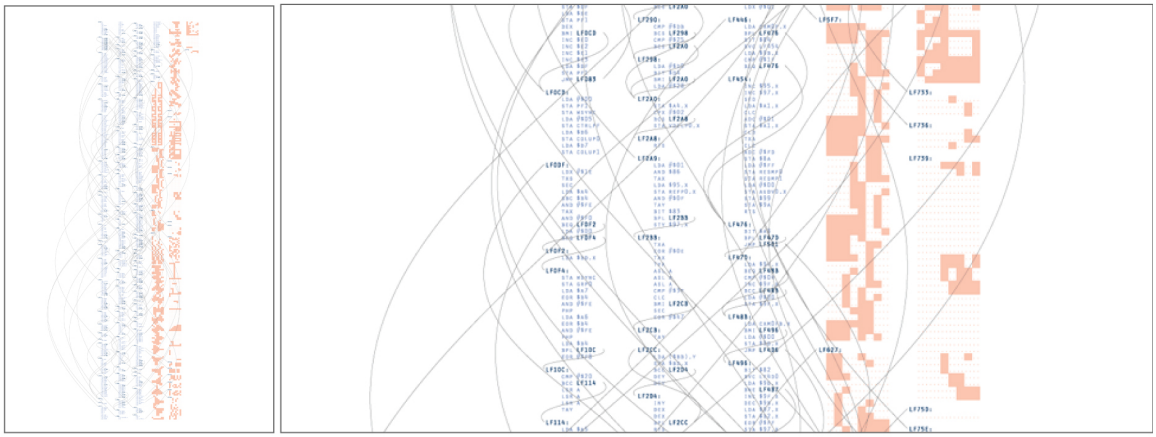


Figure 2-6: Contra from Distellamap Series by Ben Fry (2005) Digital Print. The Distellamap series translates the code from a number of Atari2600 video games into a series of visualizations. Assembly code from each cartridge is shown in blue, with connection lines denoting "go to" statements. Data is shown in orange [**?**].

- **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 is performed on a computer, the artifacts generated by computational design are not restricted to the screen. Digital fabrication technology provides the opportunity to translate digital files to physical form. Digital Fabrication is supported through computer aided design (CAD) technology. CAD involves the use of a software tool to design an object. Some form of CAD file is generally required to control a digital-fabrication machine [**?**]. Digital Fabrication is the process of using computer-controlled machines to fabricate objects specified by a digital tool path. Digital fabrication shares many of the affordances of computational design. In particular, it allows for the creation of physical objects with a high degree of complexity, without skill in craft or extensive manual labor. Digital fabrication also allows for the rapid production of small volumes of similar or identical objects. Also, because the artifacts produced through digital fabrication are derived from digital files, anyone with access to the file, and a similar fabrication machine can potentially create a copy of the object, or remix that object with others.

There are two primary forms of digital-fabrication manufacturing, additive and subtractive. Subtractive processes machine a part by removing pieces from the original material and include tools

like laser cutters, computer numerically controlled (CNC) milling machines and vinyl cutters. Additive processes create a part by incrementally adding successive layers of material. 3D printers are most commonly associated with additive manufacturing, however ink jet printers and CNC embroidery machines also fit this definition. Although advances in additive technology are occurring at a rapid pace, at this point the material options for 3D printing are limited. Most 3D printers can output objects in plastics, ceramic and metal composites [**?**]. Low-end 3D printers are even more constrained, and can only print in ABS plastic. Additive manufacturing is also extremely expensive and often constrained to small build volumes.reference wood printing 3D printers, other novel 3D printing techniques

insert picture of 3d printed and subtractive pieces

Unlike most 3D printing technology, subtractive processes can work with a wide range of materials [**?**]. Laser cutters work well with materials such as wood, leather, paper and cloth. Vinyl cutters can also be used on cloth, paper, in addition to adhesive vinyl. Milling machines can cut parts from plastic, foam, metal and wood. In specialized cases these machines can also be modified to function in an additive manner with custom attachments. An example is the adaptation of a 3-Axis ShopBot to function as a threading device to facilitate part of the construction of a Silk Pavilion, created by the Mediated Matter Group at MIT (figure 2-7 [**?**].) Large-scale subtractive fabrication machines also make it possible to work at a larger scale than additive processes [**?**].

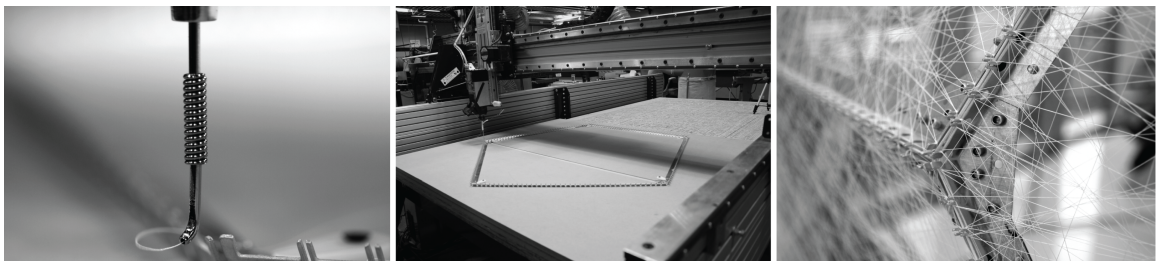


Figure 2-7: 3-axis CNC milling machine adapted as CNC deposition tool using a custom threading tool. (Part of the Silk Pavilion Fabrication process)

Widespread access to digital fabrication is growing. Digital fabrication machines are rapidly decreasing in price and increasing in availability [**?**]. This trend has allowed an increasing number of individuals and small groups to gain access to sophisticated manufacturing technologies, and signaled the growth of personal fabrication [**?**]. Consumer 3D printers can now be purchased at prices ranging from $1,000-$3,500 dollars. Small scale laser cutters and milling machines are available at prices ranging from $3,000 to $5,000. add in citations for prices In addition, other groups are producing open-source versions of commercial fabrication equipment, like the MTM Snap CNC-milling machine, the LaserSaur laser cutter and the RepRap 3D printer. Though these devices require additional levels

of expertise to assemble, they point to to exciting future developments in affordable and open forms of digital fabrication. Machines like inkjet printers and craft CNC-vinyl cutters also function as personal fabrication devices and are extremely accessible and affordable.

There are also options for individuals without direct ownership of a fabrication machine. Community hacker spaces like Artisans' Asylum in Cambridge, NYC resistor in New York and Noisebridge in San Francisco, and organizations like the FabLab network provide access to shared digital fabrication facilities. Online services like Shapeways, Ponoko and SpoonFlower offer on-demand fabrication services to individual consumers for a variety of materials and machining processes.

The rise of personal fabrication is often viewed as a component of the maker movement [**?**]. The maker subculture is a technology-literate community that is engaged in the production of devices and artifacts in line with the do-it-yourself (DIY) philosophy. While maker culture has a strong technological focus, it also encompasses hobbyist craft techniques and materials. MakerFaire, one of the predominate maker culture events, showcases a range of DIY practice across science, engineering, art, and craft [**?**]. Some of the primary tenants of the maker movement correspond to values frequently expressed in craft.

2.3 Craft

The cultural connotations of craft have varied throughout history. Once the primary means of producing functional objects, the role of craft changed following the industrial revolution. In the face of mechanized production, hand skills became less central to production, and design, traditionally unified with the role of the artisan emerged as a separate discipline [**?**]. Despite the emergence of mass production, craft has endured, both as a recreational pursuit, and as set of valued artisanal practices. My personal interest in craft focuses on three specific qualities: materiality, pleasure and craftsmanship.

- **Materiality:** One primary aspect of craft involves the manipulation of physical materials [**?**]. Working with physical materials requires the use of one's hands, and is often an intuitive process. When we craft, we experience the feel of the paint brush moving across the canvas, the carving knife through a piece of wood, a needle through cloth, or our fingertips pressing into clay. The decisions we make in the craft process are altered by the feel of working with the material.

  insert picture of material evidence of craft (clay?)

- **Pleasure:** One of the responses to industrialism was the Arts and Crafts movement, initiated primarily by William Morris, and inspired by the writings of John Ruskin. The arts and crafts movement sought to restore the aesthetics and practices of traditional craftsmanship, in

response to the negative aesthetic effects and working conditions of industrialism. Fundamental to the arts and crafts movement was the notion that the act of creating beautiful artifacts with one's hands was a pleasurable, and essential human experience [**?**]. This emphasis on pleasure is retained in conceptions of craft today.

- **Craftsmanship:** Although not all craft is functional, many forms of traditional craft, including sewing, pottery, and carpentry can be applied the creation of useful objects. In addition to this functionality, craft often emphasizes the importance of beauty in the form and ornamentation of objects. I use the term craftsmanship to describe the successful unification of aesthetics and utility in a single artifact.

insert picture of decorative object from arts and craft movement

2.4 Algorithmic Craft

Algorithmic craft is the combined use of computational design, digital fabrication and hand craft. The merging of the properties of craft with computational design and digital fabrication allows for a creative practice that exhibits the variability and complexity of computation, the precision and repeatability of fabrication and the material, functional and aesthetic concerns of craft. Algorithmic crafting allows individuals use programming and digital fabrication as a means of pleasurable and useful creative expression. In addition, algorithmic craft encourages the incorporation of the values of artisans and craftspeople in development of new methods of digital fabrication and computational design tools, paving the way for new forms of innovation in these fields. Below I describe several examples of combing digital fabrication and craft, and computational design and craft respectively.

2.4.1 Digital Fabrication and Craft

Objects that emerge from a combined practice of digital fabrication and craft are often characterized by their physical craftsmanship, technical mastery and exceptional aesthetic. Practitioners in this space blur the boundaries between engineer, designer and artisan. One of the advantages of merging craft and digital fabrication is that it ensures that machine-produced artifacts retain their personal history. For his Hybrid Reassemblage pieces, Amit Zoran blended handcraft techniques like basket weaving and slip casting with 3D-printed forms. The process resulted in artifacts that retained the evidence of hand production in conjunction forms made possible through computational design and digital fabrication (figure: 2-8)[**?**].

Another advantage of combining digital fabrication and craft is that it affords openness and flexibility in the design of functional components. Peter Schmitt's plywood servo is a re-designed RC

Figure 2-8: A selection of vases from the Hybrid Reassemblage series by Amit Zoran (2010). Glazed ceramic, SLS nylon element, epoxy glue and black spray paint.

servo in laser-cut plywood. Although the parts of the servo are digitally designed, after fabrication the completed piece is constructed by hand. This process of hand construction gives the designer greater control in the form-factor and aesthetic of the servo, while maintaining its functional qualities [**?**]. As one sample application, the plywood servo was incorporated into a custom drawing machine called the Audiograph, producing a surprising and novel object with no precedent. The objective of the audio graph was to create a device with an appearance that was drastically different to customary home electronics and explore people's reaction to it in a home setting (figure: 2-9)[**?**].

2.4.2 Computational Design and Craft

Similar to artifacts developed through digital fabrication and craft, the combination of computational design and craft also offers the opportunity to produce pieces that are connected to a distinct space, time and process, and shaped by material properties, while incorporating patterns and forms made possible through computational processes. The Random Number Multiple series was a project that produced screen prints from the work of computational designers and artists. Figure 2-10 shows a screen print translation of the computational artist Jer Thorpe's RGB- NYT word frequency piece. The original piece was a digital image, which was generated from data on the usage of the words red', green', blue' in the New York Times between 1981 and 2011. In translating the digital piece to physical form the result is distinct physical object, transformed by the material properties of screen printing. The artist describes the process on his website:

Figure 2-9: Plywood Servo by Peter Schmitt (2011). (Clockwise from upper-left: 3d CAD model of partially constructed servo, laser cut plywood parts and electronic components, Audiograph, completed servo.)

> *"This print turned out even better than I could have expected. The fine detail is amazing, the colours are rich and vivid, and the half-toning on the individual bars creates a jewel-like halo in the center that is fascinating to look at up close..This print was made with a semi-reflective ink, so it has a unique shimmer to it when viewed in the light."*

The rich material qualities Thorpe describes would be difficult to achieve by relying solely on a digital medium. In addition Thorpe describes pleasure he experienced in working with the tactile process of screen printing, and how the experience provided important creative feedback for his computational work [**?**].

Combining computational design and craft also offers the opportunity to engage craftspeople in the use of computational tools. The Processing quilt (figure 2-11, was the result of a collaboration between Libs Elliott, a seamstress and quilt-maker, and Joshua Davis, a computational designer. Elliott created the quilt based on a processing program created by Davis which generated random triangle compositions. Elliott selected one specific composition and manually translated it to a grid and cut out the necessary pieces, and crafted a completed quilt (figure: 2-11)[**?**].

Figure 2-10: RGB NYT Word Frequency by Jer Thorpe (2011). Part of the Random Number Multiples Series. Screen Print. (Counter-clockwise from upper-left: matching paint color to the digital image, close up of print, complete print.)

The resulting artifact is synthesis between computational aesthetics of the triangle composition, and the texture created by the quilted pattern. The piece exhibits the skills of Elliott and the computational expertise and style of Davis. The transition between computational forms and craft processes can be a source of frustration however. On her website, Elliot remarked about the contrast between working with a computational tool and producing something by hand:

> *"I'm not going to lie- quilting is a slow process and there's a certain level of frustration I have when I'm doing it. It's difficult to take something that works so quickly, like the Processing tool, to generate a hundred random compositions within an afternoon then force yourself to pick one image and slow it all down to an entirely manual process that takes days to produce [**?**].*

The contrast in execution time between computational design and hand craft can be addressed by the introduction of digital fabrication. By combing all three of computational design, craft and digital fabrication into a single creative process, practitioners can more easily translate the qualities of of computational forms (including complexity, generativitiy, and precision) to physical form in a variety of materials. These physical forms in turn can be shaped through craft processes, and imbue computational designs with individuality, and utility. Several forms of digital fabrication machines are capable of producing forms that are suitable for both expert and novice levels of handcraft.

Figure 2-11: Processing Quilt by Libs Elliott and Joshua Davis (2013). (Counter-clockwise from upper-left: auto-generated triangle compositions in Processing, hand-made quilt grid, Finished quilt.)

Because subtractive fabrication technologies work with materials like wood, paper and fabric, they support the creation of objects that are readily shaped by the human hand. Laser cut parts can be sanded, polished, painted, sewn or folded after they emerge from the machine. Vinyl cutters can also be used on cloth and paper. They can also produce patterns in adhesive vinyl that can be applied to screens for for screen printing. Laser cutters and vinyl cutters also fabricate at a much faster rate than 3d printers or milling machines, and therefore results in a higher tolerance for error during the craft process; it is feasible to quickly re-cut damaged parts.

2.5 Challenges in broad participation in computational design and digital fabrication

Through its association with hobbyist culture, many people consider certain forms craft to be accessible, and open to amateurs. Conversely, the combined use of computational design and digital fabrication is largely limited to experts and professionals. There are many practical barriers to novice participation in this domain. Most prominently, many of the programming languages used for computational design are difficult for novices to learn. In addition, a complex and convoluted process is required to translate a code-based design to a format that is compatible with fabrication

machines [**?**]. Furthermore, the engineering challenges involved in designing complex objects from multiple digitally fabricated parts are extremely difficult to tackle for amateurs who are unfamiliar with CAD software. Although novice oriented CAD software exists, most of this software does not support computational design. Similarly, most novice oriented programming environments cannot produce designs that are suitable for fabrication on their own.

There are also significant perceptual barriers to participation. There persists among the general public a limited perception of the applications of programming. Many people consider programming to be irrelevant to their interests, and therefore lack motivation to pursue what they perceive to be a highly specialized and difficult undertaking [5]. There are also perceptions of digital fabrication which may hinder creative and open use of these tools. Personal fabrication technology is often portrayed as a precursor to the production of replicator-like technology which can instantiate literally anything by building it directly from atoms [**?**]. This view can also act as a barrier to widespread engagement with existing forms of digital fabrication, by setting up unreal expectations for this technology,or by portraying it as a variation on traditional forms of consumerism. The idea of fabrication as a perfect replication system also eliminates the need or desire for human engagement in the fabrication process, eliminating the entry points for craft. Daniella Rosner describes this view in her research:

> *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 what's 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 [**?**].*

Algorithmic fabrication is an exceptional form of creation. Through the unification of computational design, digital fabrication and hand production, one can create functional, beautiful and unique objects. If we are to extend practice of algorithmic craft beyond people with formal training in computer science and CAD, however, it is necessary to create accessible and open tools for programming and digital fabrication. These tools must be supported with practices that foster an understanding of the principles of computation, craft and digital fabrication and how they can be applied towards personal creative pursuits.

Chapter 3

Related Software Tools and Research

Computational design and digital fabrication are supported through the use of a variety of Computer Aided Design (CAD) software. There are six primary categories of existing software that are relevant to the study of the convergence of these fields:

1. programming tools for computational design.
2. Professional CAD tools with computational-design functionality.
3. Entry-level CAD tools.
4. Entry-level programming environments.
5. Novel fabrication and CAD tools.

While some tools fall into several of these categories, it is useful to highlight the general functional distinctions and design strategies of each group of tools.

3.1 Programming tools for computational design

Many of the examples of computational design referenced in section 2.1 were created through the use of custom software tools written in a wide variety of programming languages, libraries and environments. Arguably, computational design can be performed with any programming language that has the capability to output some form of graphic data. In order to facilitate the process, some people have developed programming libraries and toolkits explicitly for computational design. Here I list some of the most prominently used tools in this space and describe their relation to digital

fabrication. openFrameworks is a popular open source C++ toolkit developed by Theo Watson and Zach Lieberman that is designed to support a number of creative applications of programming. The toolkit contains wrappers for a variety of commonly used C++ libraries including OpenGL, GLEW, GLUT, libtess2 and cairo for graphics and Assimp for model loading, and facilitates a more immediate use of these libraries in conjunction. Although openFrameworks is frequently used to create interactive screen-based projects, several extensions have been developed which allow for the import and export of .STL model data, making it feasible to use the toolkit to design for digital fabrication [**?**].

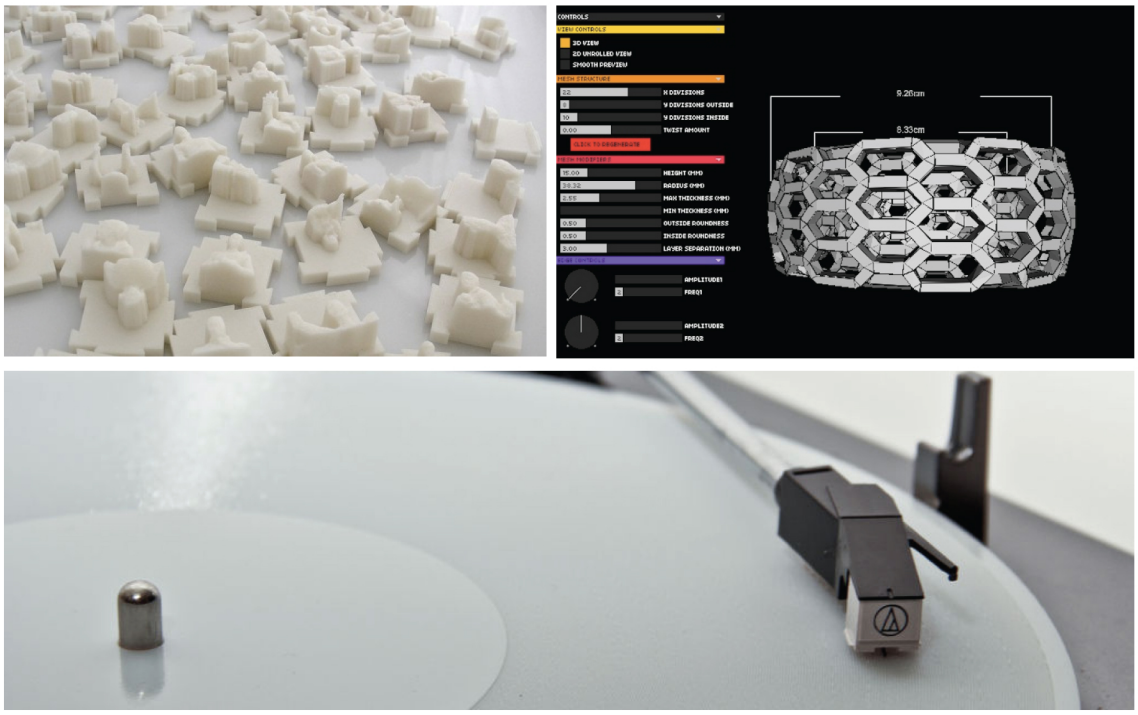


Figure 3-1: Clockwise from upper-left: Fabricate Yourself by Karl D.D. Willis, 3d printed puzzle pieces featuring conference attendees (created with the openFrameworks toolkit, Kinect and a 3d printer), Cell Cycle Bracelet and Ring creation application by Nervous Systems (created with Cinder toolkit), 3d Printed functional record by Amanda Ghassaei, (created using Processing.)

Cinder is another C++ toolkit for computational graphics creation. Cinder was developed by the Barbarian Group, and requires greater programming expertise to use than openFrameworks [**?**]. Similar to OpenFrameworks, Cinder is frequently used to create interactive installations, but it has been used for digital fabrication oriented projects. Nervous Systems used Cinder to develop their Cell Cycle iPad app, which creates cellular bracelets and rings for 3D printing [**?**].

Processing is a java-based programming language and development environment created by Casey Reas and Ben Fry. Processing is unique in that it is described as an entry level programming environment but is also an extremely successful professional computational-design tool. Processing

contains several libraries that allow it to export to PDF and DXF formats, which enable a translation to digital fabrication [**?**].

All of these examples use textual programming as the sole means of design. In addition, because openFrameworks and Cinder exist as toolkits, they require the user to write programs in general purpose programming environments such as Visual Studio or Xcode. In the hands of experienced programmers, these tools facilitate remarkable forms of creative expression through code. Because of the difficulty of independently learning textual programming, it is uncommon for people without prior programming experience to use them for computational design without a great deal of instruction and support, let alone apply them to digital fabrication applications. Due to its unique positioning, Processing can act as an exception, something I discuss in greater greater detail in the section on entry-level programming environments.

3.2 Professional CAD tools with computational-design functionality

Some professional CAD applications include functionality that makes it possible to use the software for computational design. 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 programming. 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. In all of these examples the programming interface is omitted from the primary interfaces of the application, and must be deliberately activated by the user.

Some professional tools are more explicitly developed for computational design. One of the most prominent examples is Grasshopper, a third-party add-on for the Rhinoceros 3D modeling software. Grasshopper is a data-flow programming 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 program.

DesignScript, a more recent computational design tool, developed by Autodesk, is a domain specific text-based programming environment and language that contains methods to generate and manipulate geometric models that are compatible with existing Autodesk applications. DesignScript is an add-on to the Autodesk AutoCad software and cannot be run independently. 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 advanced levels of programming [**?**].

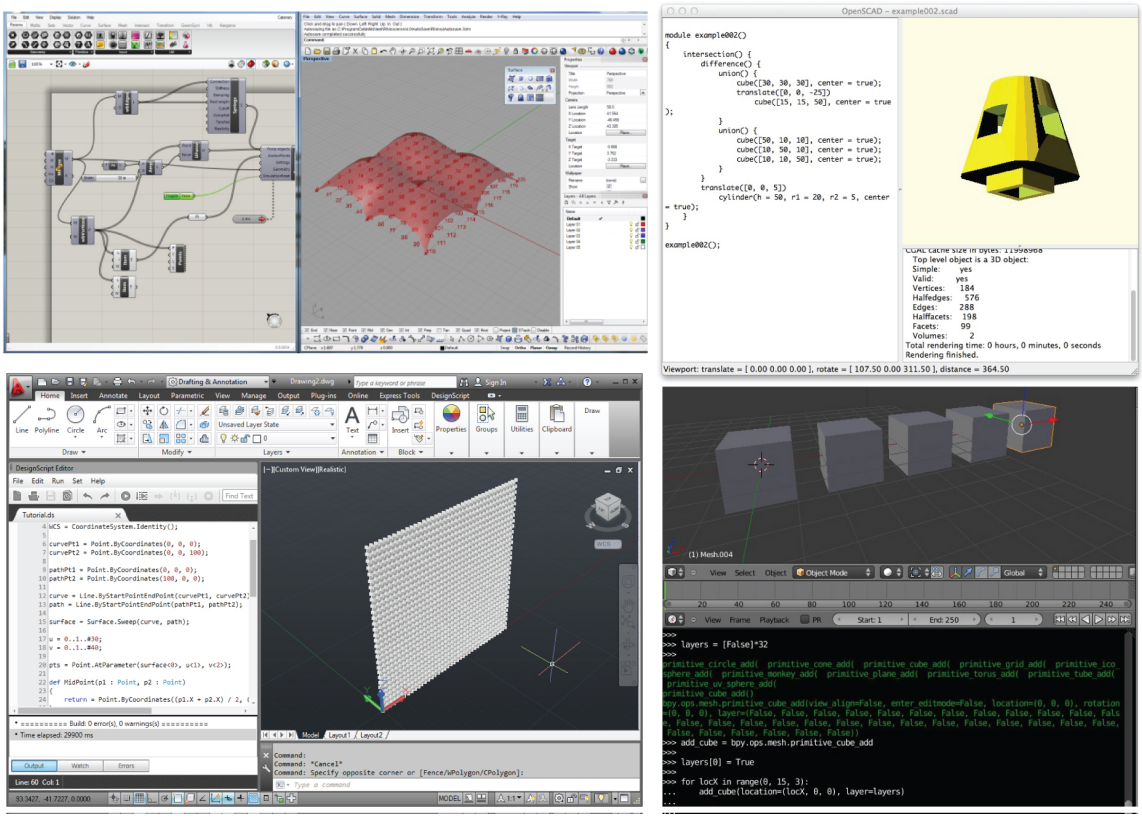


Figure 3-2: Computational design in professional tools. Clockwise from upper-left: A Grashopper program in Rhino, OpenSCAD, the Python scripting interface in Blender, DesignScript.

OpenSCAD is a script-based constructive solid geometry modeling tool developed specifically for CAD applications. OpenSCAD contains a custom programming 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 programming and scripting. OpenSCAD is explicitly developed for experienced programmers and relies on textual input exclusively, In addition, unlike the prior tools mentioned, OpenSCAD is both free and open source, and several variations and derivatives of it exist [**?**].

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. Because to their high-cost and complex feature set, these professional tools are difficult to use without prior experience or extensive practice. With the exception of OpenSCAD, another defining feature of these tools is that they only available as

plugins, add-ons or are developed to supplement an existing graphical tool, rather than serve as the primary method of design. This status of programming as secondary form of interaction poses a practical barrier to novice use of the computational functionality of these tools; the scripting tools in illustrator and photoshop are difficult to locate, Grasshopper must be custom installed along with several dependencies, and only functions on Windows versions of Rhino, and Design Script requires the prior purchase of AutoCAD to operate. While these practical barriers can be overcome, their existence often prevents less experienced users from gaining access. More importantly, 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 serving as a primary method of design.

3.3 Entry-level CAD Tools

A subset of CAD tools have 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 digital fabrication, several independently developed 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 pressing limitation of these tools is their ephemerality. Because entry

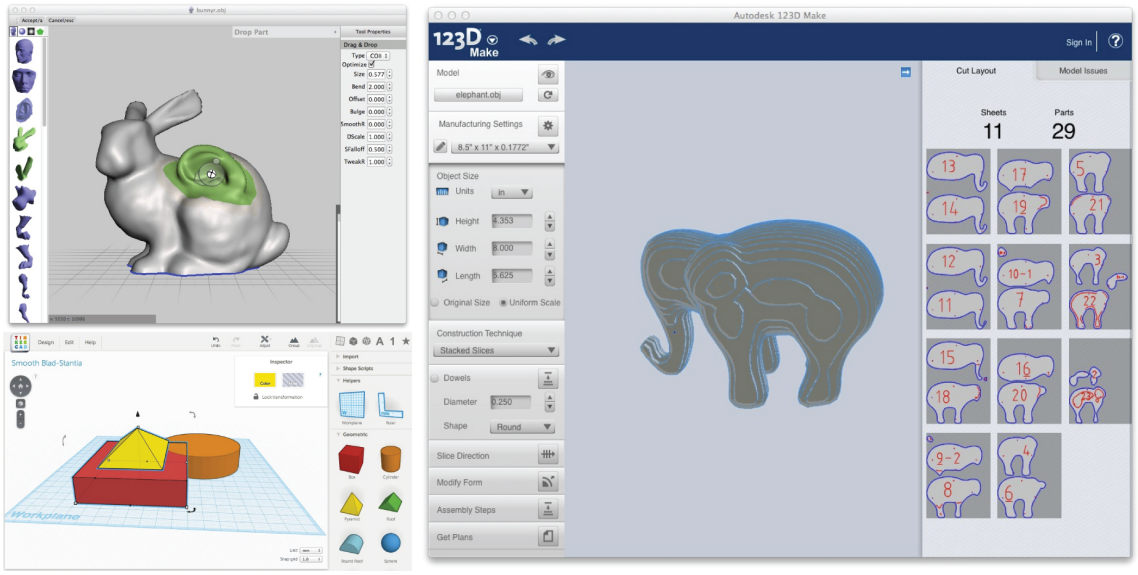


Figure 3-3: Entry-level CAD tools. Clockwise from upper-left: MeshMixer, 123D Make, TinkerCad.

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. TinkerCad was recently acquired by AutoDesk and its website restored, however its future is uncertain.

Several of these entry-level tools also feature some form of scripting or programming 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 features suggest compelling possibilities, similar to the professional level tools listed above, they are positioned as secondary ways of interacting, and are designed less deliberately than the primary features of the application.

3.4 Entry-level programming environments

Similar to entry level CAD tools, a number of wonderful applications have been created to introduce new 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 [**?**]. Logo allows users to direct the movements of a virtual "turtle" with textual commands. The turtle can leave a trace of where it has been with a PENDOWN command, enabling users to write programs from an intrinsic perspective which produce geometric forms and patterns

[?]. 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 [?]. Although Scratch is often used as a medium for interactive storytelling, it can also contains blocks that enable pen-up and pen-down functionality and can be used as a drawing tool similar to Logo. Turtle Art [?] and Design Blocks citedesign blocks are two visual programming languages inspired by Logo that are designed specifically for visual composition.

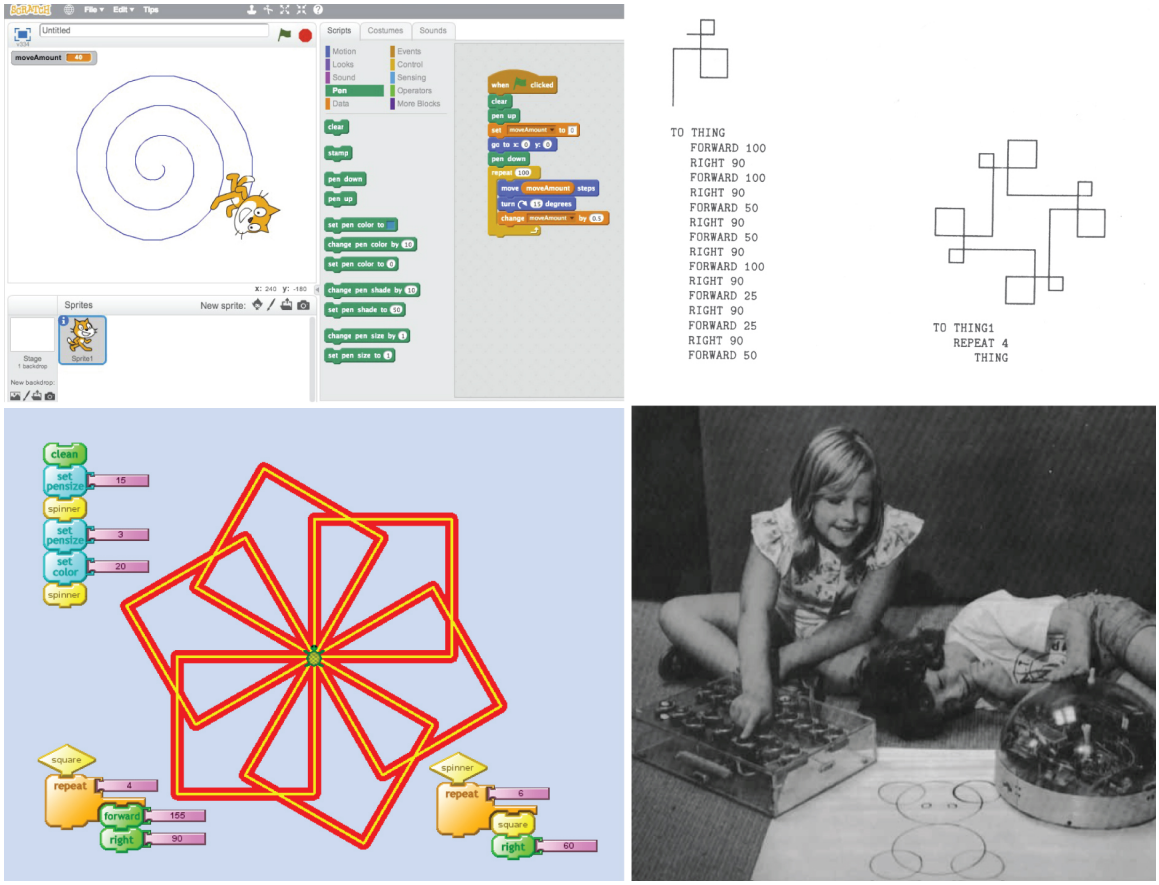


Figure 3-4: Entry-level forms of computational design. Clockwise from upper-left: A Scratch program using the pen tool, A design made in Logo by rotating a simple doodle, the Logo physical drawing robot, a sample Turtle Art program.

Logo, Scratch, Turtle Art and Design Blocks all support forms of computational drawing and, therefore, can be viewed as computational-design environments. There remains a gap, however, between novice-oriented programming environments and the novice-oriented CAD tools. In direct contrast to the novice oriented CAD tools described in the preceding section, although learning oriented programming tools can provide an excellent platform for generating digital computational design work, they often lack explicit features for generating and exporting designs that are compatible existing modes of digital fabrication. Although it is used by professionals, there have been many

efforts to make Processing suitable for entry level programmers as well. The Processing drawing application programing interface (API) is simpler than other programing environments, and many instructional materials and example programs are provided on the processing website to support novice use. These features in conjunction with the processing libraries that enable the export of digital-fabrication-compatible file formats make Processing a more viable option for entry level computational design for digital fabrication. Despite this, Processing is not explicitly designed for digital fabrication. In practice it is difficult for new programmers to use it to design objects compatible with digital fabrication machines. If we wish to open this space for entry level practitioners, it is essential to design tools with computational design for fabrication as the primary function.

3.5 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 that demonstrate new approaches for computational design and digital fabrication. Sketch It, Make It is a 2D CAD tool that allows users to constrain their designs through gestures made using a digital drawing tablet [**?**]. Spatial Sketch is a tool that allows users to create abstract 3D sketches via their gestures, and then translates the sketches into a set of slices, which can be fabricated and combined into a finished piece [**?**]. SketchChair allows users to design their own chair by sketching with a computer stylus [**?**]. The resultant design can then be cut on a computer-numerical controlled (CNC) milling machine and assembled into a 3D object. SketchChair includes a simulation tool that allows users to test the usability of their chairs before they cut them. FlatCAD seeks to connect programming and digital fabrication and allows users to build customized construction kits with a laser cutter by programming in FlatLang, a novice-oriented programming language modeled on Logo [**?**]. Spirogator is a processing based tool that allows users to digitally customize a set of hypotrochoid geared-drawing tools and then view a simulation of those tools in action. The user then has the option of either exporting the resulting design generated by the digital gears and fabricating it directly, or exporting the file paths for the gears themselves, and fabricating them on a laser cutter, to be used as physical drawing tools [**?**].

These examples share several important elements. They are restricted to a relatively narrow domain, but still support a wide range of design variations and personal expression. They contain intuitive and familiar methods of interaction often in the form of drawing and moving sliders. They contain explicit features for making the process of digital fabrication easier for new practitioners, and reduce the possibility of creating designs that will be infeasible to fabricate or are physically unstable. Spirogator and Sketch Chair's simulation tools are particularly interesting in this regard, as they assist the user in predicting some of the behavior of the resultant physical artifact prior

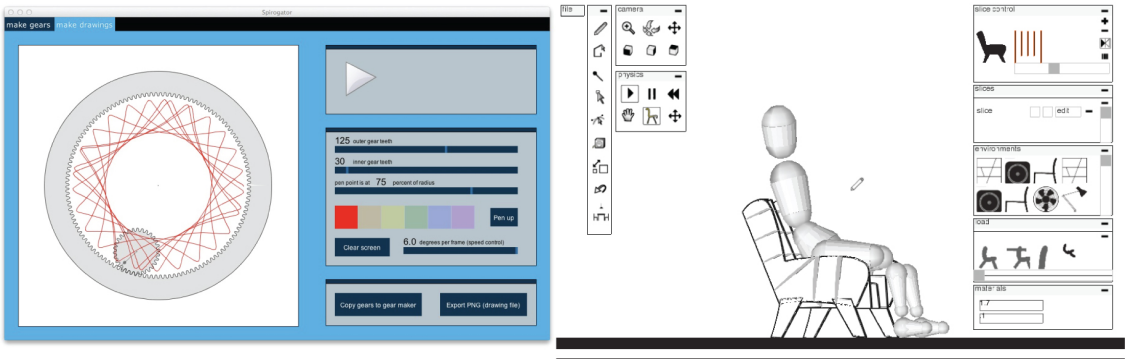


Figure 3-5: Novel CAD tools: from left to right: Spirogator, SketchChair.

to its fabrication. These qualities of domain-specificity, design flexibility, intuitive interaction and practical support for fabrication are important properties for any novice-oriented design software.

images of novel cad tools

Chapter 4

Objectives and Evaluation Criteria

As indicated by the analysis of existing CAD and computational design tools, many options exist to support novice entry into computer science and new tools are emerging that provide different ways of engaging in CAD and digital fabrication. At this point however, there are a lack of tools, which attempt to bridge accessible programing and novel forms of digital fabrication in relevant and engaging contexts for amateur programmers. Digital fabrication and computational design are two highly compatible domains with great creative potential. When combined with the values, practices, and aesthetics of craft, these shared practices offer new forms of expression and making. The objective of this thesis was to foster engagement in algorithmic craft by researching the relationship between programing, visual design, and physical construction. I conducted this research through the development of software tools that facilitated the combined practice of digital fabrication, computational design, and craft. In creating these tools, I emphasized computation by positioning programing as the primary method of generating and manipulating designs. In doing so however, I attempted to retain a high degree of accessibility and intelligibility for new programmers. Upon the basis of this general objective, I generated a set of evaluation criteria for any prospective algorithmic crafting software. A successful tool should produce the following results:

- **Allow users to successfully create physical artifacts:** The artifacts themselves should be durable and useful.
- **Afford a wide degree of variation in design and expression:** The personal aesthetic preferences of the creator should be apparent in the resultant artifact.
- **Enable people to understand the functionality and utility of the programs they write:** Individuals should emerge from the process with a general understanding of some of the key components of computer programing, with an ability to articulate how these components function in their design.

- **Allow users to create objects and designs they would have difficulty generating with conventional techniques:** The tool should support the affordances of computational design, specifically precision, visual complexity, generativity and stylistic abstraction, as well as enabling people to take advantages of the properties of digital fabrication including manufacturing speed, precision, and physical complexity.
- **Engender in users a positive, enjoyable experience:** Use of the tool and subsequent crafting activities should be pleasurable and intellectually engaging.
- **Foster a sense of confidence:** After working with the tool, people should have increased confidence in their ability to successfully program, design, and use digital fabrication tools.

4.1 Design Tools and Evaluation Methodology

Over the course of my thesis, I developed and tested three software tools to support algorithmic craft. Codeable Objects is a domain specific programing library for the design and production of lamps. Soft Objects is an expanded version of Codeable Objects aimed at computational fashion design. DressCode is a general-purpose integrated programing and visual design environment. Each tool was evaluated during one or more workshops with designers, artists, programmers and young people. I documented each workshop through pre and post surveys, interviews, and photographs of student projects. The surveys were aimed at understanding participants' previous experience in craft, programming and design, their interest in and attitudes toward craft, digital fabrication, and computation (before and after the workshops), and their engagement in and enjoyment of the workshops.

Pre-surveys were administered at the start of the workshops and focused on participants previous experience and attitudes. They also asked students to describe their opinions about how programing and craft could be combined, and how they felt programming could extend or limit creativity. Post-surveys were administered at the termination of the workshops and contained attitudinal questions that were matched to the pre-surveys. In addition, post surveys contained a range of written questions asking the participants to describe their opinion of the success of their projects and their experience using Codeable Objects, Soft Objects or DressCode respectively.

Individual interviews were conducted with the participants in the Soft Objects workshop, and the DressCode workshop. These interviews lasted an average of 15-30 minutes and were audio recorded and transcribed. During the interviews, the participants were asked to describe their experience in the workshop and talk about the process of conceptualizing, designing and producing their artifacts. They were asked to describe what they enjoyed, what was difficult for them, and what they felt they had learned through this process. Survey and verbal interview responses and project outcomes were then analyzed to determine if the essential qualities outlined in the evaluation criteria were

achieved. We also used this information to identify recurring and prominent themes in participants experiences. In the following three chapters, I detail the development, feature set and evaluation of each design tool.

Chapter 5

Codeable Objects

Codeable Objects is computational design tool that allows people to design and export the toolpaths for a laser cut lamp. Lamps possess an established function, but can vary immensely in their form-factor. As a result, the domain of lamp design offers a wide set of design possibilities for a concrete and useful end product. By introducing lamp design in the context of algorithmic craft, my

goal was to allow people to construct unique lamps comprised of computationally generated forms and patterns.

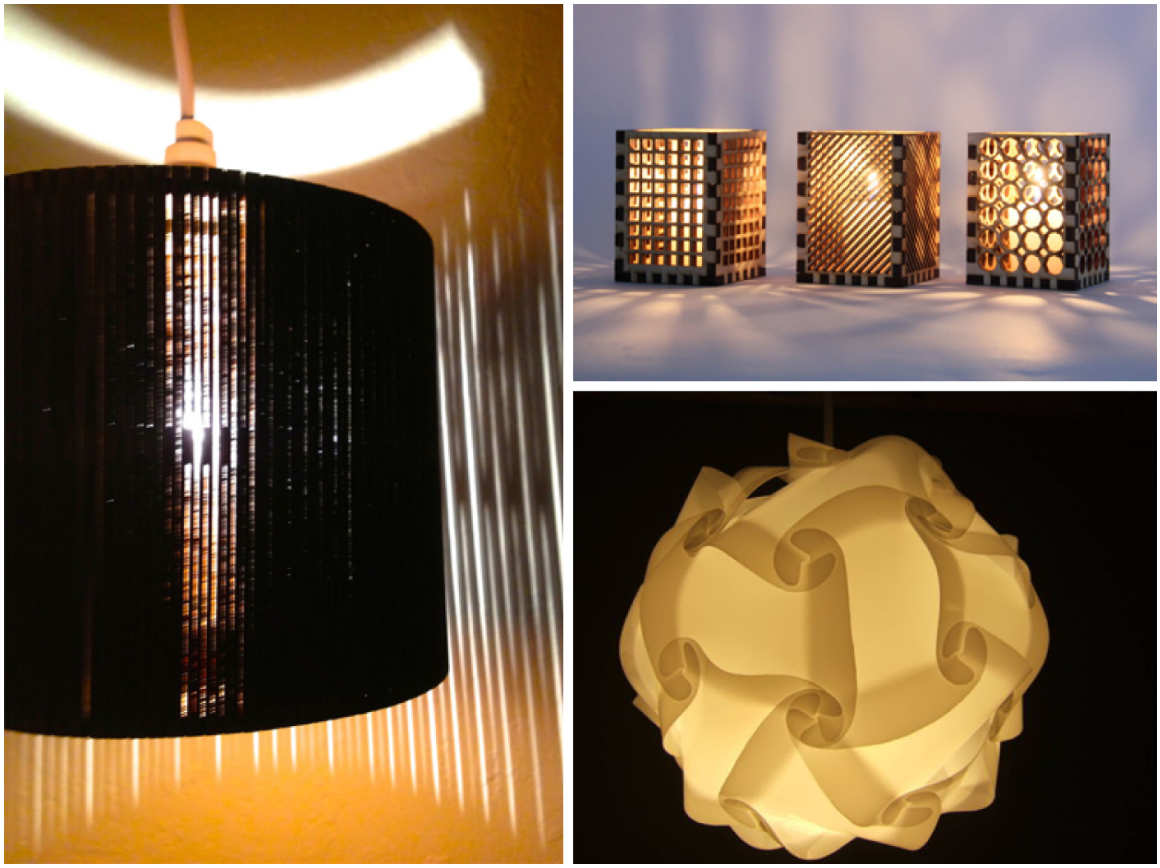


Figure 5-1: a selection of laser cut lamps from Instructables

5.1 Motivation

There is a precedent for creating DIY lamps through digital fabrication. The Instructables community tutorial website has an entire section devoted to DIY lamps, and many examples of patterns that use a laser cutter for fabrication. Many of these Instructables contain minimal design flexibility; they provide instructions that allow people to reproduce the lamp in the tutorial, but do not describe methods of deviating from the original design. The tutorials that do encourage design flexibility often require the person creating the lamp to use professional CAD tools to create their own design. In one popular laser-cut-lamp Instructable, the author recommends using Solidworks to design the form and Illustrator to create the pattern on the shade [**?**]. SolidWorks is more difficult to use than Illustrator, but has the benefit of being parametric. Conversely, Illustrator contains support generating aesthetic forms and patterns and is well suited to creating individual illustrations for

laser cutting, but lacks the parametric functionality needed for the design of artifacts with multiple parts. Both tools are extremely challenging for first-time users.

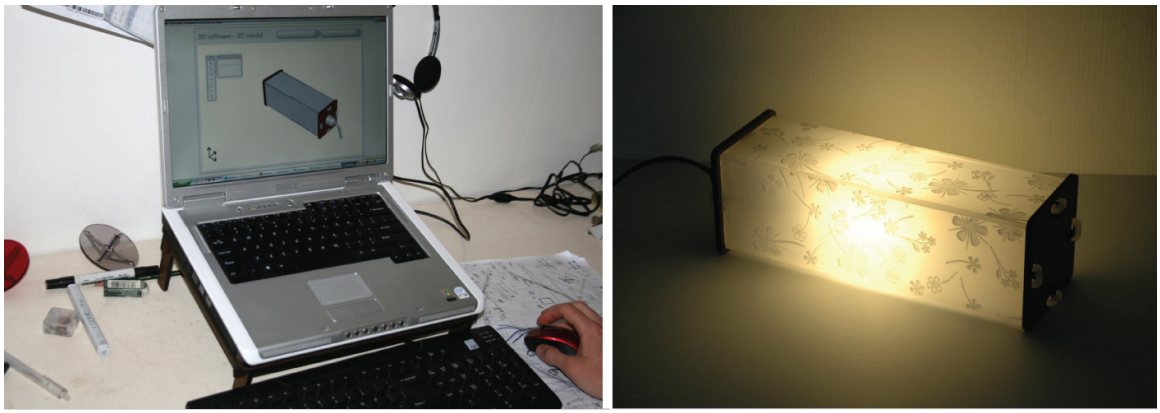


Figure 5-2: Instructables lamp tutorial with SolidWorks design process

Using existing software tools for lamp design also presents other challenges. One common way to make laser-cut 3D forms is by assembling 2D press fit pieces. I found that when creating 3D forms that were curved, it was extremely challenging in 2D-CAD software to correctly size and design parts which would fit the faces of the lamp frame, creating the lamp shade. Although it is possible to create lamps from bare laser cut frames, the shades themselves provided an excellent space for incorporating aesthetic illustrative elements. Without parametric functionality however, it is difficult to modify the aesthetics and form of the lamp in a back-and-forth manner. Computational design offered a solution to many of these problems, while simultaneously providing a form of constructive engagement in programming, fabrication and craft.

5.2 Tool Description and workflow

The first version of Codeable Objects attempted to combine parametric manipulation, aesthetic pattern generation and the conversion of a 3D form to 2D parts into a single computational design tool for lamp design. The lamp itself was comprised of five basic parts, a wooden press fit frame, a set of vellum pieces that fit over the frame to act as a shade, a set of cardstock pieces with a pattern that fit over the shades, and a pre-made standard light fixture that fit into the frame (see figure: 5-3.)

Codeable Objects was developed as a programming library for Processing and contained a set of programming methods that allow the user to describe the lamp, and define the tool paths for all three materials. In the first version of the library, all design took place via textual programming

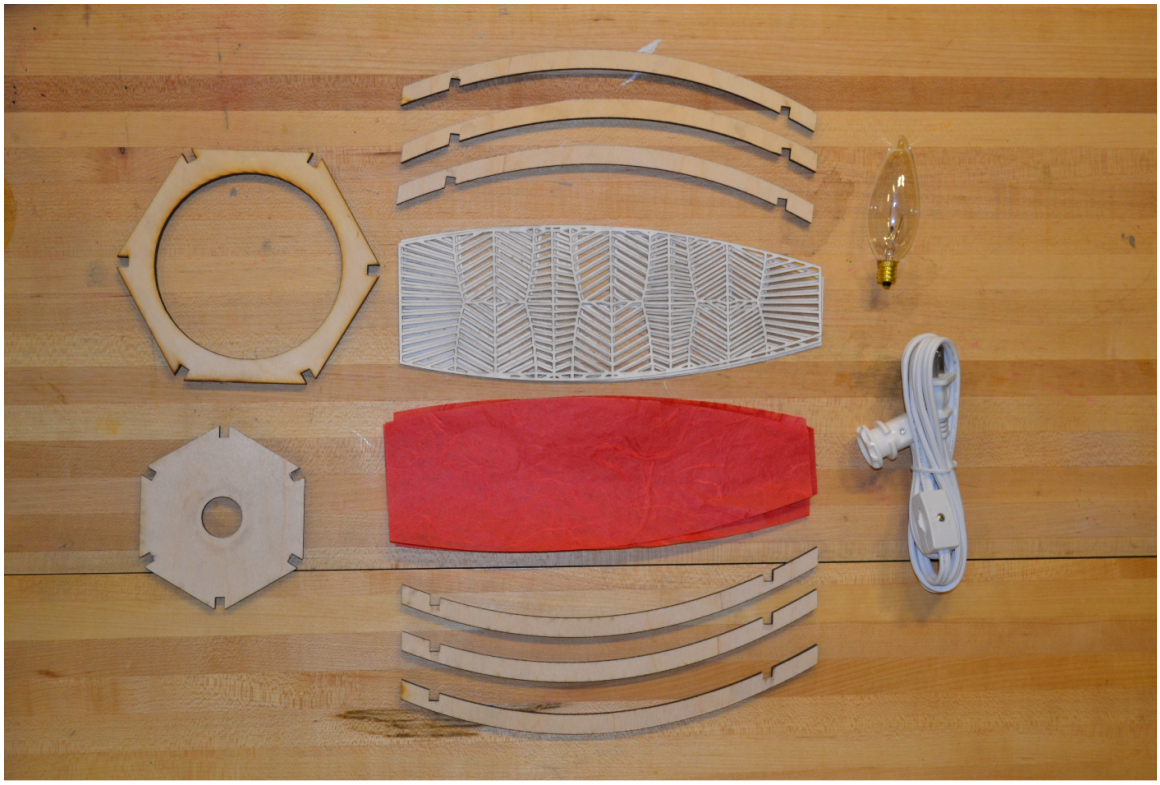


Figure 5-3: the individual parts of a lamp

and keyboard commands. There are four main functions in the Codeable that determine the height, top width, middle width and bottom width of the lamp. The library also provided access to an additional set of methods that control the number of sides, the resolution of the curve and the position of the internal structural supports. To facilitate the construction process, notches are automatically generated in all of the parts to allow the lamp to be press-fit together. The shades are also automatically generated to fit the form specified by the user. The inclusion of this feature gives the user freedom to customize the shape of their lamp, without having to worry about the mechanics of construction.

Codeable Objects also includes a second set of programming methods that allow users to describe the decorative components of the lamp by specifying coordinates in polar or cartesian space. Upon compilation, the coordinates are used by the application to calculate a design using a Voronoi diagram, which is automatically clipped to the dimensions of the lamp shade. As the form of the lamp is altered, the shades and patterns are adjusted to fit. Once the code is compiled, a graphic preview is displayed. For the pilot version, users could use key-commands to toggle between a view of the form of the 3D form of lamp, the Voronoi-diagram pattern, and a 2D preview of the press fit parts (fig:5-4.) A final key-command allowed for the resultant design files to be exported as three separate PDFs, containing the paths for the press-fit frame, the shades, and the pattern files.

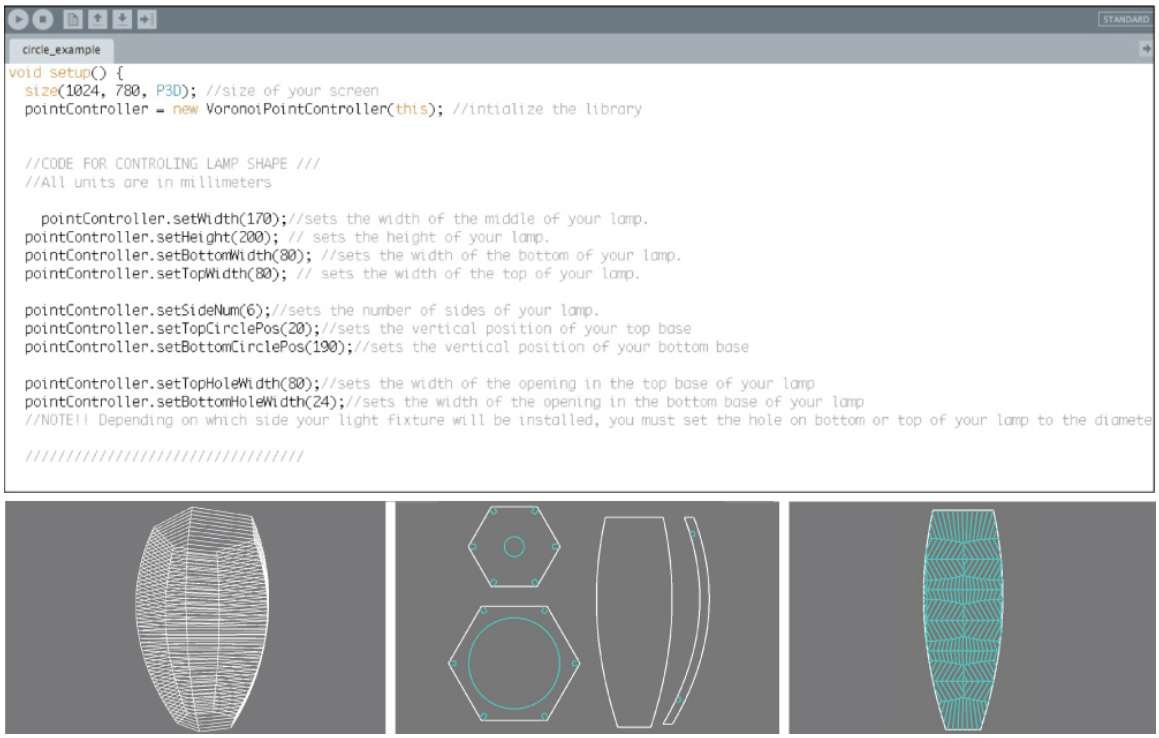


Figure 5-4: The first version of Codeable Objects, with only text-based interaction

The functionality of Codeable Objects was designed to provide platform that allowed for flexible computational design and the creation of complex forms and patterns, while greatly simplifying the process of translating the design to a suitable format for digital fabrication. The laser cutting process produced parts that expedited the construction process, but still but required care and dedication in assembly. Combined, these properties were designed to foster an experience that merged programming, digital fabrication and hand-craft and (hopefully) resulted in a useful finished artifact.

5.3 Evaluation

The first evaluation of Codeable Objects was conducted with a group of nine graduate students, ranging in age from 24-34, during a six-hour workshop. Five participants were women. According to self-reported pre-survey data, all but one of the participants were intermediate to experienced programmers. Five of the nine had previous experience with Processing. The participants indicated they had little or no prior experience in design. What experience they had was primarily gained in high school art classes and college elective courses. During the workshop, each participant engaged in the design and fabrication of their own lamp. Participants received instruction in the use of Codeable Objects and a basic explanation of the principles behind the geometry of the lamp. The pilot version of Codeable Objects was packaged with a set of example programs that contained the

basic code for initializing the library and defining the parameters of the lamp, along with a variety of point generation methods. The examples included algorithms to generate spirals, circles and grids in points. Participants were also provided with access to construction materials, and received training in the use of the laser cutter. They given approximately four hours to design the structure and ornamentation of their lamp, followed by two hours for fabrication and assembly.

Figure 5-5: Several of the finished lamps from the first workshop

5.4 Results

All but one of the participants in the lamp workshop successfully completed their lamp. The one exception was a user who wished to incorporate a specialized light fixture into their piece, but unfortunately damaged her parts while waiting for the fixture to arrive. The participants with little or no prior programming experience primarily relied upon tweaking or remixing the example programs to design the form and pattern of their lamp. Those with more programming experience experimented the library to produce a wide range of forms and patterns. One participant wrote a program that decomposed a black and white image into a point cloud and used that as the basis for her pattern. Another participant wrote a program that used a Gaussian distribution of points to achieve the gradual variation he desired in his final pattern.

The physical assembly process required additional time beyond the duration of the workshop for most participants. This was partially the result of the bottleneck on the laser cutter, however the craft and construction components of the project took longer than expected. Despite this, all the participants returned after the workshop to complete their projects. Some participants chose to add additional steps to the construction process, such as sanding their parts. The physical objects produced were both attractive and functional; participants displayed their lamps in their offices and

homes after completion. One participant returned several days later to build a second lamp so that he would have a matching set for his bedside tables (figure:5-5.)

5.5 Discussion

Because of their prior expertise, the experiences of the majority of the participants in the first study are not indicative of the feasibility of Codeable Objects for novice programmers. Their experiences instead stand in contrast to the experience of the novice coders in the successive workshops and provide important information about the usability and workflow of the software. Some of the experienced programmers provided immediate practical assistance by developing new example programs for Codeable Objects, including an extremely popular program for cosine and sine wave pattern creation. Because the library was open-source, they were also able to submit upgrades to the interface design and functionality during the workshop itself, which were later incorporated into the core version.

The experienced programmers in the lamp workshop exhibited limited knowledge of computational design prior to the start of the workshop. When asked in the pre-workshop surveys how they thought programming, design and craft could be combined, people were either uncertain, or described the combination as method to create dynamic interactivity, rather than a tool for the design of form and pattern:

> *"You can combine software and hardware and make craft more dynamic (e.g. sensors)."*
>
> Lamp Participant pre 1

> *"[programming] gives [you] the ability to make something dynamic."*
>
> Lamp Participant pre 3

Following the workshop, participants demonstrated an awareness of some of the specific affordances of computational design, and the benefits of combining it with digital fabrication:

> *"I understand now how programming can be used for quick prototyping and mockups that can be used to inform final design decisions. This is easy [and] helpful when using physical materials where mistakes can be costly."*
>
> Lamp participant 2

> *"Using programming in the design process adds some exciting and unique capabilities over traditional design and crafting, including mixing in different algorithm and ideas from other existing software, and rapid prototyping of complex designs."*
>
> Lamp participant 6

Participants were also pleased with the creative affordances of the tool, and described how the software enabled them to expand their programming abilities to the realm of art and craft with greater success:

> *"I think programming makes designing more accessible because you don't have to be able to draw or paint."*
> Lamp participant 4

> *"I love the idea of being able to combine my interest in programming for creative expressions."*
> Lamp participant 6

One participant remarked that she had always believed that she was terrible at art, but that making the lamp had altered that perception. Although the primary objective behind Codeable Objects was to make a form of algorithmic craft that was accessible to non-programmers, in the first workshop, it provided a pathway for programmers to apply their skills to design and craft. From these responses, it is apparent that even among experienced programmers, algorithmic craft has the potential to expand people's understanding of the applications of programming and motivate them to apply computation to other forms of production and expression.

Several participants also put forth detailed critiques of the programming process, which brought up larger questions on the practice of computational design. One participant reacted against defining the generative qualities of the Voronoi diagram patterns as a design method:

> *"Changing the parameters didn't always generate the pattern you have in mind. It was more like generating a few semi-random patterns and you choose one that looks good. It is rather a trying-and-choosing rather than designing /making something you planned to have. I think "design" involves "intention" and "planning." Programming, crafting, and design should be combined in the way that entails prior planning and intentions as opposed to cutting together the semi-random choices, which could be good but I wouldn't call that design."*
> Lamp participant 6

As this quote indicates, the attributes of randomness and generativity do not automatically lead to optimal or good design decisions. Some deciding factor has to play a role in the process, but the criteria for this decision are often ambiguous. This criticism touches on a core debate about the role of conscious design and the restriction of intuitive creativity in computational practices. The emergence of comments like this are encouraging, because they reflect the engagement of the participants, not just with the task at hand, but in a critical evaluation of the creative implications of this form of creation.

5.5.1 Challenges

There were also elements of the process that were problematic for the participants. It became clear during the workshop that textual programming was not the best method to design the form of the lamp. Many of the participants became frustrated about having to set the parameters and then wait for the compilation process to complete before they could view the resulting form. This issue was addressed partly in subsequent versions of the tool by replacing the textual parameters with a set of sliders in the compiled application, which would adjust the form in real time, across each of the views (figure: 5-6.)

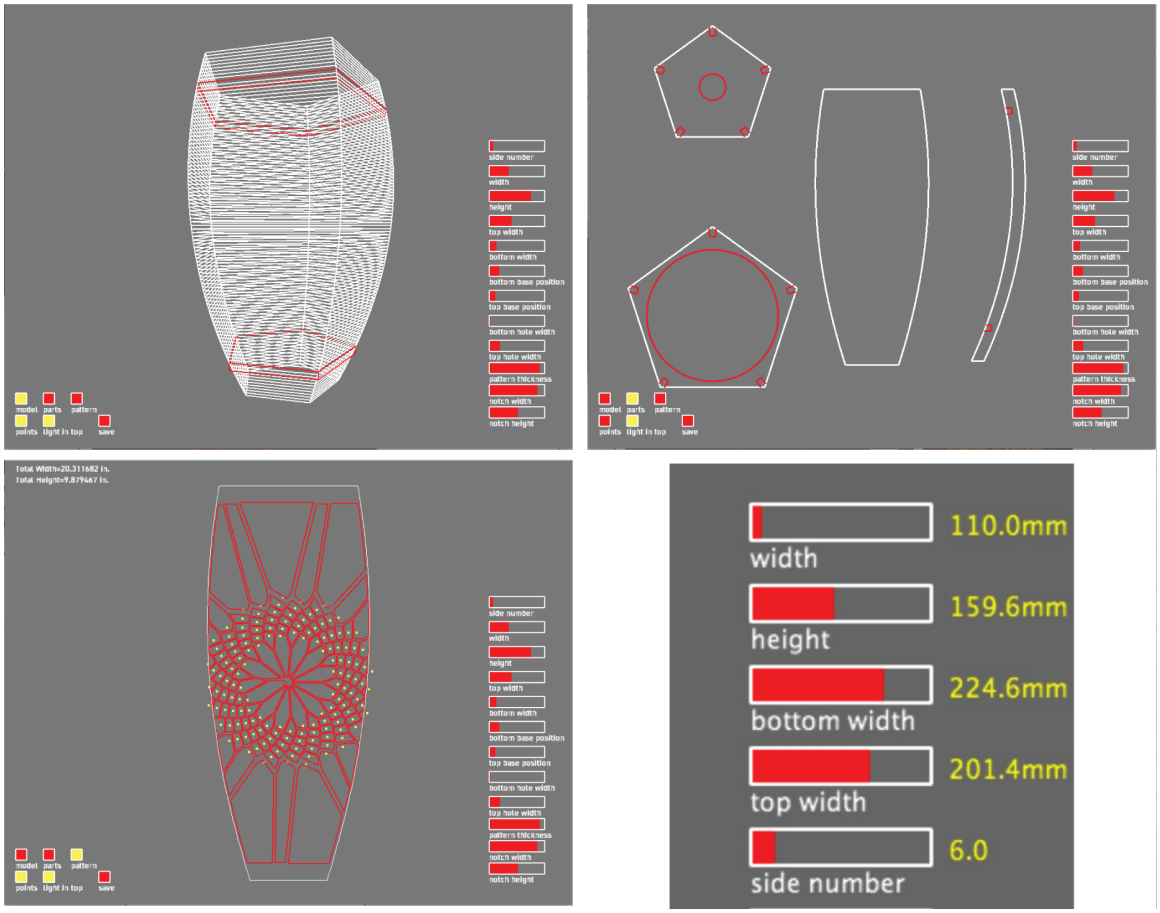


Figure 5-6: Revised graphic view with sliders

The textual programming method was useful for creating pattern on the shades. The simple method of specifying points in a programming context provided a space for creativity and resulted in greater variation in the lamps. If the tool had relied on a more standard set of graphical user interface(UI) components, like sliders to control the point generation, it is doubtful that the same range could have been achieved. On the other hand, it was clear that the less experienced program-

mers had more difficulty deliberately designing the patterns of their lamps, and relied primarily on adjusting and remixing existing examples.

Another restriction of Codeable Objects was apparent in the design process. Although adjusting the parameters and input values to a system can be considered a form of design, it only touches the surface of the design opportunities made possible algorithmic craft. With Codeable Objects, people cannot modify the program which defines the range of forms and patterns that are possible, unless they alter the source code of the library. This limitation prevented the evaluation of some of the more interesting computational design processes such as allowing people to create personal stylistic abstractions for patterns and forms, by writing their own algorithms. The stylistic limitations of Codeable Objects contributed to the high success rate in project completion, and the general attractiveness of the resulting projects, but limited the design flexibility.

One other defining component of the Codeable Objects pilot workshop was the contrast between the difficulties that arose through computational design and digital fabrication and the challenges in crafting. The difficulties people experienced while designing and fabricating their projects were often discrete; such as correcting for mathematical error in coordinate placement, or having the incorrect setting on the laser cutter. More complex problems arose in these contexts as well, such as confusion around principles geometric point placement, however they were problems that could be addressed through verbal instruction and explanation. The challenges encountered in the crafting session were of a material or physical quality. People struggled with finding the best techniques for assembling the parts and finishing individual pieces so that the resulting product maintained an attractive appearance. Most participants were surprised at the amount of time required to complete the physical assembly, and were sometimes frustrated when variations in the crafting process violated the precision and perfection of the digital design, and laser cut parts.

Some of the frustrations in the physical construction process were addressed in subsequent workshops by creating a paper variation of the lamp that was faster and easier to assemble and required no gluing (figure: 5-7.) In addition, a feature was added to the software which reported the approximate material size required for a design, so that users could ensure theirs would fit on the bed of the laser cutter.

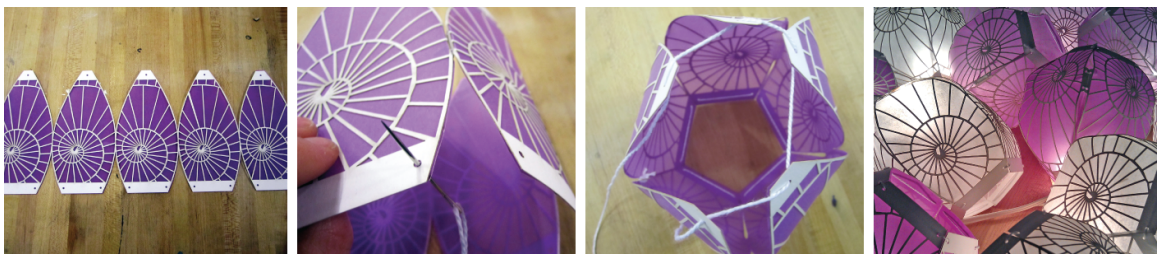


Figure 5-7: The revised paper lamps

Although there is often an opportunity to improve a user's experience through changes in the interface and artifact design, it is useful to distinguish between frustrations that are the result of a software interface, and learning processes that are inherent in design and craft. In both craft and computation, care and practice contribute to more successful results. Furthermore in craft, material variation and inconsistency can be difficult to manage, but it can also contribute to the uniqueness and value of an object. The process of supporting people in algorithmic craft, is not just about building tools that make the programing process easier and the crafting component less prone to risk. Instead, computational tools should be designed to remove technical barriers in programing and fabrication so that people can address the more interesting (and often more difficult) design challenges of envisioning and refining computational systems. Craft processes should be feasible to achieve, but also recognize that part of the pleasure and accomplishment of crafting comes from the risk and difficulty it entails.

5.6 Summary

Codeable Objects was successful in that it allowed people to produce useful and beautiful objects with personal value, through a relatively pleasurable experience in computational design, digital fabrication and craft. It also produced in experienced programmers, a recognition of the aesthetic and design potential of computation, and brought up deeper questions about the role of conscious choice in computational design. The workshop demonstrated the importance of immediate visual feedback for informing design decisions, and also indicated the importance of applying textual programing to formats that clearly demonstrated its advantages. The workshop also highlighted the utility of open-sourcing algorithmic crafting tools, as this lead to immediate innovations in the functionality of the software when testing with advanced users. For subsequent tools, my goal was to build on these success while attempting to diminish some of the design limitations of Codeable Objects. I also began working with a representative group of designers: young people with no prior programming experience.

Chapter 6

Soft Objects

After an evaluation of the successes and limitations of the Codeable Objects library I made an effort to modify the library in a way that would allow for a broader range of computational design approaches and end products. In particular, I was interested in exploring the domain of algorithmically-crafted garments and fashion accessories. Fashion is an exciting domain to connect to computation, because it appeals to groups of people who are often under-represented in computer science, particularly women and girls. Computational fashion design has the potential to resonates well with algorithmic craft, because it offers the opportunity to apply digital fabrication to textiles

and fabrics, and introduces sewing and pattern-making as components in the construction process. Because garments and fashion accessories are created to be worn, computational fashion design requires the creator to consider questions of comfort, sizing and personal taste and and style when writing code.To support computational fashion design in the context of algorithm craft, I expanded Codeable Objects into a more general programming library entitled Soft Objects and evaluated it over a 10-day workshop with young designers.

6.1 Motivation

With the growth in public awareness of digital fabrication, there is a great deal of enthusiasm for fashion applications of digital fabrication technology. Much of this excitement is directed towards 3D-printed wearables and textiles. In July 2010, Iris Van Herpen released her Crystallization collection, which featured her first computationally designed, 3D printed piece, marking the first time a 3D printed garment had appeared on the runway [2]. Van Herpen and many other fashion designers have continued using 3D printing as a medium for fashion since then. As a result, computationally designed, digitally fabricated fashion is often synonymous with 3D printing. The 3D printed garments and accessories produced by professional designers like Van Herpen serve as inspiration for the future of digital fabrication, and demonstrate wearable forms that would be impossible to create through any other means. For the average person however, computationally designed and digitally fabricated garments of this nature present considerable limitations. Given current technology, cost and material restrictions, the majority of 3D printed garments are impractical for every-day wear and require advanced fabrication techniques that are unavailable to many non-professional designers. Garments like the N.12 bikini, designed by Continuum [**?**] are intended to be more practical, and available to consumers, but for the time being they are limited in scale, and still come at a steep price point. The construction of 3D printed garments of this form appear to have little in common with existing methods of garment production, like sewing, knitting and embroidery. Though garments of this nature are inspirational and groundbreaking, individuals with sewing, knitting or other textile manipulation skills may perceive their interests to be incompatible with computational design and digital fabrication in this form.

Although perhaps less publicized than 3D printed fashion, other other designers are merging fashion with computation and digital fabrication in a way that blends new technology with established approaches. Diana Eng's Laser Lace tee collection contains laser-cut machine-washable t-shirts with floral-inspired iconography, and her Fibonacci scarf is created through traditional knitting techniques, meshed with a Fibonacci knit pattern. Eunsuk Hur's modular fashion pieces are inspired

Figure 6-1: 3D printed fashion (from left to right: Crystallization 3D Top by Iris Van Herpen, Drape Dress by Janne Kyttanen, N12 Bikini by Continuum Fashion, Strvct shoe by Continuum Fashion

by tessellations and fractal geometry. By creating garments through laser-cut interlocking pieces, Hur's aim was to produce items that were robust and durable, also gave the user the opportunity to use their inner creativity to come up with new and interesting items, by rearranging the individual components (figure:**??**.)

Examples such as these demonstrate a space in computational fashion design that is compatible with non-digital interests, and skill-sets. Subtractive fabrication machines, like laser cutters and vinyl cutters are dominant in this type of work, because they work with traditional materials and can produce garments at a much lower cost and larger scale than 3D printers. Variation in materials also can translate to a wider set of possibilities for hand crafting as well as different aesthetics and styles. It should be noted that accessible forms of 3D printing can produce compelling wearable objects, however they are generally on the scale of jewelry and small accessories. Garments designed and produced as a blend of digital fabrication and textile materials and construction processes correspond well with the values and practices of algorithmic craft. The Soft Objects programming library was designed to allow new programmers to design the forms and aesthetic components of fashion accessories and garments through programming, and then physically construct the garments using subtractive forms of digital fabrication and sewing.

6.2 Tool description

The Soft Objects library contains a set of methods that allows users to draw shapes and patterns and then export those shapes and patterns in a vector-file format that is compatible with x-y axis digital-fabrication machines. Similar to Codeable Objects, Soft Objects is used within the Processing programming environment and contains a set of programing methods that enable the design of visual forms and patterns. SoftObjects allows users to define and manipulate basic geometric primitives such as Points, Lines, Curves and Polygons. These primitives can then be collected

Figure 6-2: "Ready to wear" computational fashion (from left to right: Fibonacci Scarf by Diana Eng, Biomimicry laser-cut bracelet by Stefanie Nieuwenhuyse, Laser Lace All-Over Tee by Diana Eng, Interstice bracelet by Nervous Systems, Modular Fashion by Eunsuk Hur

within Pattern and Shape objects—structures designed to capture surface decoration and 2D structure, respectively—to form increasingly complex designs. Soft Objects is formulated on an Object Oriented Programming (OOP) paradigm, which lets users create and manipulate collections of geometric primitives—Patterns and Shapes. This structure differs from Processing's drawing API, which uses a functional programming approach. Users are presented with a 2D preview of their designs when they compile their code. The structure of Soft Objects enables users to simultaneously apply transformations to all of the elements in a collection that make up a complex pattern or shape. The objective behind this structure was to open the design possibilities in a format that was suitable for creating complex 2D designs and aesthetic patterns for clothing and accessories, while ensuring that a user's designs would be compatible with subtractive fabrication. To simplify the process of garment creation, Soft Objects included functionality to import existing cut patterns as scalar vector graphics files (SVGs). This allowed users to merge programmatically generated designs with pre-sized shirt, dress and pants pattern-templates.

Soft Objects supports a variety of digital-fabrication machines by allowing users to save designs to vector portable document format (PDF) files. PDFs can be used by different production tools, including ink-jet printers, vinyl cutters, laser cutters, and computationally controlled embroidery machines. Output from Soft Objects can be fabricated on essentially any x-y axis tool. 3D structures can be created by assembling fabricated pieces. Figure 6-4 demonstrates the workflow from code to a finished object. The Soft Objects library also contains a collection of pre-defined algorithmic patterns that can be initialized, including Voronoi diagrams, Koch curves, and L-Systems, and an extensive set of example programs that users can modify and combine to produce individual results. These examples and pre-defined algorithms were created as a way of quickly exposing new programmers to some of the complexity and variability that is possible through computational design, and demonstrate the abstract qualities of computation, without immediately requiring them

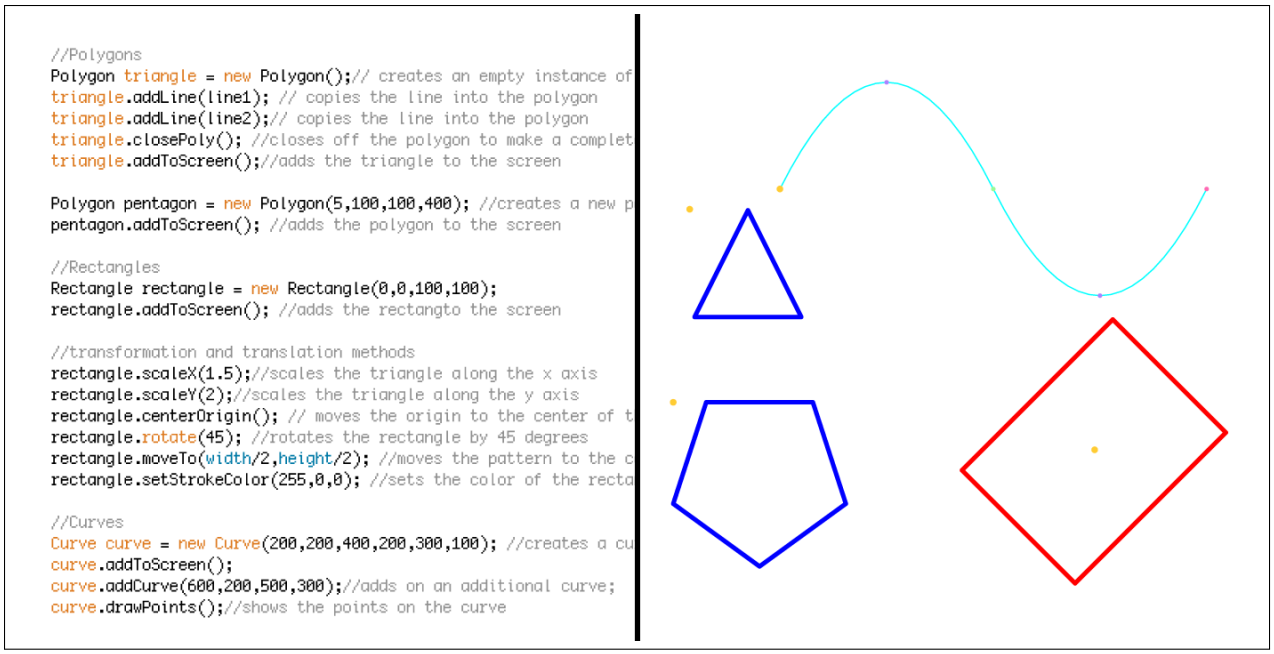


Figure 6-3: Soft Objects primitives

to learn substantial amounts of programing structure and syntax.

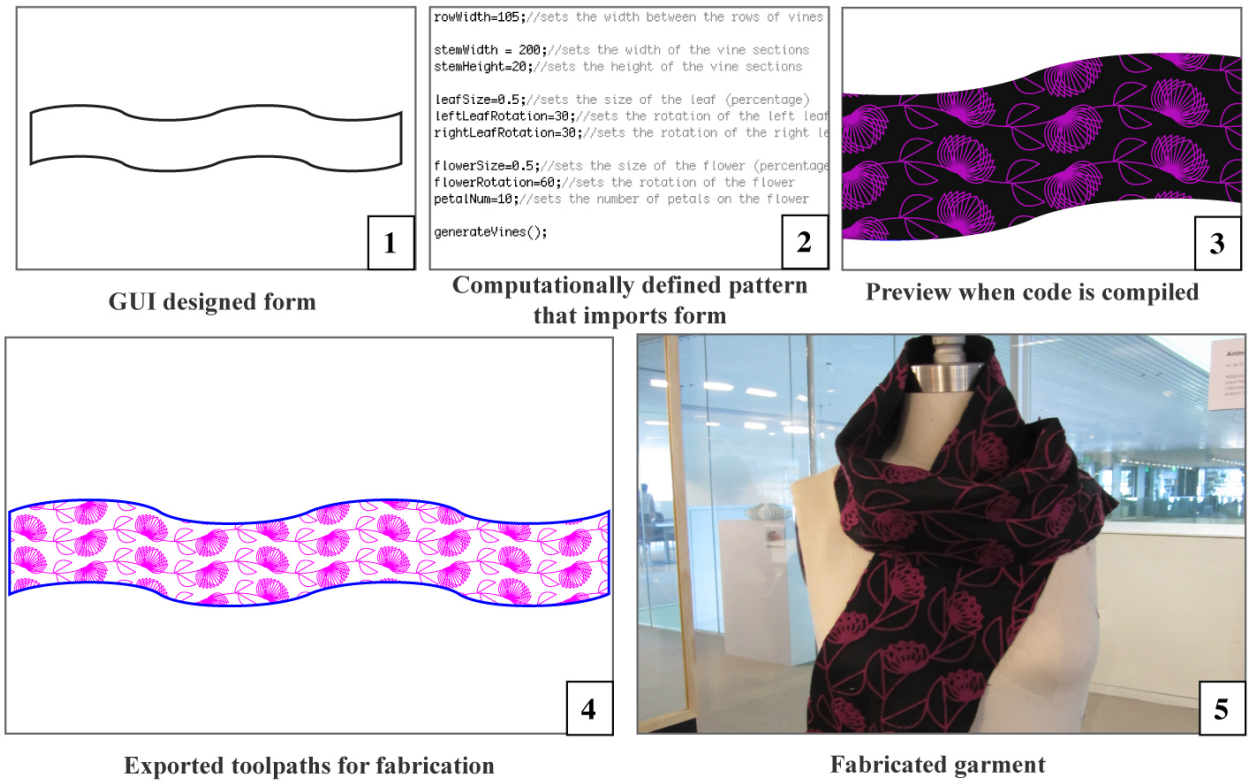


Figure 6-4: Soft Objects workflow

6.3 Workshop

The evaluation of Soft Objects was conducted during a 10-day workshop with a representative group of participants—eight young adults, aged 11-17, 75% male and 25% female. A significant majority (88%) stated in pre-surveys that they had little or no prior experience in programming, and only one participant had prior experience in Processing. All of the participants indicated some level of prior experience in art, design, or craft. Most attributed their design or craft experience to art or drawing classes. The workshop was conducted at the Nuvu Magnet Innovation Center for Young Minds. Participants were given 10 days to conceptualize and construct a garment using a combination of computational design, digital fabrication, and sewing and crafting. The workshop was more open-ended than the Codeable Objects workshop; participants could produce any type of garment they wished as long as components of it were computationally designed and digitally fabricated. During the workshop, participants were introduced to Soft Objects and the concept of computational fashion through a multi-step process that engaged participants in different levels of programming through the construction of different garments and accessories. First, participants were provided with a small set of example programs similar to the lamp workshop. This step allowed them to manipulate a core set of parameters to generate the pattern and form of a scarf, which they then cut on the laser cutter (figure: 6-5.)

Figure 6-5: Bracelets and scarves from preliminary activities

Second, participants were instructed in a number of primary programming concepts, including iteration, function definition, and the use of variables and primitive data-types. During this instruction, participants were guided through the process of independently using Soft Objects and generating their own programs from scratch. They used these programs to create a design for a wooden bracelet (figure: 6-5), which was then laser cut and assembled. These two initiation activities were intended to demonstrate the stylistic affordances and design properties of computational design and digital fabrication, as well as providing participants with the practical foundation to

begin conceiving of and executing their own ideas. During the remainder of the workshop, the participants were asked to conceive their of own garment concepts and designs and provided with the resources to design, prototype, and craft finished artifacts.

6.4 Results

Participants in the fashion workshop were successful in using programming and digital fabrication to design and produce finished garments. During the initiation activities, participants independently wrote and compiled programs of their own and produced physical products based on the design generated from that program. Furthermore, with assistance from the instructors, the participants were able to apply more sophisticated programming methods to produce a diverse set of final products (figure: 6-6). One pair of students developed an "armor dress by writing a program that geometrically described a single "scale shape, imported a dress pattern from Illustrator and filled it with rows of scales that corresponded with the dimensions of the dress. Another pair created a geometrically inspired dress with a patterning of different-sized octagons and squares that were laser cut from starched fabric. Another student created American-flag-inspired pants using a program that generated random orderings of red and blue stripes on a white background. One group that was less interested in the process of sewing clothing created a program that generated a recursive virus-like pattern and then screen-printed the pattern on pre-made sweatshirts and t-shirts.

Figure 6-6: Completed garments (from left to right: octagon dress, flag pants, samurai dress, viral sweatshirt)

On the post survey, when asked if they "were able to complete a finished project to their satisfaction, 100% of the fashion participants responded in the affirmative. The resultant garments were attractive and functional, indicated by the fact that participants from the fashion workshop kept and wore their creations. Direct comparison of the pre-and post-workshop surveys also demonstrated

that on average, participants in the workshop indicated their interest in crafting increased after the workshop, as did their enjoyment of the design process. 88%of the participants in the fashion workshop indicated that they felt more comfortable programming after the workshop than before.

6.5 Discussion

The Soft Objects workshop occurred over a longer time period than the Codeable Objects workshop, and explored a wider range of crafting and computational techniques. As a result, I had an opportunity to spend greater amounts of time observing and talking with the participants. Because the participants were novice programers, their experiences better reflect the target demographic of this research. Through this workshop we confirmed that algorithmic craft activities can actively support the expression of personal identity among young designers. It also fostered feelings of confidence in programming and supported aesthetic and technological literacy. The workshop promoted a deep understanding of computation as evidenced by critiques of the participants as well as demonstrating the importance of physical prototypes in the design process. Finally, the workshop promoted a sustained engagement in programming.

6.5.1 Identification as a programmer

Similar to the participants in the Codeable Objects workshop, the fashion participants began with vague ideas about the applications of computation. When asked in the pre-workshop surveys how they thought programming, art and craft could be combined, participants responded in writing in the following ways:

> *"To be honest, I am not sure too sure how it would all be combined because I don't know much about programming."*
>
> Fashion Participant Y
>
> *"With programming, we can make programs do things for us."*
>
> Fashion Participant J

In addition, the participants had almost no prior knowledge of computational design. Those participants who had some form of prior programming experience associated it largely with interactivity and actuation:

> *"I would love to combine things like T-shirts and speakers or other different types of technologies."*
>
> Fashion Participant J

Following the completion of their final projects, participants were much more descriptive about the applications of programming:

"I think [programming and fashion] are really interesting but I never thought they could ever be together in one concept, and it's awesome that I know that now- that you can design aesthetically pleasing things from coding."

Fashion participant K

"I've never thought of programming as physical, I thought it was only in computers. But then when we made the scarves and stuff, I thought that was really fun."

Fashion participant M

Many people also expressed a growing confidence in their ability as programmers. During the interviews, several separately stated that that while they did not feel completely comfortable programming independently, the experience made programming feel significantly more accessible:

"For someone who never had any programming explained to them before, when you look at [computational drawing examples] it feels really inaccessible, but now that I've been taught a little bit of [programming], I can kind of crack at the walls a little bit and understand how it works, and that makes it more accessible to me."

Fashion participant S

"Even though now I'm not really a Processing expert now, I've just experienced it and it's not as scary to me, like the idea of coding, you just kind of have to learn some stuff and practice it more, but I think I definitely understand the concept of it."

Fashion participant K

Finally, the programmers in the fashion workshop expressed feelings of pride and a sense of accomplishment in their new-found programming skills:

"I'm actually kind of proud. I know what's going on. It feels different... I just thought [programming] was just about these huge programs that you have to piece together, and that you have to be really, really smart to do it, but I can do it."

Fashion participant P

"I think it was really cool that we used [programming] for fashion, cause I think a lot of people might think people who do fashion aren't really smart or something, and then they think that people who design code are like brilliant coders and can do really awesome stuff with it."

Fashion participant K

The confidence, sense of belonging, and personal agency demonstrated in these comments stand in contrast to popular views of programming as specialized and inaccessible. These sentiments indicate that introducing programming in the context of algorithmic crafting not only has the potential to change people's understanding of the relevance and applications of computation, but also promote a personal awareness of technological literacy and competence.

6.5.2 Application of computational affordances

Aside from a general understanding of the potential applications of computation, participants were successfully able to leverage the advantages of computational design in their final projects. For example, the octagon dress used parametric design principles as the participants were able to change the size and orientation of the octagons in the dress by modifying several parameters at the start of their program to affect the entire pattern, rather than rotating and adjusting each shape independently. The samurai dress took advantage of the computational properties of precision and automation. With help from an instructor, the creators programmatically generated an individual vector file for each row of scales of the dress, expediting the production process (figure:6-7.) The samurai dress is particularly interesting because it shows a remarkably successful transition from the design expressed in original concept art by the participant, and the resultant garment, demonstrating the adherence to the person's creative vision through computation. Lastly, the viral shirt, demonstrated an application of generativity. The aesthetic of the viral pattern was produced through the use of a weighted random-number generator to determine the number and length of the branches in each recursion of the pattern. Although the implementation of the weighted number generator was facilitated through the help of the instructor, the participants came up with the idea of using it on their own.

Figure 6-7: Progression of samurai dress (from left to right: concept sketch, computationally generated pattern,laser-cut components of final garment)

Many of the participants demonstrated an understanding of the rationale behind the methods they were using. One of the creators of the armor dress project compared the process of programming the design of the dress to that of manually drawing it:

"With drawing you can achieve everything programming can, but I would prefer to program it. [programming] can be pretty convenient...the computer is helping me. Like if you want to make pizza, the computer is like a pre-made crust."[Fashion participant E]

Participants also articulated an understanding of specific programming functionality. One participant described the point at which she understood the application of parameterization:

"One moment that stuck out was when you helped me make a code with original geometry that could be changed so that when you changed one thing it changed everything and that was cool because I felt like I actually made something that could be changed and then applied." [Fashion participant K]

The ability to understand and describe how computational support one's creative objectives is essential in motivating an individual to spend time learning and implementing these methods. Once the aesthetic possibilities of the recursive viral pattern were apparent to the designers, they became engaged in better understanding the underlying algorithm, so that they could produce a pattern to their exact specifications. Algorithmic craft can provide the motivation for participants to tackle complex computational problems with sophisticated approaches, when problems are clearly grounded within the design objectives of the individual. A continual challenge in engaging new programmers in algorithmic craft is in selecting programming approaches that allow for compelling aesthetic and design possibilities, but remain approachable for first-time coders.

6.5.3 Aesthetics and Identity Expressed through Code

The fashion workshop provided the opportunity for participants to use computational aesthetics as a way to express their personal style. Fashion can serve as a means of self-expression and for conveying one's identity. Discussions on fashion conducted at the start of the workshop indicated that participants were aware of the connection between fashion and identity and were eager for opportunities to create clothes that expressed their style. As a result, the majority of the garments created in the workshop contained an expression of the "fashion sense of the participants who created them. For example, the participant who created the flag pants was very explicit that the pants have some form of an American flag motif, but not resemble the traditional, and as he put it "tacky flag pants that he commonly saw. He wanted his flag pants to be "something that he would actually want to wear. His programming choices were made in direct consideration of his desire to create a pair of pants that he felt were fashionable.

When asked about the experience of making and designing his pants he said:

"[The workshop] definitely changed my impression of making clothes, I thought it was pretty quick to make clothes, but it actually takes a long time, and it's also really fun. I love the fabric I made." [Fashion participant M]

His enthusiasm also was evident in the fact that after the pants were complete, he tried them

on and wore them for the remainder of the workshop. This level of enthusiasm was common among participants; they all proudly modeled their creations, and many of them wore them home. This behavior suggests a relationship between the decisions made in a programming context, and the participant's desires to express their visual identity. The participants were selective in the code they wrote to design their garments because they intended to wear the garments, and as a result, be represented by them. This powerful affective relationship between computation, design and self-expression provides a natural way to engage people in programming and design by supporting their personal interests.

6.5.4 Physical and Digital connections

One of the challenges of Algorithmic Craft, alluded to in the Codeable Objects discussion, is that the practitioner must work between digital designs with physical materials and processes. Physical prototypes often serve as a key point of transition between these spaces. In the fashion workshop, prototyping played an important role, and demonstrated how computational tools can support and sometimes hinder the prototyping process. The focus on fashion made it possible to supply the participants with large amounts of inexpensive test fabric. The laser cutter could cut fabric much more quickly than thick materials, which allowed participants to produce numerous prototypes of their projects before creating a final piece. Most groups produced two or three prototypes, with one participant creating six iterations of a single jacket. This rapid production process formed a direct connection between discoveries made in the physical prototyping space and decisions in the programming realm. In the case of the octagon dress, (figure:6-6), the participants first cut test rows of octagons to determine the appropriate scale, then adjusted their design by modifying their program. When they had cut out a second more complete version of the dress, they rotated one of the shoulder straps on the physical prototype and formed an idea for a one-sided shoulder strap. They implemented this design change in the digital version of the dress by making additional changes to the size and rotation of the shapes defined in the code. When asked about this process in the interview one of the participants said:

"I think it was really fun that we got to do a prototype first because then if you don't like it, you don't feel a lot of pressure because you can make it again really fast, and there's no stress because if it doesn't turn out well, then it's not your final project. [Fashion Participant K]

The combination of programming, rapid fabrication, and physical construction allowed for a design approach that transitioned from programming to fabrication to programming adjustments based on the fabricated elements, and then back to fabrication. This iterative approach resulted in a closely linked cycle of physical and digital engagement.

6.5.5 Enthusiasm in crafting and coding

One of the most encouraging aspects of following the fashion workshop, was the participant's enthusiasm and desire to continue making. Participants talked extensively about what they would like to make in future with programming, consisting citing that they would like to continue making clothing, or other personal functional items like furniture and "things they could use around the house. The experience of both sets of workshop participants also demonstrated the ability of these techniques to produce objects that were designed to complement personal items and living spaces. When asked what she would like to make if she continued to program, one participant responded:

"Things like we're making now, things that you would want to keep or use, things that look nice as opposed to like computer games, or "input-output devices. I think those are fun, but it's not as cool as things that you can hold in your hand. I actually hung up the scarf I made in my room, and now I can be like "I made this on Processing and people will be like what? It's cool! "[Fashion participant K]

This enthusiasm, combined with the high potential for individual expression and sense of accomplishment encouraged us to continue exploring fashion and garment production as topic space for algorithmic craft. While the fashion workshop highlighted many positives in this sense, we also encountered several areas for improvement.

6.6 Limitations

The most evident barriers in the Fashion workshop involved the syntactic challenges of programming. Many participants expressed a frustration with the syntax in both surveys and in-person interviews. Although the workshop participants were able to generate their own programs, they required more assistance from an instructor to write some of the commands. In addition, a feeling of needing to memorize programming syntax frequently translated to a sense of frustration. One participant stated in an interview:

"I couldn't memorize things, so it also was frustrating for me to always have to get you to help me write the code." [Fashion participant K]

Many people requested some form of written "cheat sheet" that listed the key methods and how to use them. They also pointed out that you often had to write a lot of code (including import commands as well as setup and draw functions), even for simple tasks. Writing code for the first time is always challenging, however the high levels of frustration registered by the participants often focused on aspects of programming were extraneous to the design itself. Because I wrote Soft Objects as a Processing library, it required that the syntax correspond to Java, a difficult language for beginners. Java is a general application language and has many syntactic requirements that are unnecessary for computational design applications. Based on this difficulty I concluded that future

algorithmic crafting tools should explore domain-specific languages that directly applied to design and fabrication.

Along with difficulties with the programming syntax, participants struggled with some of the post-processing techniques. In order to be suitable for digital fabrication, many of the participant's computationally-generated designs required some post processing in Adobe illustrator. Usually this required using illustrator's shape boolean functionality to merge shapes or expand outlines so that the vector paths would correspond to the desired cut pattern on the laser cutter (figure:**??**.) Although the methods to perform these operations were simple, they needed to be repeated every time the design was modified programatically. This was not only inefficient, but sometimes prevented people from determining if their designs were feasible for fabrication when they were in the programming environment. This difficulty encouraged me to focus on ways of removing the need for illustrator from the process altogether so that all design modifications could be initialized and updated programatically.

Many participants also struggled with the concept of prototyping. These struggles were evident in practical aspects, such as participants not saving their programs and digital design files to come back to later (despite repeated reminders from the instructors to do so). Participants also frequently spent too much time on assembling their early prototypes and were frustrated when they realized they would have to make design changes and repeat some of the manual labor. Although we took pains in the workshop to introduce participants to the concept of prototyping, these instructions were not always absorbed. This may present an opportunity for future algorithmic crafting tools to contain specific features to encourage and support the physical prototyping process. One possibility is to include simulation tools that preview the constraints and behaviors of physical materials. The value of working with physical prototypes should also be supported however, by features that allow the designer to scale their files and fabricate first in miniature, and software that allows easy access and management of multiple versions of a design throughout the design process.

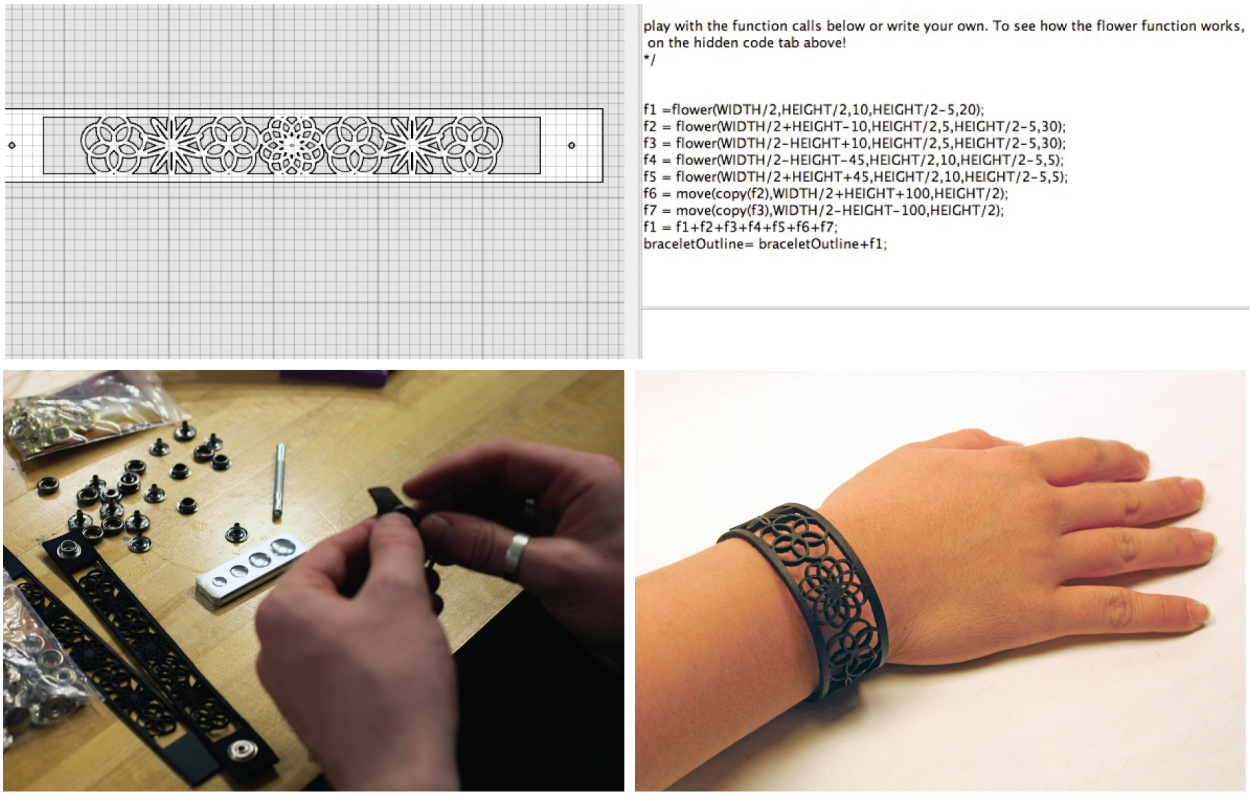
Finally, as in the Codeable Objects workshop, participants were frustrated by the delay between adjusting their code and seeing the results. Although, not an absolute solution to the challenges in learning programming syntax, a programming environment with more immediate feedback could assist with the issue of syntactic challenges by providing novice users with improved ways to visualize the effect their syntactic changes have on their design. The implementation of background compiling, the process by which code is automatically compiled and executed as changes are made, has been applied successfully in several tools for novice programmers, including Scratch and Alice [**?**], and more recently with Khan Academy [**?**]. Our reliance on the Processing for Soft Objects made the incorporation of real-time compilation infeasible, however it quickly became a goal for future tools.

6.7 Summary

Participants in the Soft Objects workshop were able to create personal wearable items that were beautiful, functional, and personally meaningful through computational design, digital fabrication and craft. During the workshop, the participants learned new skills in programing and digital fabrication, as well as techniques in pattern-making and sewing. The participants emerged with an awareness of applications of computational design for digital fabrication, specific understanding of some of the primary elements of programming, and a desire to continue using computation to build their own objects. Although there are many areas for improvement in future tools, the Soft Object workshop primarily demonstrated the importance of incorporating approaches that further reduce the practical challenges of programming. Possible approaches include removing extraneous syntax, reducing the number of programing methods, providing rapid and informative visual feedback, and adding features that do a better job of helping users make their designs feasible for fabrication. The objective of these features is not to trivialize the process of algorithmic craft, but to remove the barriers to independent and deliberate creation through computational design, and to ensure the successful translation of digital designs to physical forms that are compatible with craft practices.

Chapter 7

DressCode



The preliminary work of Codeable Objects and Soft Objects clearly demonstrated that algorithmic craft offers a compelling opportunity for personal, creative expression through programming. My goal following these first two tools was to address several of their primary limitations in an effort to better engage novice programers in the computational design aspects of algorithmic craft. DressCode is a stand-alone programing environment and design tool created to support new programmers in open, independent algorithmic craft. To evaluate Dress Code, I conducted two separate day-long workshops, one with experienced programmers and designers, and one for young people who were

new to programming. I also worked with FUSE, an out of school STEAM exploration program to develop a set of online activities using DressCode, which are currently in the preliminary stages of evaluation.

7.1 Design principles and Software Features

The focus on fashion in the Soft Objects workshop allowed young people to use computation to create wearable artifacts of their own design. I felt this was a particularly compelling form of engaging young people with computation, and decided to continue working in domain of fashion and garment creation for DressCode. Although the name of the tool reflects an emphasis on fashion, as an application DressCode is general purpose, and can be applied to many forms of algorithmic craft. To preserve the emphasis on fashion, the majority of the example projects and artifacts I have created with DressCode thus far, are fashion-oriented. The workshops I conducted, as well as the curriculum I helped develop also center on creating wearable artifacts.

With DressCode, I attempted to improve on Soft Objects in several ways. DressCode contains its own programing language with a simplified syntax and a limited set of textual programming methods, to allow people with little or no prior programming experience to work with the language quickly and effectively. The methods within the programing language are developed with digital fabrication applications in mind. They provide the user with the means to design freely, while ensuring that the designs created will translate easily to 2-axis fabrication machines, without the need for additional design software. The DressCode interface is developed to equally prioritize textual programing and visual design, and provide constant visual feedback for programing decisions. The environment has two-panels, one displays a graphic rendering of the user's current design, the other displays their programing code. As a user makes changes to their code, the effects on the design are rendered in the graphic panel. By providing a specialized programing language and immediate support for digital fabrication was to assist non-programmers in independently making design decisions with programing. In short, I wanted to make it as easy as possible for people to decide on their own desired style or aesthetic, and then realize it by writing their own code. The following section describes the features of the DressCode software in greater detail.

7.1.1 Interface Design

The interface of DressCode is divided into two sections, a design panel and a coding panel. The coding panel contains a primary window for entering text, an output console for print output and error reporting. When the play button is pressed, the program in the code window is run and the resulting design is displayed in the design panel. The design panel approximates many of the features

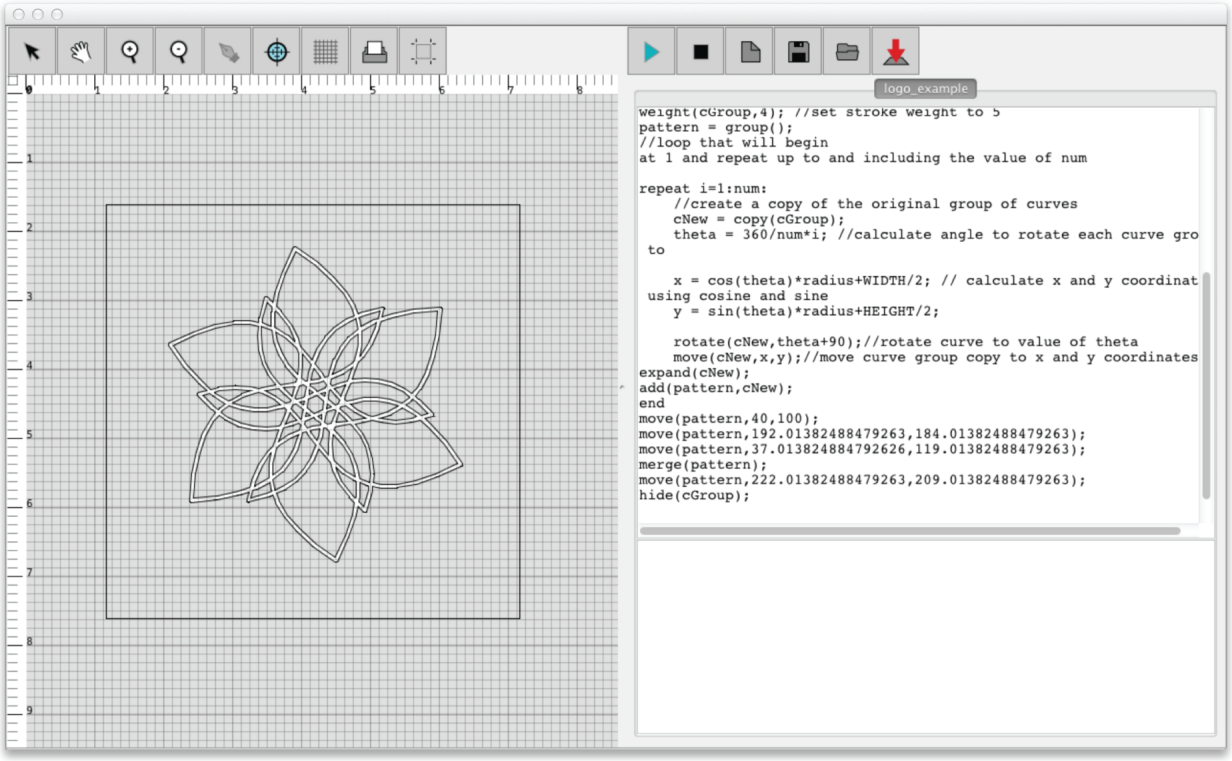


Figure 7-1: The DressCode interface

of a digital graphic design tool, with rulers and a grid, and a drawing board. The drawing board defines a reference for the coordinate system referenced by the drawing API with the upper left hand corner corresponding to (0,0) in cartesian coordinates. Users can resize the drawing board and set the units to inches or millimeters at any point during the design process. The print button opens a dialog that allows the user to export their current design in a vector format. The design panel also features a set of buttons that allow for graphic manipulation of a user's design. The target button for example, allows the user to graphically select a set of coordinates and have them appear in the programming window as text, and can be used like an eyedropper for pixel coordinates. The plus and minus magnifying glasses and the hand tool are used to pan and zoom in and out of the screen. Lastly, the arrow tool allows for design elements to be graphically selected and manipulated. After a design element is moved with the arrow tool, with the corresponding code for the move appears in the programing window. The graphic components of the the DressCode interface were developed to enable the conceptual and practical execution of digital designs for fabrication, by specifying designs in real-world units. The graphic manipulation tools were designed to be intuitive, while simultaneously working in conjunction with the programing environment, providing people with another method to understand and alter their programs. Furthermore, I theorized that by maintaining a visual display of the design with consistent updates would provide a greater incentive for people to experiment with their code, and in doing so, gain a better understanding of the

algorithms behind them.

7.1.2 Programing Language

The DressCode programing language is interpreted with semantic functionality that is simulated through a Java-based library. We relied on the ANTLR framework to generate the necessary lexing and parsing methods for the language, and developed the semantic functionality using java and the java openGL (JOGL). When a program is run in DressCode, the raw script is first tokenized and then parsed to generate an abstract syntax tree (AST). During this phase, all user-generated function definitions are stored in memory. If any parsing errors are encountered, they are output to the console as "compiler errors". Assuming the parse is successful, DressCode then walks the resultant AST and attempts to execute the semantic functionality of the program (figure:B-2.) After being executed, the resultant design is rendered on the display panel. Any runtime errors are displayed in the output console. For most programs, this process is instantaneous, however some programs with complex operations require several seconds to be executed.

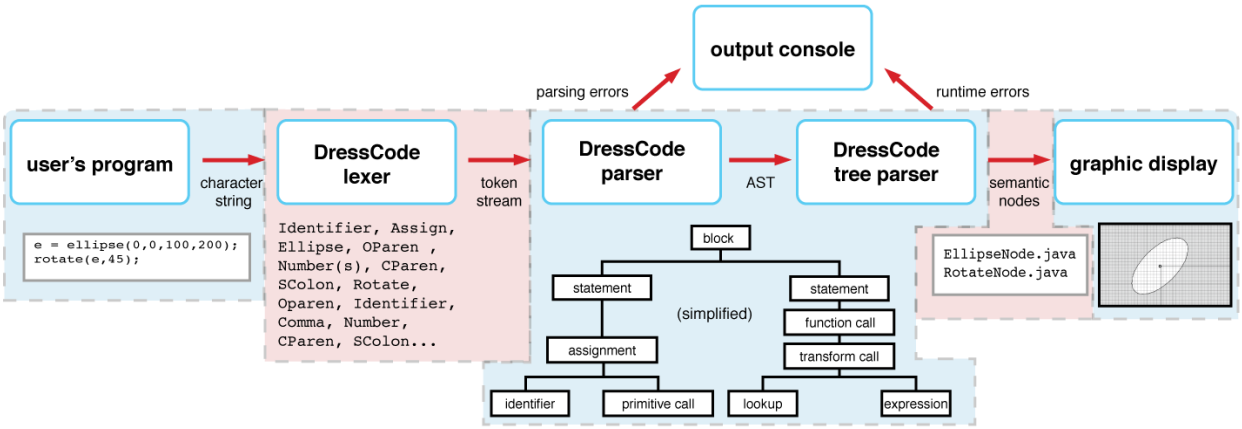


Figure 7-2: Interpreter structure

DressCode supports number, string, boolean, and drawable datatypes (figure **??**.) In the Soft Objects workshop, participants often became pre-occupied with memorizing datatypes, and had difficulty knowing which type to declare when creating a variable. To address this barrier, in DressCode, variables are dynamically typed, and can be assigned to datatypes that differ from their original assignment at any point. DressCode determines how to treat the data assigned to the variable based on context.

```
1 s1 = "hello";
  s2 = "world";
3 s3; //variable without initial assignment
```

```
s3 = s1+" "+s2;
println(s3); //prints hello world

n1 = 2;
n2 = 2.5;
println(n1+n2*10); // returns 27.0
```

The language also contains support for basic expressions, as well as block statements, including conditionals, loops and user-defined functions. There are two possible loop statements: repeat statements and while statements. Loops are extremely useful for computational design, especially when creating repeating patterns. For new programers however, loops are a difficult concept, and in many programming languages however loop syntax is often confusing. I attempted to simplify the loops of DressCode by creating an automatic update condition, and designating them with keywords I thought would more accurately reflect their function. I also added several methods that allow users to create loop-based patterns with a single statement, by specifying the number of repeats, and the resultant pattern of the loop (wave, arc, grid, spiral or ellipse).

```
//repeat statement
repeat i=0:10:
ellipse(0,i*10,10,10); //draws a vertical row of 10 ellipses
end
```

Custom functions can also be defined by the user, along with any number of dynamically typed arguments. User defined functions are stored in memory and can be defined at any point in a user's program, and called prior to their definition.

```
//basic function defintion
def foo(a,b,c):
println(a+b+c);
end

//function call
foo(1,2,3);  //prints 6

//function with return statement
def bar(a,b,c):
return(a*b*c);
end

result = bar(4,5,6));
println(result);//prints 120
```

Drawing API

The API is organized around the creation and transformation of 2D shape primitives. By duplicating and manipulating these primitives in a structured manner, it is possible to generate complex and interesting designs from simple forms. The API is composed of three primary main categories, divided by functionality: shape primitives, shape transforms, shape booleans, math operations and property access. A selection of the primary methods from each of these categories is detailed in figure B-3.

Shape Primitives	Shape Transforms	Shape Booleans	Shape Properties	Math
ellipse	move	union	getX	sin
rect	rotate	diff	getY	cos
poly	hide	xor	getOrigin	aTan
line	show	clip	getWidth	tan
curve	copy	merge	getHeight	random
point	group	expand	dist	round
import	scale			map
	mirror			

Figure 7-3: A selection of methods from the DressCode API

Shape Primitives: DressCode shape primitives currently serve as the primary means of drawing (figure: B-4.) The simplest primitive is a point, initialized two numerical parameters that denote the x and y coordinates. All closed-shape primitive methods (ellipses, rectangles and polygons) require an argument of either two x,y coordinates, or a point. This coordinate defines the origin of the shape and determines where the shape will be drawn on the canvas. Ellipses and rectangles have an additional two optional parameters which specify width and height. If only two arguments are provided, the ellipse or rectangle will be drawn with a default width and height; if three arguments are given, the width and height will be the same. Polygons also have two optional parameters following the x and y origin coordinates, however they specify number of sides and length of each side, rather than width and height. Lines can be initialized either as four numerical values, specifying start and end x and y coordinates, two points, or as a vector, with a origin point, a magnitude, and a heading in degrees. Curves are defined by a set of four points or eight coordinates, which determine the start, first control point, second control point and end point of a 4-point bezier curve. One other method of primitive generation was added in at the request of some of our early users: vector paths

stored in Scalable Vector Graphics (SVG) format can be imported and drawn in the DressCode environment, and used in conjunction with primitive transformation and boolean methods.

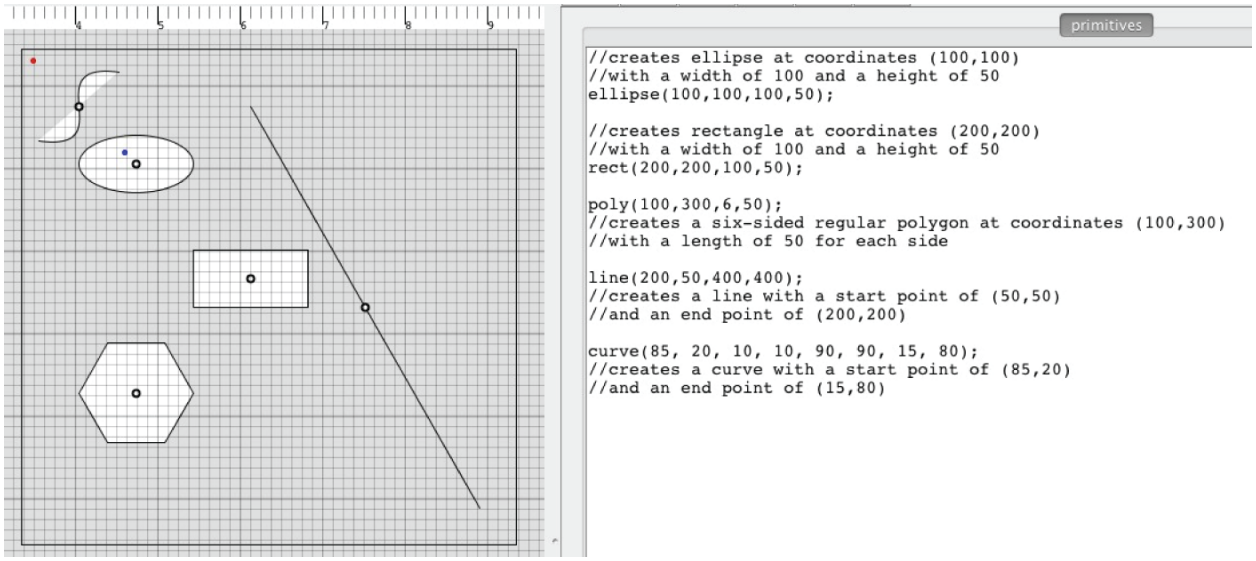


Figure 7-4: DressCode shape primitives with visible origins

Shape Transformations: There is no required "draw" method for DressCode, as there are in many other graphics APIs, Instead all primitives are automatically drawn in the display panel in the order of their initialization, once the program has been run. In Soft Objects and Codeable Objects, when we required users to manually call draw method for any and all shapes in their program, they often forgot, and then had difficulty determining the reason why parts of their designs were not visible.

Each shape primitive method returns an instance of the primitive it creates. A programmer can either create shapes with no reference as in figure B-4, or they can assign them to an identifier at initialization, allowing them to reference and modify the shape later in the program. The shape transformation methods allow for a range of modifications to a shape, by passing a reference to the primitive the first argument, followed by different parameters for the modification, depending on the method (figure: B-5.) We began with a basic set of transformations, including move, rotate, methods that allowed for the modification of the stroke weight and color of the shape, and a hide method that prevented a primitive from being drawn. Based on our early user testing however, we gradually added in additional methods, based on what people wanted to do with their designs. This included a moveBy method, scale and mirroring functionality, a copy method, and an addition to the rotate method which allowed a shape to be rotated around a point other than its origin.

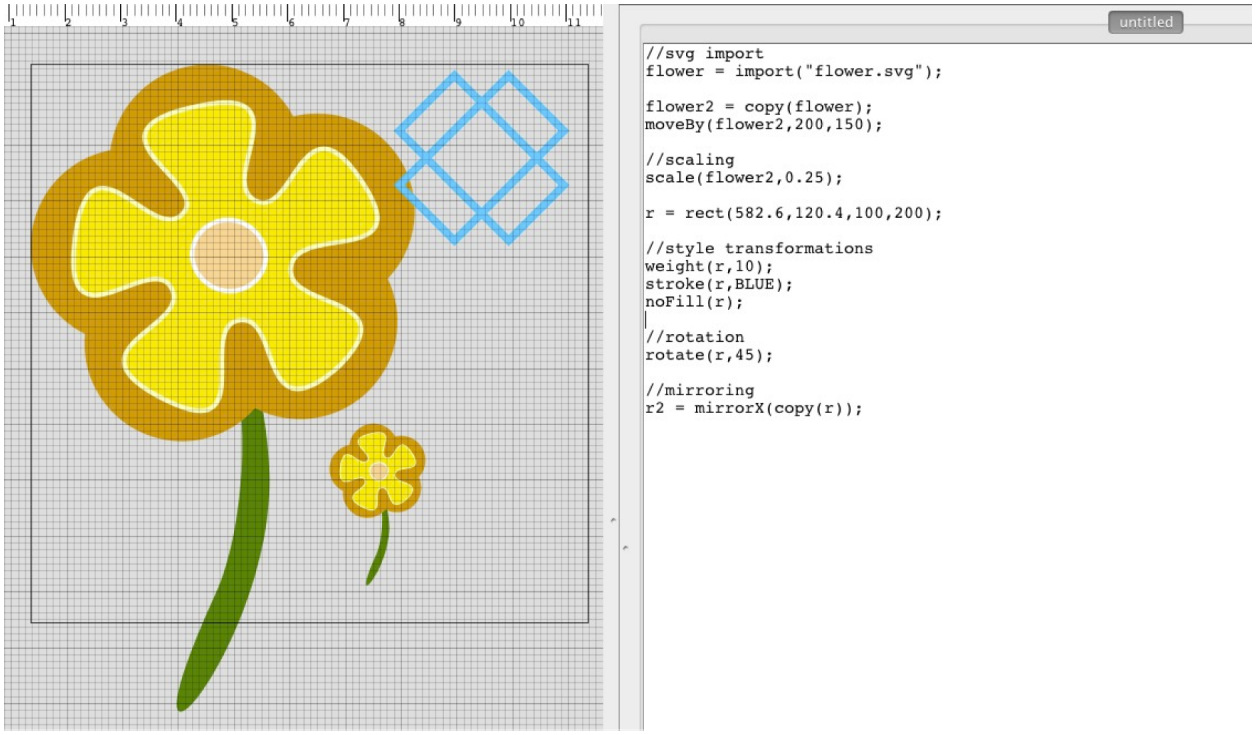


Figure 7-5: SVG import and sample of transformation methods

DressCode primitives can also be programmatically "grouped". Grouping provides an organizational structure for more complex configurations of primitives allows for the automatic transformations to multiple shapes. Groups have their own origin, which is used as the reference point for all move, rotation and scale methods applied to the group (figure: B-6.) For example, a collection of ellipses that is grouped and then moved to the center of the canvas would move the origin of the group to the center, and all the ellipses relative to the new origin. The origin of a group is automatically calculated as the centroid of all of the objects in the group and updated each time a primitive is added or removed. Groups are treated similarly to lists in that individual objects can be accessed and manipulated by via their index. Originally, primitives within a group were transformed relative to the origin of the group, however this caused confusion among many of our initial users. Each individual primitive is now translated according to the coordinate system of the canvas, regardless of whether it is in a group.

Polygon Booleans: DressCode also features a subset of transformation methods for polygon boolean operations (figure: B-7.) The union, difference, exclusive or and intersection methods accept two primitives as arguments and return the corresponding shape, or group of shapes, depending on the operation. In addition, the merge method performs a union on all objects within a group

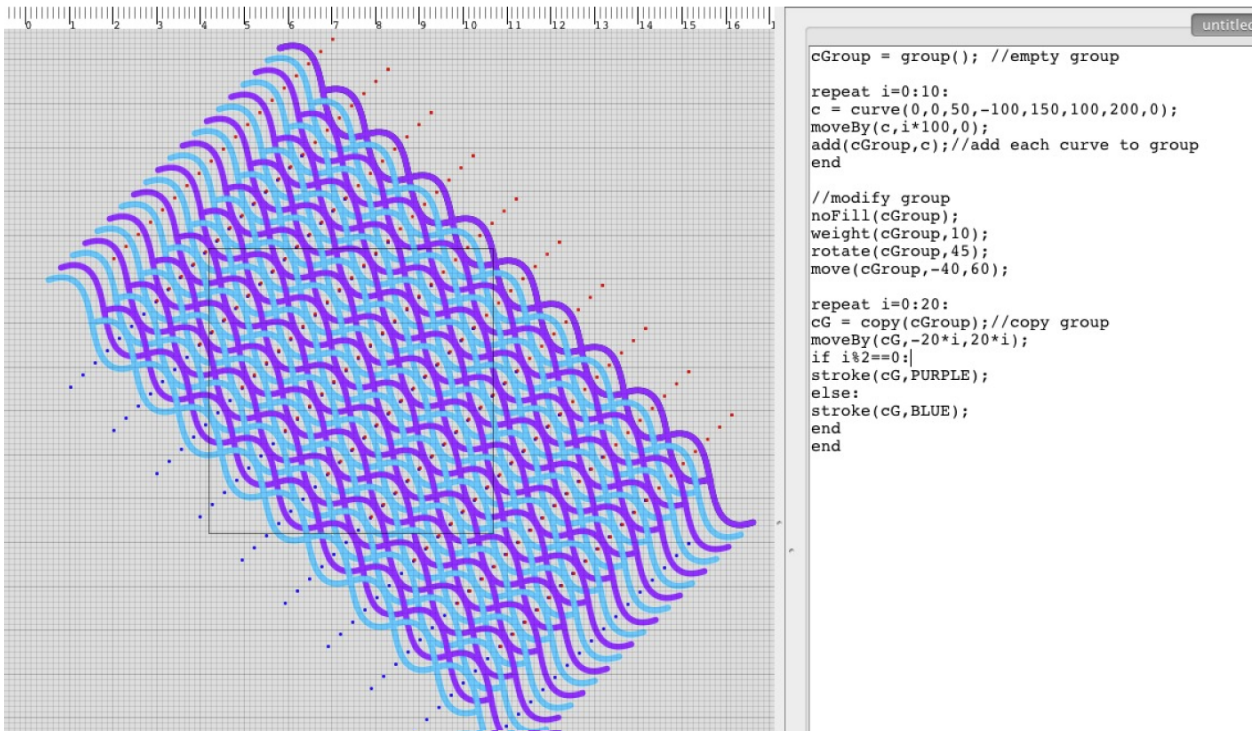


Figure 7-6: Grouping with transformation and repeat statements

and returns a single primitive. The expand method converts an object's stroke to a filled polygon with dimensions that match the weight of the stroke. The boolean methods were added for multiple reasons. Polygon boolean operations play an important role in CAD design for fabrication, because they allow for the creation of vector paths that appropriate for subtractive manufacturing processes. In particular, the merge method can function as a catch-all method for preparing designs for fabrication, and the expand method is useful for converting line drawings to paths that will retain their appearance when cut. In prior workshops, we relied on illustrator to prepare file paths for manufacturing, which was difficult, time-consuming and a hindrance to the computational design process. By integrating boolean operations as a part of the programming language in DressCode, it not only prevents users from having to rely on multiple software tools to design, but allows booleans to be used in a computational context, facilitating the creation of a whole new range of forms with simple primitives.

Both the syntax and functionality of the boolean operations have been modified significantly since the first version of DressCode. In early versions, the majority of the booleans were expressed through mathematical notation, rather than with built in functions (e.g. ellipse1 + ellipse2 would perform a union). After some of our early user testing however, we depreciated this syntax, because many testers felt that mathematical notation was not intuitive beyond the union and difference operations. In addition, the mathematical notation required that all booleans be set equal to an

identifier (e.g. ellipse1 + ellipse2 = e3), in order to evaluate as a valid statement. The new method syntax still returns a resulting boolean, but can also be called without a corresponding assignment statement. Semantically, we originally had the boolean statements operate in a destructive manner as is the case in many graphic drawing programs; union(ellipse1, ellipse2) would destroy the original two ellipses and set the identifier ellipse1 equal to the resulting boolean. This was confusing however, because in a imperative program, the programmer could still reference the destroyed ellipse2. What we opted for instead, was a non-destructive boolean method wherein union(ellipse1,ellipse2) would return a third shape, which would automatically be drawn on the screen, and could be assigned to an identifier as needed. The original ellipse1 and ellipse2 would be hidden, but could both still be referenced without conflict, and could be revealed again on the canvas with the show() method.

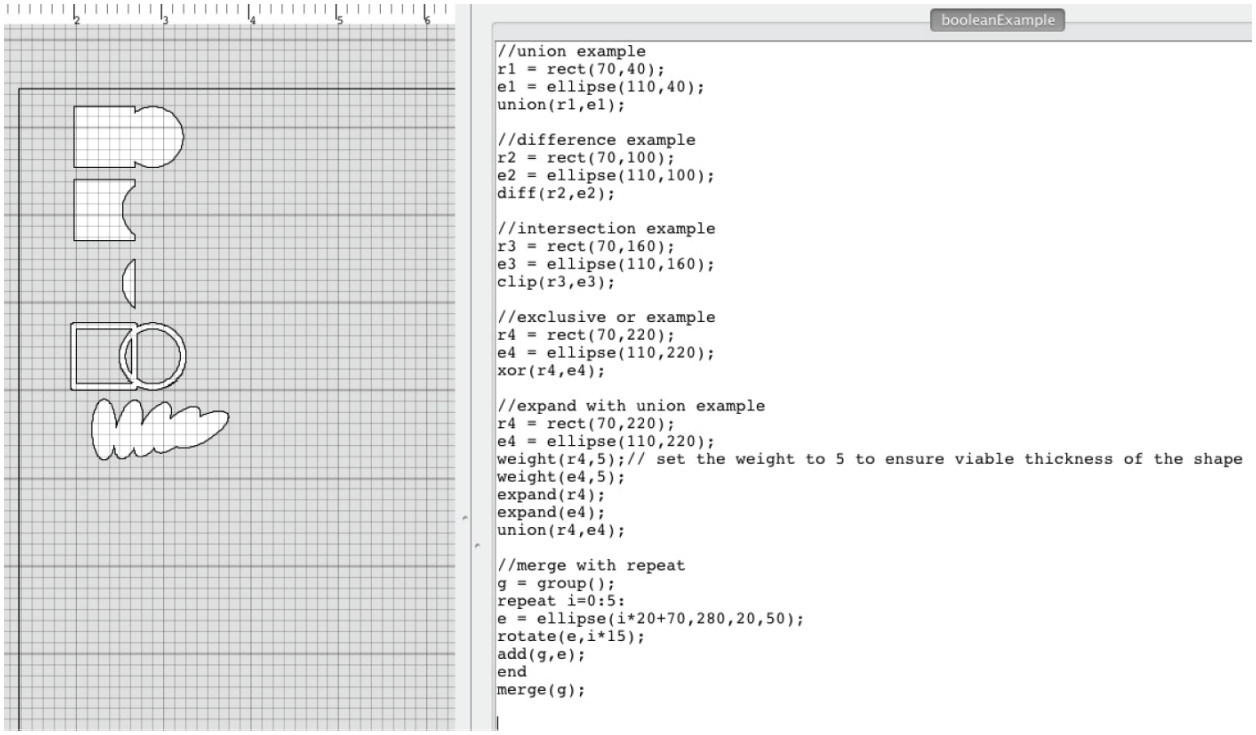


Figure 7-7: Polygon boolean operations

Additional Methods: Aside from methods for shape generation and transformation, DressCode features a set of getter methods that return information about a shape primitive, including its location, origin point, width and height, as well as a method to determine the Euclidean distance between two objects. There are also a set of methods that enable more advanced mathematical operations, including trigonometric functions, mapping, rounding and random number generation. The random method posses an interesting questions for implantation, as it generates different num-

bers each time a a program is run. This can be a useful design feature, allowing for the generation of numerous patterns with random variations. Figure B-8 demonstrates variations of an algorithm that produces randomly symmetrical stripe patterns. It can also become problematic however, when the user wishes to keep the same randomly generated number for each interpretation of their program. Finally, we recently added a set of methods that allow the specification of numerical values in inches, centimeters and millimeters, or in the current units of the drawing board. These methods are helpful when sizing and transforming primitives to real world measurements rather than pixel values.

Constants: DressCode has a small set of constants to assist with design. WIDTH and HEIGHT correspond to the width and height of the current drawing board. PI corresponds to the numerical Pi value. Lastly, there are a set of color constants; RED, GREEN, BLUE, PURPLE, PINK, etc., that can be used to define the color of objects with fill and stroke methods. Custom colors can also be specified with 0-255 RGB values.

Figure 7-8: Stripe pattern variations using the DressCode random method

7.2 Fabrication and Crafting Process

Similar to the prior tools described in this thesis once a design is exported from DressCode, it can be imported to the control software of any 2-axis fabrication machine and fabricated into a set of parts.Depending on the material and fabrication process, a variety of end products are possible. In the process of protopying DressCode, I experimented several different fabrication machines, crafting techniques and artifacts. I tested DressCode primarily with the laser-cutter, however I also used it with more accessible machines including low-cost craft vinyl cutters and inkjet printers. I also examined sending DressCode files to online fabrication services such as Shapeways and SpoonFlower, to produce 3D-printed jewelry and larger inkjet printed garments. I tested the resultant pieces from these fabrication processes with different crafting techniques that ranged in difficulty. Highly accessible techniques included making artifacts that required basic paper-folding or iron on appliqus for clothing. More difficult techniques included sewing garments, screen printing onto fabric, and jewelry making. There was frequently correlation between how easy a craft was to create, and how well it corresponded with the DressCode design process, indicating that for more sophisticated crafting processes like sewing, more advanced organizational functionality might be required for the DressCode software.

To provide some support in the crafting process for independent users, my undergraduate research assistants and I created a set of example templates in DressCode that are directed towards specific end projects and fabrication machines (figure: 7-9). We also began documenting some of the crafting techniques required to complete these projects online [**?**]. Mainly though, I relied on in-workshop instruction or curriculum development to assist people in using DressCode to complete finished artifacts.

7.3 Workshops

We informally tested DressCode with members of the FUSE development team, and by the staff at Dream Yard, an after school STEM education space in the Bronx. Following these tests, and a period of additional development, DressCode was evaluated during two separate day long workshops at the MIT Media lab. The first workshop was conducted among experienced designers, artists and programmers, who were selected individually from MIT, Harvard and the Cambridge community. The second workshop was conducted among ten young adults selected from youth organizations in Cambridge and art and craft workshop mailing-lists at MIT.Both workshops used DressCode to create laser-cut leather crafts.

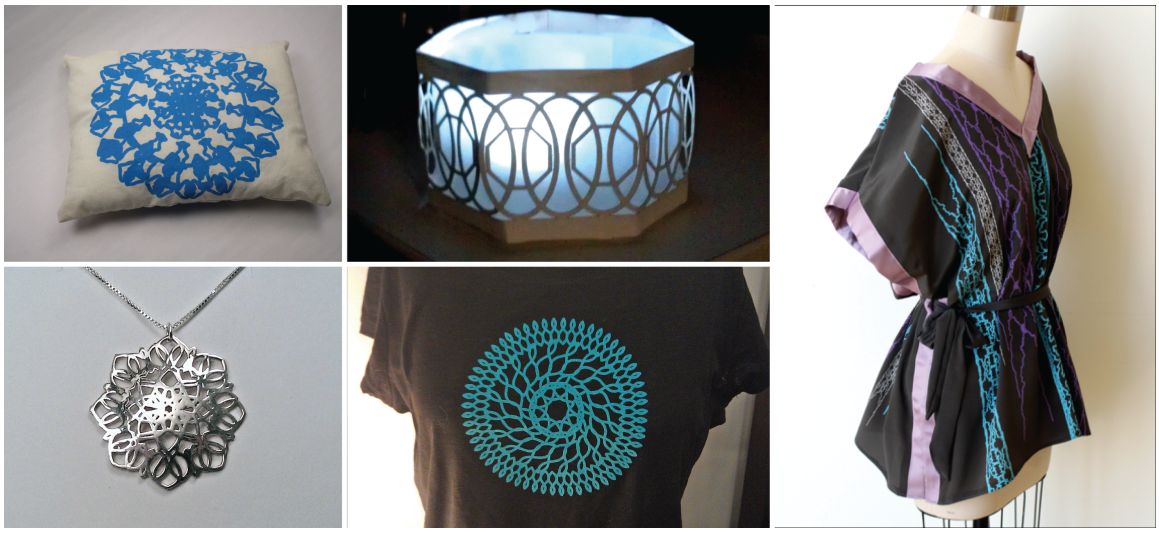


Figure 7-9: Several example artifacts created with DressCode Designs (clockwise from top left: vinyl-cut screen printed pillow, vinyl-cut tea light, ink-jet printed silk caftan, laser-cut iron-on tshirt, 3d printed silver pendant)

7.3.1 Expert Workshop

The expert workshop was comprised of eight participants between the ages of 19 to 33. Five of the eight participants were female and three were male, and all had prior experience in programming. 75% of the participants indicated a strong prior interest in design, and 50% enjoyed crafting. All participants had formal training at a college level in either art, design or computer science

The expert workshop was deliberately open and informal. We began the workshop with an open 30 minute discussion on the connections and intersections between design, arts and crafts and programming. Following this discussion the, participants were given a 45 minute general tutorial on the basics of the DressCode environment and programming language, after which participants were given access to the DressCode online code reference and a physical handout with quick references for some of the key methods. We also provided with a leather bracelet template and several example programs that generated different patterns for the template as examples. Participants were given the option to design within the template or rely on modifying these examples, but were also free to create whatever they wanted. Following the programing tutorial, participants were shown the basic fabrication process, and introduced to the materials, consisting of several varieties and colors of leather. Participants then had 3-4 hours to design their piece. As they completed their designs, they were taken in groups to the shop to fabricate their parts and then given access to craft tools and a variety of fasteners (snaps, rivets and jewelry connectors) to convert them into finished artifacts.

Figure 7-10: The expert workshop

7.3.2 Youth Workshop

The youth workshop was conducted among ten young adults, aged 15-17, 20% male and 80% female. Five of the youth participants were participants in the Teach to Learn, Learn to Teach program in Boston cite program and find a description. Of those surveyed three participants had prior experience with Scratch, and two had worked with the Arduino programing environment in a prior Media Lab workshop. All of the participants said they had some prior experience in art, design, or craft. Prior to the workshop, 60% of the participants indicated that they did not feel comfortable programing on their own, however the majority indicated that they were interested in learning more about the process.

The youth workshop was more structured than the expert workshop with the goal of helping to familiarize participants with the possibilities and affordances of computational design and fabrication. We began by passing out three colors of post-its for the categories of of art and craft, design and programming. We then held a 5 minute brainstorm where we asked each participant to write down people, tools, ideas and projects the associated each category on the respective colored post-it. After, we collected the post its, and put them up on a board in related clusters, and used the resulting associations to instigate 30 minute discussion on the participants ideas about the connections between the three domains. We ended the discussion by talking about some of the possible craft applications of computational design, and began the introduction to DressCode.

Figure 7-11: The youth brainstorm and discussion

All of the design work in the youth workshop was structured around the concept of radial symmetry. I chose radial symmetry because it is a relatively easy pattern express mathematically, it serves as an excellent motivation for using a loop, and it is a pattern that is commonly found in nature, making it possible to generate designs that are floral,biological or organic in appearance. In addition, radially symmetric forms are often aesthetically appealing to people. To begin, I explained the concept of DressCode and then spent 20 minutes leading the group, step by step through the process of creating a basic algorithm that used a repeat loop to create and rotate a number of ellipses around a central point (figure: 7-12.) After every participant had successfully created their version of the algorithm, I showed them how they could use the same algorithm to create different designs by changing the number of iterations of the loop, the type of shape that was drawn, and the width and height of the shape. Participants then had 15 minutes to experiment with different parameters, and produce different designs, followed by a group show and tell session (figure: 7-13.)

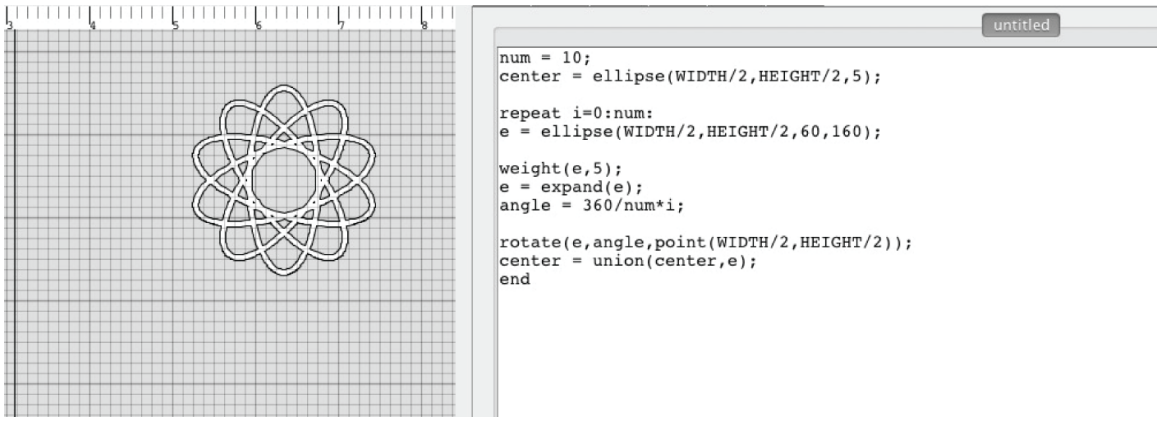


Figure 7-12: The first algorithm learned by the youth participants

Figure 7-13: Some example variations on the first activity

Next, the participants were provided with example programs that contained a bracelet template. The example programs featured a pre-written function that duplicated the algorithm they had written in the previous activity, with arguments corresponding to same values they were manipulating before. Participants were then given time to use the function to make multiple elements of the original radial pattern with different parameters. A second tab in the example program revealed the code for the radial function, which they could modify if they desired. The example template was constrained so that the dimensions of the bracelet would automatically correspond to the size of the drawing board. By measuring their wrist and resizing the drawing board, they could generate a bracelet of the correct size, and maintain their original design (figure: 7-14.)

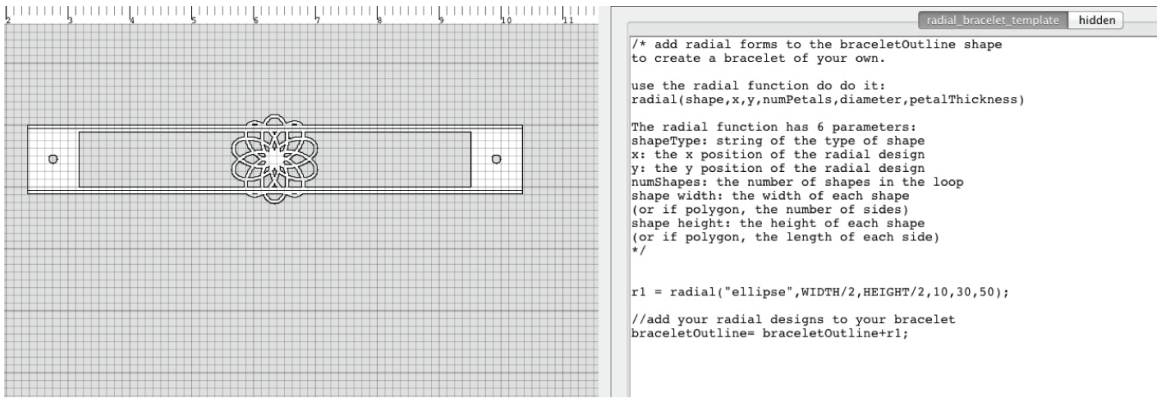


Figure 7-14: The bracelet template with one instance of the radial function being called

Participants were given approximately two hours of free design time. When they had settled on a final design, they first printed out a paper copy and ensured that it fit their wrist. Following that, they selected their material and were taken in groups to the laser cutter where they cut their piece. Afterwards, they were shown how to attach the fasteners to the bracelet, and provided with additional handcrafting tools that allowed them to further alter the piece by hand.

7.4 Results

7.4.1 Expert Results

All but one participant in the expert workshop indicated that they were able to successfully able to complete a finished product. The types of end products varied. While most participants produced leather cuffs, people also created finger puppets, earrings and a belt using the software. Many participants also mixed materials and techniques and created There was a high degree of variation in the algorithms used to create the finished products. Several participants relied exclusively on the radial symmetry algorithm, but others created patterns with computationally ordered curves, rectangles and lines in different arrangements. The participant who created the leather belt designed the pattern so that it demonstrated the evolution of the algorithm that comprised it; as the individual patterns progress across the length of the belt, they grow in complexity. Because of the high degree in variation among approaches and end products, conducting the expert workshop was difficult. Many of the participants began with design goals that were at odds with the affordances of DressCode, such as creating representational characters in the case of the finger puppet example, or creating organic, irregular forms. In addition, several of the end projects required instructor intervention to ensure they were suitable for fabrication. Overall, the expert workshop produced a promising set of results and demonstrated the limitations and benefits of using DressCode across a broad range of design approaches. The expert participants also provided feedback on specific features to incorporate into future versions of the DressCode software. The random method and unit conversion methods were added directly following the workshop based on participant suggestion.

7.4.2 Youth Results

The youth workshop was more structured than the expert workshop, and produced a more consistent set of physical products as a result. All participants in the youth workshop were successfully able to use DressCode to produce a unique leather cuff (figure: 7-16.) Despite the consistency in form, the patterns and approaches among the resulting bracelets were highly varied. Each participant was also able to successfully write their own radial symmetry algorithm in the initial lesson and produce a variety of patterns with it. During the bracelet design portion, several participants directly modified the algorithm in the template to better fit their design goals, however all participants primarily relied on radial symmetry in their bracelet design. Eight of the participants said they planned to wear the bracelets they created during the workshop, however two indicated that they planned to give them as gifts, rather than keep them for themselves. A comparison of the pre-and post-workshop surveys showed that several of the participants' comfort in programing increased following the activity. In

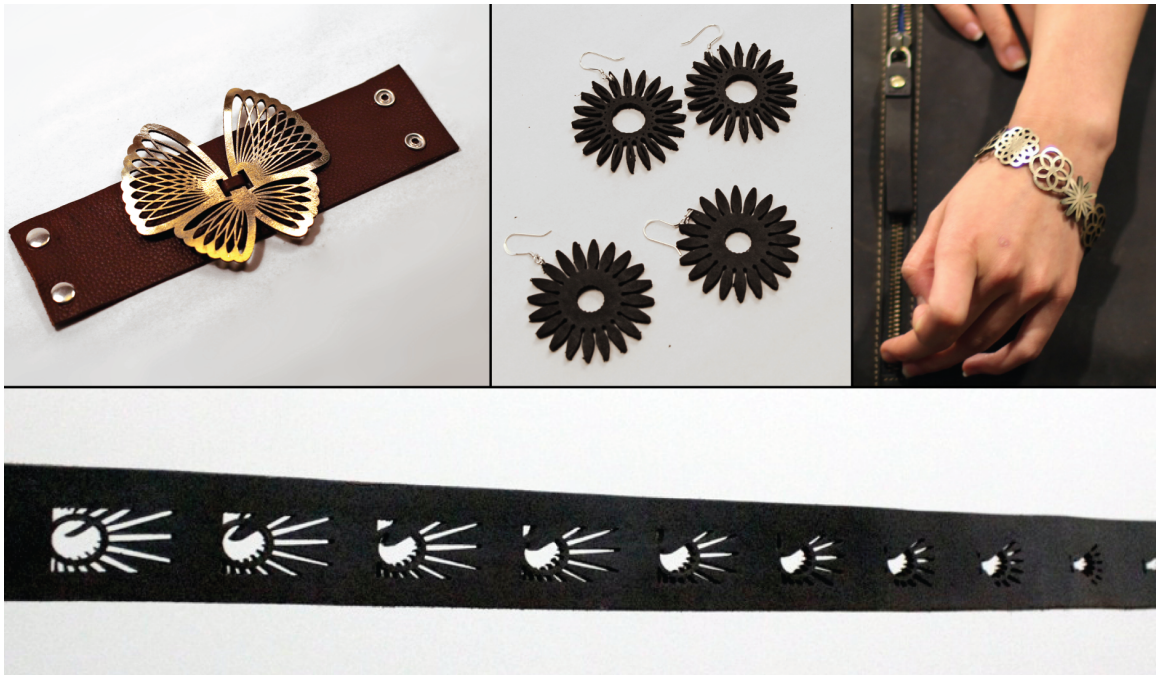


Figure 7-15: a sample of the completed projects from the expert workshop (clockwise from top left: butterfly cuff, leather earrings, patent leather bracelet, algorithmic progression belt)

addition, more participants indicated that they thought programing was a good tool for personal expression, and design respectively and all participants stated that they believed they could use programing to make things that were beautiful, whereas several had disagreed with this statement before the workshop.

Figure 7-16: a sample of the completed leather cuffs from the youth workshop

7.5 Curriculum description

7.6 Preliminary curriculum results

7.7 Discussion

7.7.1 Starting notions of craft and computation

When developing a tool for combining craft and computation, it is important to understand people's prior conceptions of these domains. In the case of the DressCode study, there was a strong distinction between the views of the two groups of participants on the respective roles of craft and computer science. During the expert discussion, two separate forms of craft were discussed. The first referred to craft as hobby and amateur practice. The experts also demonstrated an appreciation for craft as an artisinal practice, wherein handcrafted artifacts held higher value than mass produced ones because of the time and expertise required for their production. In the youth discussion of craft, the recognition of craft as artisanal practice was completely absent. Instead they referred to craft as an recreational activity, that anyone could do. The perception of craft as open and accessible by young people is encouraging, however many youth participants also dismissed the difficulty of craft in comparison to programing. One young woman articulated her view on the difference between writing a program and throwing pottery, stating that she knew she could throw a pot, but that she would be unable to write a program, despite having no prior experience in either medium. The variation in expert and youth perspectives on the levels of expertise present in craft is understandable, given the educational differences between the two groups.

There was also a difference between the amount of variation in the crafting processes between the two experts and young people. Several participants in the expert group experimented with layering different materials on top of one another for a different effect. One participant sewed a portion of her project rather than attach portions with snaps. Another used the snaps as an aesthetic component, by placing them across the length of the piece and then snapping on different colored pieces of leather in different patterns. There was limited variation in the approaches the youth practitioners took in the crafting portion. Furthermore, many of the youth participants remarked on the surprising difficulty of the handcrafting component.

> *"Something that was difficult was hammering the snaps in straight enough for them to function well."*
>
> Youth Participant F

"It was enjoyable being able to wear it and making sure that it would fit. It was a little stressful figuring out how to put the clasp on. "

Youth Participant K

"I think the actual physical copy [of my bracelet] is slightly less attractive than the one I had on the computer."

Youth Participant C

"I hate being anxious because I don't know if it's going to come out right. I like the designing on the computer, but once it comes out it looks like crap. So I'm really happy that I came to this (workshop), this is the first one that came out."

Youth Participant ST

It is likely that there is a connection between people's perception of craft, and their ability to engage in new crafting techniques, despite encountering difficulties along the way. section currently not complete

7.7.2 Enjoyment in Programing

One of the most important points of evaluation for DressCode was how successful novice programers felt in their attempts to use it. In the Soft Objects workshop, participants were excited about the potential of programing, but unanimous in their frustration in using traditional programing syntax with the Soft Objects library, and requiring instructor assistance to execute simple actions. It is difficult to evaluate the DressCode workshop parallel to Soft Objects because the DressCode workshop took place over a shorter time period in the context of a more constrained activity. When evaluated as an independent case of novice programing however, the response in the DressCode environment by novice programmers was promising. In general the youth participants expressed a sense of enjoyment and low levels of frustration with the coding portion. When asked if they found the programing enjoyable, participants responded in the following ways:

"It wasn't as hard as i thought"

Youth Participant 1271996 (survey)

"[It was enjoyable] because it was very straightforward to use and easy to understand."

Youth Participant 091897x (survey)

"Yes, I didn't have much experience with programming before the workshop so it was fun to screw around and see if I could push the boundaries of what I could achieve by hand."

Youth Participant 091897x (survey)

"[The] computer programing was enjoyable for me because i had wanted to do it before but never had the chance. so this workshop gave me exposure to something that i wanted

to do for a while. and i totally loved it."

Youth Participant 092297m (survey)

"I liked [the language], it wasn't tough, it was very simple. That barrier was removed for me so I could focus on the design."

Youth Participant ST

It should be noted that the last quote was from a participant who had prior coding experience. All of the other quotes came from participants who were new to programing. Some of the participants did express difficulties, however they were often described in conjunction with enjoyment or with the the recognition that through some effort, they could be overcome:

"The computer programming was enjoyable for me; however; it was a bit frustrating and confusing at times."

Youth Participant 123097f (survey)

"It was fairly enjoyable, if arduous and frustrating. Once I figured out how to do it it wasn't too hard."

Youth Participant 100397c (survey)

"It was alright except a bit hard to learn but once you know it its pretty much simple for the rest of the time."

Youth Participant 05231997 (survey)

Other youth participants also elaborated on the ways in which they felt programing had been useful in the design process. Their responses evoked some of the key affordances of computational design discussed in section 2.1. For example, one participant described the advantages of programing for creating precise design elements:

"Basically with programing, it like- say I wanted to draw it by hand with the mouse- it would be a lot harder to make the image um , perfect's not the word..everything was scaled perfectly. Say you wanted certain shapes to be certain angles or certain sizes, if you were to code it, you can get it at the right angle, where if you were to draw it by hand, you wouldn't know what angle it is." Inteviewer: *" So the precision?"* Youth Participant N: *"Yes."*

The recognition of the ability to create complex designs through repeated simple actions was another common sentiment. Again from the interviews:

"[My design] feels complicated and simple to me at the same time- it may seem simple to me because I know the rules behind it."

Youth Participant R.

Participants also discussed the parametric qualities of designing with code:

> *"[I liked] the programming part.. learning different strips of code and what they did, and just changing the basic codes they gave us so basically it looks more unique"*
> Youth Participant N

> *'Something that was useful about using programming to design this bracelet was that it was easy to change the sizes of the shapes very quickly."*
> Youth Participant 123097f(survey)

Responses like these suggest that DressCode was successful in introducing participants to some of the qualities of computational design, similar to the SoftObjects workshop. Perhaps the greater success was that participants were able to implement these qualities in a more independent manner than with Soft Objects, allowing for a greater sense of personal agency, and a broad appreciation of the power of computational design. This appreciation was indicated in part by the enthusiasm of the participants following the activity:

> *"Now i know that you can use computers and programing and stuff to design pretty and physically appealing things, instead of just complex robots and stuff."*
> Youth Participant 092297m (survey)

> *"It was really fun and I gained a lot of knowledge from it- knowing how to program, knowing different types of ways to manipulate shapes and radials."*
> Youth Participant S

> *"[The bracelet] is a physical manifestation of my success in coding, maths, and programatic design, and that's what I see when I look at it. It is a little coding confidence totem that can inspire me to continue developing my skills and passion in this area - wow, I didn't realise I felt that strongly about it!"*
> Expert Participant 090585p (survey)

These positive reflections on the potential of programing, combined with the appreciation of specific computational affordances in design, and feelings of accessibility and feasibility of the programming process, are encouraging for a day-long programing activity with first-time coders. An analysis of these successful results in the context of the prior workshops with Soft Objects and Codeable Objects point to several distinguishing factors. The first is related to the feature-set of DressCode itself. Although the tool still shows significant room for improvement, it appears that the simplified programing syntax, improved drawing API, and more-immediate visual feedback accounted for some of the success of the activity. By providing the participants with a small set of useful programming methods that could be used to generate complex forms and patterns, textual-computational design became relatively feasible for novices. In addition, the features in DressCode

that made the transition from design to fabrication less convoluted (by not requiring any post-processing), reduced the technical concerns of the participants and allowing them to focus on using programming to design, rather having to ensure their patterns would work in a practical sense.

Another key factor in the success of the activity was the structure of the workshop itself. By relying on radial symmetry as a design mechanism, participants were given a task that had a wide range of design possibilities, but only required them to understand a few key concepts. This differed significantly from using the Voronoi diagram in Codeable Objects- an algorithm that is easy to explain in concept, but extremely difficult to implement. It also differed from the approach of the SoftObjects workshop, where the wide variety of computational approaches made it difficult to simultaneously ensure that participants designs were successful and that people understood the algorithms behind them. When a longer-term workshop is conducted with DressCode, I will attempt to expand upon the constraints of the first workshop by selecting an additional set of algorithmic patterns similar to radial symmetry in simplicity and design feasibility. Patterns such as spirals, waves and basic fractals offer some possibilities, but there are many others.

One final element in the success of the DressCode activity was the presence of several well-prepared facilitators. For the workshop, I had four facilitators, including myself. The three additional facilitators had prior experience working with DressCode and were included in the development of the activity itself [1]. While a part of the facilitator assistance came in the form of addressing bugs and answering syntax questions, they were also invaluable in providing support in the design process. The other facilitators assisted participants by suggesting new design directions, helping them to evaluate their designs, and being supportive and encouraging throughout the day. Although it is desirable to strive for developing novice programing environments that allow for independent use, the benefit of well-structured in-person support should not be overlooked. Especially in the context engaging people first-time programing experiences with computational design, it is beneficial to have support staff who can simultaneously provide both technical and critical assistance.

7.7.3 Design Processes of Youth and Expert practitioners

Because DressCode was developed to support personal visual expression through programing, a significant part of my evaluation focused on the design strategies that emerged through the use of the tool. The interviews and post surveys indicated that participants in both the expert and youth workshop engaged in a variety of design strategies in working with DressCode. When asked about her design, one young woman described a process of conscious planning. For example:

> *"Well I wanted this one (pointing to an element of her bracelet) the first one made out of radial ellipses; I wanted it to look like a flower when they layer down, and the*

[1] It should be mentioned that I did have several wonderful facilitators for the Soft Objects workshop. Their presence was more sporadic however, and their involvement was not as deliberately planned as in the case of the DressCode activity.

second one is kind of like the same thing, but I wanted to have the pentagon or hexagon, and the next one is the snowflake, and I wanted it to look like a snowflake because my favorite season is winter, the next one the center looks like a different sort of snowflake because all snow flakes are different, so I chose different shapes, and the last one is a rectangle-like a square in the middle and then it's like hexagons."
Youth Participant S

She also described the activity as requiring "creativity and thought and dedication", due to the high degree of complexity. When asked how she found programing useful in her design she said:

"You type it in and it brings it to life for you. You can do it on your own....you don't have to buy it. It's different than the casual way that somebody gets a bracelet."

Other participants also indicated that impromptu choices played a role in their final design, in combination with pre-mediated decisions:

"A few days before the workshop, I was thinking about how I will not likely wear a leather bracelet, but a belt I would definitely wear. I didn't really have an idea for what kind of algorithmic design I wanted to make. During the discussion at the beginning of the workshop, somebody used the word "evolution," and I started thinking about making a design that repeats and gradually changes across the belt. I had a hard time coming up with an interesting shape with an interesting transformation. I started with a blossom of rectangles, and tried a bunch of variations. At some point somebody said it was like an animation, which gave me some positive energy in thinking about the design (and I thought I could be a human zoetrope, spinning in sync with a strobe). It started to get more interesting when I added the masking effect. I eventually made the individual pieces making up the ellipses instead of rectangles, which both made the thing softer, and emphasized the transition as they collide with the mask, forming sharp corners."
Expert Participant 071279R (survey)

"I saw that the duplication of a large shape was kind of happening with a lot of people's designs, so I decided to try and do something a little different. (Show's a first design (ellipses side by side on a line). That was the second phase of this, and then I just started screwing with numbers to see what I could get and I came up with this. I didn't know if I wanted the entire thing to be symmetrical or not, but then I thought that that was getting too boring (the symmetry), so then I changed the number of shapes in the loop to an odd number instead of an even one so that it would offset the design, and came up with this."
Youth Participant R

'I kind of like started off playing around with the main stuff and once I got that I narrowed it down and tried to switch around different parts to see which parts worked best."

Youth Participant C

'I used an example I liked and decided to spend time doing an earring or single pendant so I could focus on manipulating the example to see how the program worked. I also picked something that repeated the same unit for simplicity and the time limit."

Expert Participant 081764

Youth and expert participants varied in the levels of intention in the design process. Because of their professional experience and education, on average the expert participants approached the activity with a greater level of deliberation and were better able to articulate their design process. However, the responses from the youth practitioners demonstrate that they also felt they were able to make conscious design decisions in a programing context. Deliberate design choices were also made based on the materials available. Participants made decisions based on the material properties of the leather as it folded or bent, and considered the fact they were planning wear the resulting piece:

"The ellipse [in the design] was added once I understood that the teeth sticking inward in the design (formed by the inverse of the ellipses sticking outward) would pop out from the surface of the belt when it was curved around my waist- I saw this when we made a paper prototype. The ellipse held the teeth in place."

Expert Participant 071279R (survey)

"I deliberately made it more delicate and feminine, because I'm putting it on black leather, I just made it a lot prettier- with thinner lines- I have more thin than thick."

Youth Participant J

"Well, I printed out the different widths of the line because if you're going to wear it, you want it to be durable. The thinness of it, I don't want it to be easily ripped, rather than if I was like a thing to hang on a wall, I could just do anything I wanted, because it's not like it's going to have to withstand any elements. And you want it to look nice because I care about what I wear, cause what you wear, you're presenting an image of yourself to people."

Youth Participant R

"Instead of doing something that I thought would look cool, I tried to think of something I would generally wear, I didn't pick bright pink, because I knew I would never wear that."

Youth Participant K

A few adult and youth participants relied on an experimental design approaches, or came to their final design through trial and error:

> *"You just have to fool around with it until you like how it comes out."*
> Youth Participant G

> Youth Participant S: *'I just typed it in and started messing around with this, and then I gave it a go."*
> Interviewer: *"How did you know you were done with it?"*
> Youth Participant S: *" I ran out of ideas."*

> *"[I decided on the design] mostly through trial and error, I found what was easy/reliable to do but looked interesting. I spent some time rounding out the form of the bracelet, this was the most directed portion of the design process."*
> Expert Participant 081764

Experimentation through trial and error can be a useful tactic, however too much reliance on this form of design can result in a diminished sense of accomplishment. As one participant stated, he enjoyed the design process at first, but for him:

> *"the more "trial and error" kind of ruined it a little. The more anyone messes up with anything it kind of takes away the fun from it of it. But then the end result made up for it. "*
> Youth Participant N

Control over one's design plays a large part in the experience of the designer. Whether their choices were made through forethought, or realized mid-process, that ability of participants to realize their design objectives through programming had a positive impact on their experience. The participants who articulated intention in their design process emerged with a greater feeling of ownership over the finished artifact, and a more positive view of computational design than those who felt they relied on trial and error. In the case of computational design with novice programmers, conscious design processes can be difficult to achieve. The advantage of DressCode over SoftObjects and Codeable Objects was that it enabled designers to make changes and quickly see the results of these choices in their design, allowing for a greater level of control. In the case of participants who felt they relied on trial and error, future tools should endeavor not only to make computational design more accessible, but also assist in the communicating the history of a design to a designer. This may provide them with better sense of their process, and give them better platform upon which to base future decisions.

7.7.4 Computational Aesthetics

One of the more challenging areas of study in algorithmic craft is evaluating the aesthetic qualities that result from this form of design. Several key questions are, what aesthetics conducive to computational design? What styles are difficult to achieve? How do these aesthetics resonate with young and expert designers? How do these aesthetics translate to physical (and wearable) artifacts? With DressCode, I attempted to evaluate individuals' opinion on the aesthetics of their finished pieces, and some of the motivations behind them, with more rigor than in the Soft Objects and Codeable Objects workshops. In interviews, participants were asked to describe the visual components of their design and reflect on how the aesthetic was similar to, or different to things they had made in the past. In describing their designs, a variety of associations emerged. Frequently, participants described their designs as having floral, lace or snowflake-like qualities:

> *"I have multiple designs that I'm just kind of playing around with. But I think I'm going to do this one, and by this one I mean one variation of these (style), because the only thing that changes is the width of the actual lines. So this kind of doily lace one, I guess. Yeah, it reminds me of a lace doily or something, and this would kind of offset it (referring to the leather)"*
> Youth Participant R

> *"A small flower of leather. One flower has lacy petals the other flower is more solid. The flowers are going to be earrings."*
> Expert Participant 081764 (Survey)

> *"A flower and then in the middle I want to say it resembles a butterfly or something with wings. "*
> Youth Participant N

Participants also described the aesthetic of their design in terms of their mathematical aspects. They discussed the geometric, linear and precise nature of their finished bracelets, as well attributing a general mathematical quality to them:

> *"I tried to contrast ellipses shapes with the more geometric looking ones."*
> Youth Participant C

> *"Geometric? They were "mathematically" very simple, mostly I relied on repetition and rotational transformations to create designs out of ovals."*
> Expert Participant 102287 (Survey)

> *'I think it is beautiful. It's part mathematically beautiful."*
> Expert Participant 101593y (Survey)

"I think that my design is beautiful because it is linear and intricate but not too intricate.... [It's] linear, strong, [and] geometric"
Youth Participant 091897x (Survey)

Unexpectedly, aesthetic connotations converged in one participant's piece, and produced a personally relevant narrative:

"The rose right there, and then the scientific part of it, those two things I like- nature and science, but sometimes nature conflicts with science. I want to do aerospace engineering when I get to college, but aerospace engineering conflicts with the environment in some ways, aerospace has to do with machines so that means a lot of gas is burning stuff like that. "
Youth Participant S.M.

Participants also described a mix of complexity and simplicity in their pieces. Some described their pieces as deliberately simple, whereas others attempted to contrast simple components against more complex ones, for a compositional effect.

"I think the middle part is probably the most striking and that's why I put it in the middle. It's a really intricate circle mandala thing and then just triangles"
Youth Participant C

"The [part] that sort of looks like a star, and is a little less intricate than the rest and it sort of draws the eye more than the rest of them"
Youth Participant J

"I think it's simple. Instead of having a lot of different shapes, you can make out this is one shape, this is one shape, and this is one shape, three different parts. And this other thing is a bunch of shapes here, a bunch of shapes here and a bunch of shapes here and just kind of looks odd. So yeah, it's simple"
Youth Participant N

"They are simple designs in the end, I like that the simple design modified from the sample reflects my aesthetic,. My designs could have been a lot more intricate given the sample."
Expert Participant 081764 (survey)

Because the patterns expressed in mathematics are often reflected in patterns encountered in the natural world, it is not surprising that participants associations ranged between natural terminology (flowers, butterflies and snowflakes), to mathematical descriptors (geometric, ordered, repetitive,

linear). What is remarkable is the range of descriptions present both in the context of the visual elements expressed in the designs, and the composition of the entire piece. Although computational design provides specific affordances and limitations that stem from the properties of computation, it is important not to confuse these affordances as restrictions on the types of things it is possible to create. Similar to non-digital forms of creative media; computational design presents a broad space of aesthetic possibilities, whether it is in the hands of novice programmers and young designers, or experienced programmers and trained designers. The emergence of narrative qualities from one participants design is a particularly compelling example in this regard.

To encourage the creation of personally relevant aesthetics algorithmic craft tools should be open enough to support numerous creative applications. This openness should be tempered by introducing the tools in a way that provides users with a compelling motivation for their use. Algorithmic craft provides one immediate motivation through the goal translating a programmed design to a specific physical artifact, however this can be extended by providing goals for the aesthetics of the resultant piece. For future DressCode workshops, it may be useful to motivate participants by asking them to create artifacts which express a specific feeling or emotion, represent a character, or tell a story. Design briefs of this nature may further people's conception of how computational design can be used to create personally relevant artifacts, and result in an even broader set of aesthetic possibilities in computational design. is this point too obvious?

One other important element in the discussion of aesthetics is the social connotations of certain forms and patterns. Because I emphasized designs with radial symmetry in the workshops, many of the resulting patterns had a floral quality, although it was possible to create patterns without floral qualities by using other forms of repetition, and relying on polygon primitives as opposed to elliptical forms. Despite this flexibility, several of the male participants in the youth workshop remarked on the feminine aesthetics of their finished pieces:

> *"I put it on my wrist and I was like wow that's feminine. That's very feminine."*
> Youth Participant G

> *"I think it is beautiful for females. Not me. Because, I don't go that way. Because on a female's wrist it looks right not on mine."*
> Youth Participant G

Part of this feminine association may be related to the resulting artifact- in this case a wrist cuff or bracelet. Both male and female participants already wore similar accessories prior to the workshop however, so much of the gender association appeared to be the result of the patterns themselves. Although the participant above talked about the feminine aesthetics of his piece in a joking manner, there are serious implications behind his comments. As described in the discussion of the Soft Objects workshop, algorithmic craft in the context of fashion accessories and wearables

offers a powerful opportunity for the expression of one's visual identity. When helping people to create artifacts that they will personally wear, the social perceptions and possible stigmas of certain patterns and forms of those artifacts should be accounted for. This is especially important when working with young people, whose visual identity can be subject to criticism by their peers. Following the workshop, I worked with several male colleagues to develop non-gendered example algorithms for the bracelet design (figure:7-17.) This is not to say that gendered aesthetics are to be avoided altogether; in many cases they are both relevant and desirable. Instead, the cultural, gender and social implications of abstract computational aesthetics should be carefully considered in relation to the demographics of an intended user group.

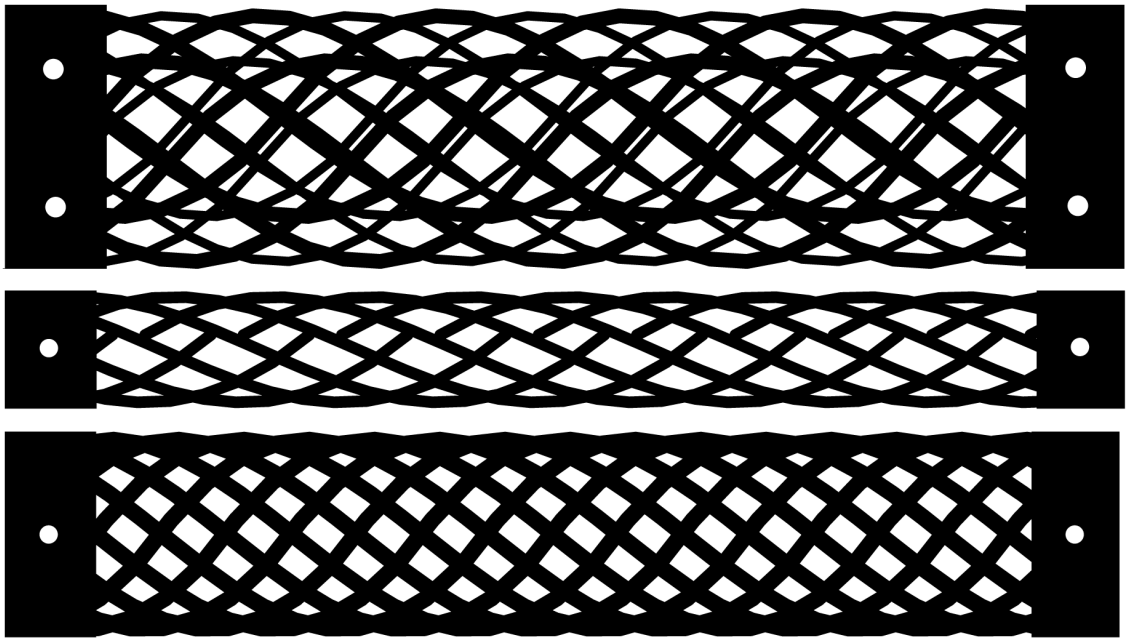


Figure 7-17: Alternative bracelet aesthetics

7.7.5 Personal value in Algorithmic Craft

Although the aesthetic properties of a finished artifact are important to consider, there are many other components of algorithmic craft that define one's experience. in all of the workshops, other factors also played an important role in the participant's overall experience. When we asked participants to evaluate their designs, they talked about more than how the object looked. Many participants described the unique quality of their piece:

"I think it's interesting. I'm assuming this design is unique to me unless someone else goes and makes exactly the same thing, which I highly doubt because of all the random

calibration things that I ended up adding."
Youth Participant C

Youth Participant C: *"You cant find it at American eagle. I buy a lot of my bracelets there, and you cant just go there and find this. I think its cool if someone were to say Oh were did you get that from, You cant find it. (Laughs). "*
Interviewer: *"You can say that you made it yourself, you love how its unique?"*
YouthParticipant C: *"yeah"*

As was the case in the Soft Objects workshops, some of the participants also described how their personal preferences played a strong part in the design of the piece.

"usually when you make something you're making it like oh I hope society likes this but this[making the bracelet] was about self satisfaction, so I don't have to take into consideration someone else's opinion."
Youth Participant S

Comments like these demonstrate participants' awareness of the value in personal craft practices, in contrast to commercial forms of production. The youth participants entered the workshop with no awareness of the artisanal craft, instead defining craft by its high level of accessibility. By the workshops completion however, the participants described some of the key affordances of craft in their own work, including uniqueness, exclusivity and personal relevancy. Another significant element discussed by participants was their surprise in personally being able to create in this manner through programing. People responded in the following ways when asked to describe their finished piece:

"II think [the programming] kind of reminded me of learning something difficult in like math, because I don't really like math, but when I figure out something, then I start doing it, and can do it and then I feel proud of myself"
Youth Participant C

Interviewer: *"What stands out to you about your design?"*
Youth Participant S.M.: *"That I actually created it. I'm not really a programing person, doing it and typing it out."*

Youth Participant J: *"It's on computers and I'm terrible at computers and for me to actually get something like this is really big."*
Interviewer: *"How do you feel about that?"*
Youth Participant J: *"Really proud."*

Finally, participants talked explicitly about the value in being exposed to programing in new ways, and their desire to continue working in computational design and digital fabrication.

"I don't know if this is my preferred medium, but I definitely like it. Making other accessories. If I spent more time playing around with it, I could definitely get something good. I'm glad I did it, it's interesting, It's something that I don't have a lot of experience in- just branching out."
Youth Participant R

"I thought it was a good experience, It was different and it sort of opens your mind to thinking about what else you could do with programing, fashion and stuff like that."
Youth Participant J

"I would love to get into more computational design, I think it's really really cool, because I'm an amateur graphic designer as well."
Youth Participant ST

It is apparent that participants not only valued the artifacts they created, but also benefited from the process of producing them. The sense of pride and accomplishment in programing echoes the sentiments of the novice programers in the Soft Objects workshop. The participants in the DressCode workshop were even more vocal about their accomplishment however, in part due to the fact that they were able to program much more independently than in prior workshops. Algorithmic Craft allows people to produce unique objects that have personal and social significance beyond their appearance. Simultaneously, it introduces people to a new context for computation, while simultaneously promoting a sense of personal technological competence. The participants' desire to further engage in similar activities demonstrates the potential for the approach of algorithmic craft to promote continued engagement in programing as a creative practice.

7.7.6 Design history and selection

7.7.7 Relation to Other Design Tools

7.7.8 Prototyping pt 2

7.7.9 Connection to other applications

7.7.10 challenges in crafting

Chapter 8

Conclusion and Future Directions

8.1 Functional Properties of Algorithmic Crafting Tools

Based on my analysis of existing CAD and computational design tools, and my study of professional computational design and digital fabrication practices, I hypothesized that the following functional properties are necessary for an effective algorithmic craft software:

- **Emphasis on computational design:** Programing should be the chief method of generating and manipulating designs, and this focus should be reflected in the interface of the software.
- **Novice-oriented programing syntax:** The programing syntax and application programing interface (API) should be designed to accommodate individuals with no prior programing experience, and should be limited to methods and structures relevant to the task of design.
- **Support of sophisticated design techniques:** Although the programatic aspects of the program should seek to cater to new programmers, the tool should work for a range of design expertise and objectives: from amateurs with an expressed interest in design and aesthetics, to those with prior design experience.
- **Design methods that facilitate digital fabrication:** The software should include programing and drawing methods that allow for the production of designs that are suitable for digital fabrication and support for exporting to relevant file formats.
- **Prioritization of visualization and simulation:** Users should be given ample visual feedback to inform their programming decisions.
- **Simple workflow from software to fabrication:** The transition from the design tool to the fabrication device should require as few intermediary steps as possible.

- **Free and open-source:** The software should be freely available, and able to function on multiple platforms with low requirements for computational processing power to afford high levels of access to casual users. If possible, the software should also be open source, in order to encourage the proliferation of additional novice oriented tools that can be developed for the specific needs of distinct user groups.

- **Domain specificity:** In order to support first-time users, the tool should be constrained to a the design of a particular set of end products. Or, if the software is general purpose, it should be packaged with a set of well documented example designs and projects that clearly demonstrate its key applications.

- **Fabrication and craft-specific documentation:** In addition to the documentation of the application and programing language, the fabrication and crafting techniques intended for use with the software should be well-documented. This documentation should include details on suitable materials, fabrication machine access and settings, and tutorials on the craft components of example projects.

magical qualities- enjoyment, dedication, pleasure, excitement, return

Processes- planning vs experimentation Prototyping The role of craft The affordances of algorithmic fabrication - how best to communicate them? The aesthetics of computational design Critique in computational design

difficulty in reconciling practice and felt experience of craft with discrete knowledge of computation

Version Control (The loss of design) Better selection mechanisims Longer-term studies Targeted audience (revised) Importance of in-person instructors- in relation to craft related activities

Appendix A

Interview Questions

Design Evaluation:

1. Can you describe your design to me? What words best capture what you created?
2. How did you decide on this design?
3. What stands out to you about this particular design? (What is the first thing that strikes you or comes to mind when you look at it)
4. What types of crafts have you made in the past? (if nothing then dont ask 5)
5. How is this design similar to other things you have made in the past? How is it different?
6. Do you think your design is beautiful? Why or why not?

Programing as a Design Tool:

1. Was the programing enjoyable for you? Why or why not?
2. What was useful about using programing to design this bracelet?
3. What was difficult or confusing about using programing to design this bracelet?
4. Were there things you wished you could have done differently? If so, what?

Fabrication:

1. How did the fact that you were designing something you were going to wear affect your design process?
2. What was your reaction when you saw your piece come out of the laser cutter?
3. What was enjoyable about putting the bracelet together after it was cut? What was difficult?

4. Out of the entire process, programing, fabricating and finishing your piece, what did you like the most? What did you like the least?

5. What are you going to do with the bracelet once youre done with it?

Future Motivation:

1. Would you want to use DressCode for other projects in the future? If so, what sorts of projects would you want to use it for?

2. Would you want to use programing and computation for projects in the future? If so, what sorts of projects would you want to use it for?

3. Would you want to work with a laser cutter again? If so, what sorts of projects would you want to use it for?

Appendix B

Full DressCode Language Specifications

B.0.1 Interface Design

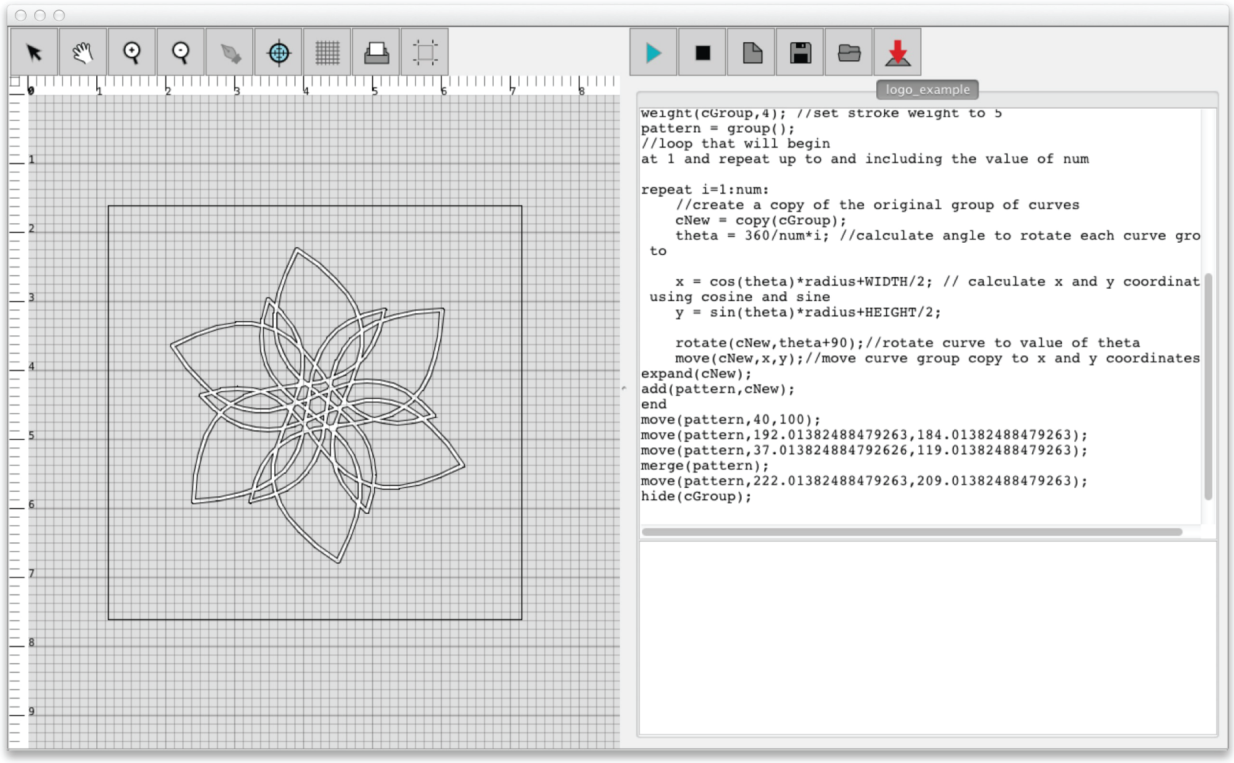


Figure B-1: The DressCode interface

The interface of DressCode is divided into two sections, a design panel and a coding panel. The divider between the two panels can be resized as needed. The coding panel contains a primary window for entering text, an output console for print output and error reporting and a row of buttons on the top. When the play button is pressed, the program in the code window is run and the resulting design is displayed in the design panel. The first version of DressCode ran the code each time the enter key was pressed or a new statement was completed, however we received many requests from our initial user tests for manual control of the run process. The stop button attempts to terminate programs that are taking too long to run.The following three buttons allow for the creation of new programs and the saving and opening of existing programs. The final button opens a dialog that enables the user to select an Scalable Vector Graphics (SVG) file to import into their script (the rationale for this is described in B.0.2 section below.)

The design panel is primarily composed of the drawing canvas. The drawing canvas has a set of rulers and a grid, along with a black rectangle in the center which serves as the drawing board. The drawing board defines a reference for the coordinate system of the canvas, with the upper left hand corner corresponding to (0,0) in cartesian coordinates. Designs can be drawn on any part of the canvas, including outside the drawing board, however the exported designs file-dimensions will always correspond to the size of the drawing board. The right-most button in the design panel allows for the specification of the units of the project, and the resizing of the drawing board. The print button opens a dialog that allows the user to export their current design in a vector format. I experimented with various vector file formats including DXF, SVG and PDF, however for the time being, I settled on PDFs as they were compatible with the specific fabrication machines I intended to use in my workshops.

The grid button allows the user to toggle between a grid in the units that correspond to the current units of the project (either millimeters or inches), pixels, or no grid. The target button allows the user to graphically select a set of coordinates and have them appear in the programming window as text, and can be used like an eyedropper for pixel coordinates. The plus and minus magnifying glasses and the hand tool are used to pan and zoom in and out of the screen. Lastly, the arrow tool allows for design elements to be graphically selected and manipulated. After a design element is moved with the arrow tool, with the corresponding code for the move appears in the programing window.

B.0.2 Programing Language

The DressCode programing language is interpreted with semantic functionality that is simulated through a Java-based library. We relied on the ANTLR framework to generate the necessary lexing and parsing methods for the language, and developed the semantic functionality using java and the java openGL (JOGL). When a program is run in DressCode, the raw script is first tokenized and then

parsed to generate an abstract syntax tree (AST). During this phase, all user-generated function definitions are stored in memory. If any parsing errors are encountered, they are output to the console as "compiler errors". Assuming the parse is successful, DressCode then walks the resultant AST and attempts to execute the semantic functionality of the program (figure:B-2.) After being executed, the resultant design is rendered on the display panel. Any runtime errors are displayed in the output console. For most programs, this process is instantaneous, however some programs with complex operations require several seconds to be executed.

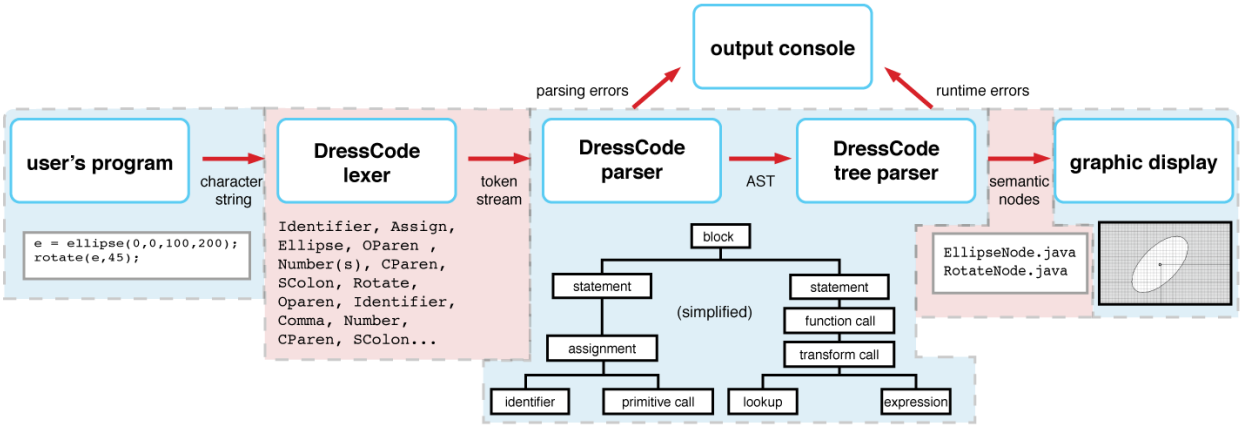


Figure B-2: Interpreter structure

The language is imperative in nature, with statements being executed in the order in which they are read. The exception to this is user defined functions, which may be defined at any point in the program, and called before they are defined. Each statement in DressCode must be terminated by a semicolon. I spent some time experimenting with terminating statements with newlines, similar to whitespace sensitive languages like Python, as I thought this might be more conducive to novice use. Unfortunately, the implementation of selective whitespace recognition is a challenging task to define in a grammar, so while we were able to develop a version of DressCode that recognized statements terminated by a line break, it would have taken significant time to develop this functionality to a level of robustness appropriate for user testing. For the short term, we have opted for a syntax requiring semicolons, however we plan to continue to explore newline statement termination in the future.

Language Structure

Here, I define some of the primary elements of the DressCode syntax and drawing API, and the rationale for their structure. A more thorough language reference is available on the DressCode wiki[**?**].

DressCode supports number, string, boolean, and drawable datatypes (figure **??**.) The first four of these types are relatively standard in comparison to conventional programing languages. Numbers include integers and floating point values. Strings include any sequence of characters enclosed in double quotations. Booleans have two possible values: true, and false. Drawables are a special type, discussed in detail in the API description below. DressCode also contains a list data structure, which can store multiple kinds of datatypes.

```
myString = "hello";  //string
myNum = 10.4;  //number
numbermyBool = true;  //boolean
myList = [10,11,false,"world"];  //list with multiple data types
println(myList[3]); //prints world
```

Variable identifiers in DressCode must begin with a letter which followed by 0 or more letters or digits. Variables can be initialized through assignment, or declared and assigned later in the program. All assignment in DressCode is dynamically typed, and variables can be assigned to datatypes that differ from their original assignment at any point.

```
s1 = "hello";
s2 = "world";
s3; //variable without initial assignment

s3 = s1+" "+s2;
println(s3); //prints hello world

n1 = 2;
n2 = 2.5;
println(n1+n2*10); // returns 27.0
```

The language also contains support for basic expressions, as well as block statements, including conditionals, loops and user-defined functions. For mathematical expressions, standard order of operations is maintained, unless parentheses are introduced. All block statements are signified by a keyword followed by a colon, and terminated with the end keyword which is not followed by a semicolon.Conditionals are defined with the if keyword, and may an optional else clause and zero or more else if clauses.

```
//if statement
if 5<10:
println("true"); //prints true
end
```

```
//if statement with else if and else clause
i=10;
if i<10:
println("less than 10");
else if i==10:
println("equals 10"); //prints equals 10
else:
println("greater than 10");
end
```

There are two possible loop statements: repeat statements and while statements. Repeat statements begin with the repeat keyword and are followed by a the initialization: a variable identifier with a numerical assignment, followed by a colon and the test, a number that determines the point at which the repeat statement will terminate. By default, all repeat statements have an update value of 1, however this can be modified by following the test value with "add" and a 3rd value specifying the update condition. While loops are initialized with the while keyword and followed by a test condition.

```
//repeat statement
repeat i=0:10:
ellipse(0,i*10,10,10); //draws a vertical row of 10 ellipses
end

//repeat with modified update condition
repeat i=0:10 add 2:
println(i); //will print 0,2,4,6,8
end

//while statement
c=0;
while c < WIDTH:
ellipse(c,10,20);
c = c+20;
end
//draws a row of ellipses that span the width of the canvas
```

Finally, custom functions are defined with the def keyword, followed by an identifier and a set of parentheses containing 0 or more arguments separated by commas. The function block is terminated with the end keyword like all other block statements. Just like general assignments, functions arguments are dynamically typed. Functions, loops and conditionals all have their own scope and will prioritize identifiers based on that, however if they cannot locate an identifier within their own scope, they will look for it in the level above.

```
//basic function defintion
def foo(a,b,c):
println(a+b+c);
end

//function call
foo(1,2,3);  //prints 6

//function with return statement
def bar(a,b,c):
return(a*b*c);
end

result = bar(4,5,6));
println(result);//prints 120
```

Drawing API

The API is organized around the creation and transformation of 2D shape primitives. By duplicating and manipulating these primitives in a structured manner, it is possible to generate complex and interesting designs from simple forms. The API is composed of three primary main categories, divided by functionality: shape primitives, shape transforms, shape booleans, math operations and property access. A selection of the primary methods from each of these categories is detailed in figure B-3.

Shape Primitives	Shape Transforms	Shape Booleans	Shape Properties	Math
ellipse	move	union	getX	sin
rect	rotate	diff	getY	cos
poly	hide	xor	getOrigin	aTan
line	show	clip	getWidth	tan
curve	copy	merge	getHeight	random
point	group	expand	dist	round
import	scale			map
	mirror			

Figure B-3: A selection of methods from the DressCode API

Shape Primitives: DressCode shape primitives currently serve as the primary means of drawing (figure: B-4.) The simplest primitive is a point, initialized two numerical parameters that denote the x and y coordinates. All closed-shape primitive methods (ellipses, rectangles and polygons) require an argument of either two x,y coordinates, or a point. This coordinate defines the origin of the shape and determines where the shape will be drawn on the canvas. Ellipses and rectangles have an additional two optional parameters which specify width and height. If only two arguments are provided, the ellipse or rectangle will be drawn with a default width and height; if three arguments are given, the width and height will be the same. Polygons also have two optional parameters following the x and y origin coordinates, however they specify number of sides and length of each side, rather than width and height. Lines can be initialized either as four numerical values, specifying start and end x and y coordinates, two points, or as a vector, with a origin point, a magnitude, and a heading in degrees. Curves are defined by a set of four points or eight coordinates, which determine the start, first control point, second control point and end point of a 4-point bezier curve. One other method of primitive generation was added in at the request of some of our early users: vector paths stored in Scalable Vector Graphics (SVG) format can be imported and drawn in the DressCode environment, and used in conjunction with primitive transformation and boolean methods.

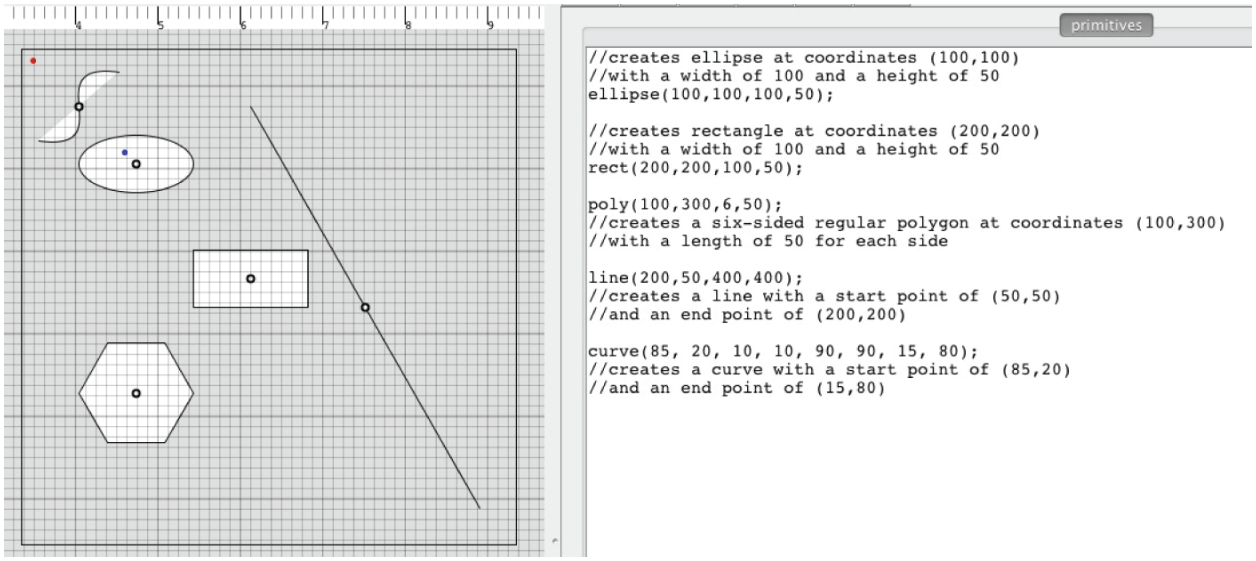


Figure B-4: DressCode shape primitives with visible origins

Shape Transformations: There is no required "draw" method for DressCode, as there are in many other graphics APIs, Instead all primitives are automatically drawn in the display panel in the order of their initialization, once the program has been run. In Soft Objects and Codeable Objects, when we required users to manually call draw method for any and all shapes in their program, they

often forgot, and then had difficulty determining the reason why parts of their designs were not visible.

Each shape primitive method returns an instance of the primitive it creates. A programmer can either create shapes with no reference as in figure B-4, or they can assign them to an identifier at initialization, allowing them to reference and modify the shape later in the program. The shape transformation methods allow for a range of modifications to a shape, by passing a reference to the primitive the first argument, followed by different parameters for the modification, depending on the method (figure: B-5.) We began with a basic set of transformations, including move, rotate, methods that allowed for the modification of the stroke weight and color of the shape, and a hide method that prevented a primitive from being drawn. Based on our early user testing however, we gradually added in additional methods, based on what people wanted to do with their designs. This included a moveBy method, scale and mirroring functionality, a copy method, and an addition to the rotate method which allowed a shape to be rotated around a point other than its origin.

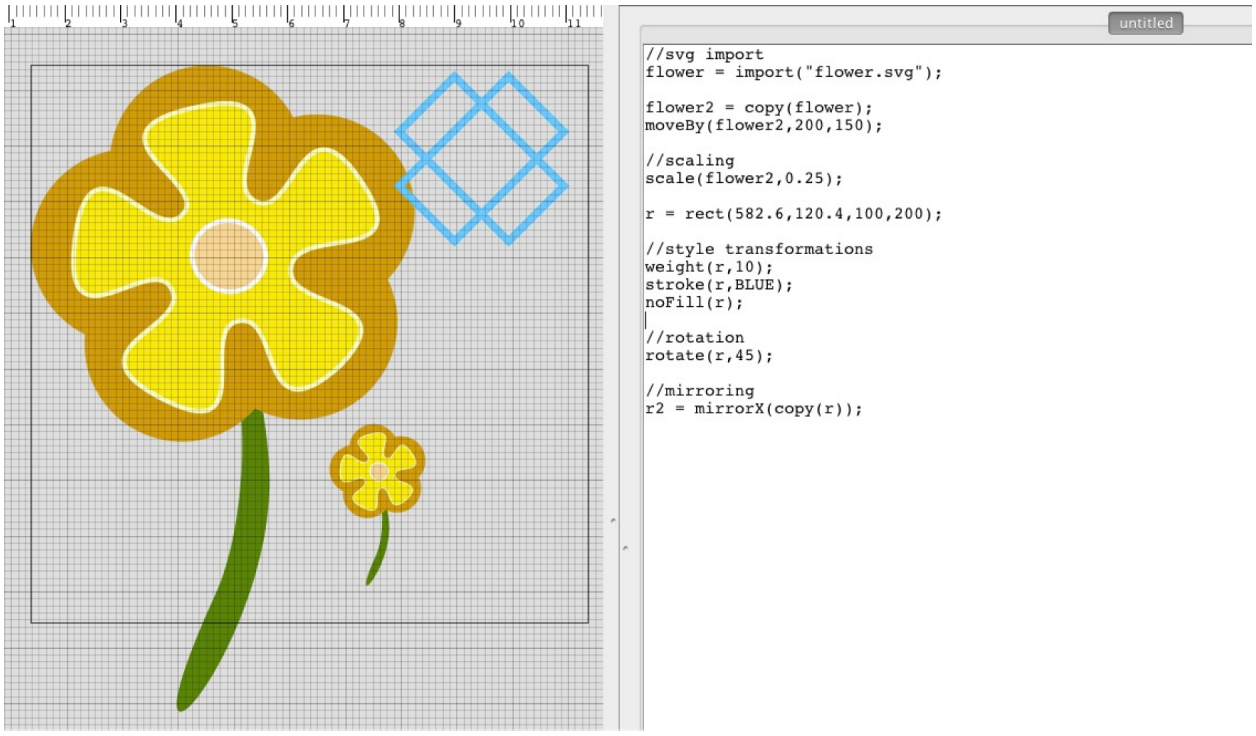


Figure B-5: SVG import and sample of transformation methods

DressCode primitives can also be programmatically "grouped". Grouping provides an organizational structure for more complex configurations of primitives allows for the automatic transformations to multiple shapes. Groups have their own origin, which is used as the reference point for all move, rotation and scale methods applied to the group (figure: B-6.) For example, a collection of

ellipses that is grouped and then moved to the center of the canvas would move the origin of the group to the center, and all the ellipses relative to the new origin. The origin of a group is automatically calculated as the centroid of all of the objects in the group and updated each time a primitive is added or removed. Groups are treated similarly to lists in that individual objects can be accessed and manipulated by via their index. Originally, primitives within a group were transformed relative to the origin of the group, however this caused confusion among many of our initial users. Each individual primitive is now translated according to the coordinate system of the canvas, regardless of whether it is in a group.

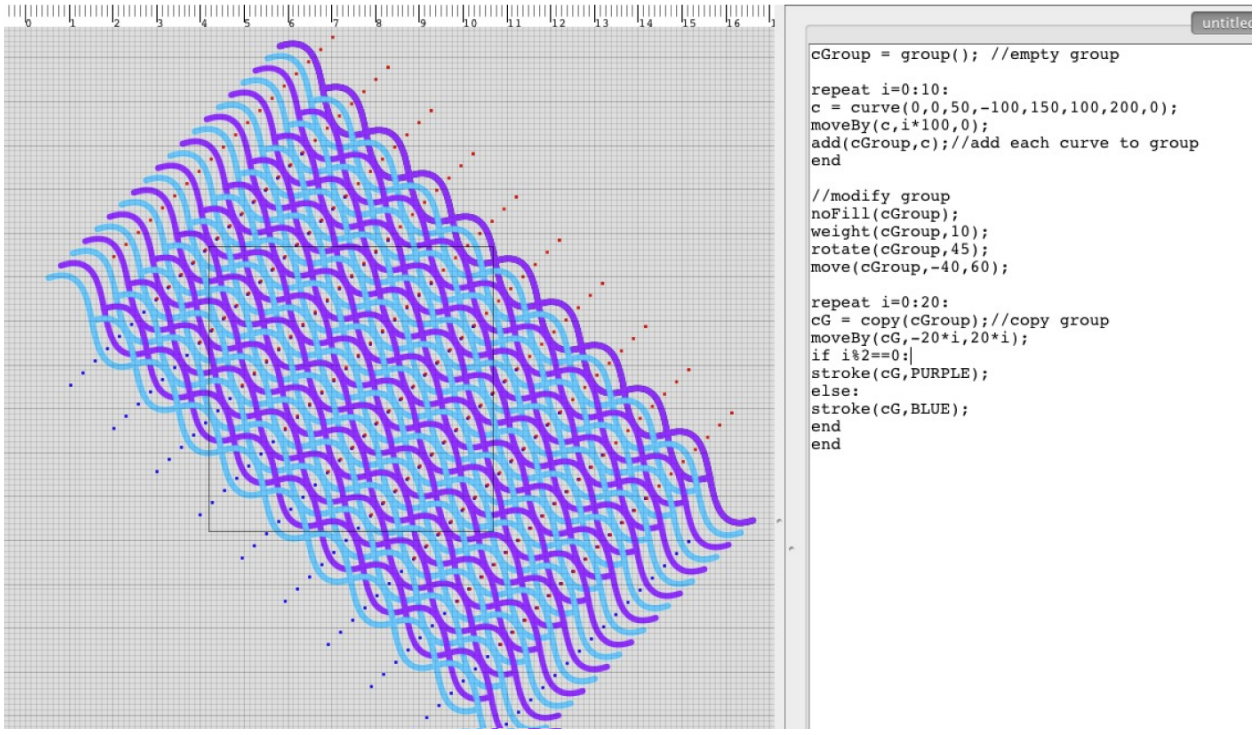


Figure B-6: Grouping with transformation and repeat statements

Polygon Booleans: DressCode also features a subset of transformation methods for polygon boolean operations (figure: B-7.) The union, difference, exclusive or and intersection methods accept two primitives as arguments and return the corresponding shape, or group of shapes, depending on the operation. In addition, the merge method performs a union on all objects within a group and returns a single primitive. The expand method converts an object's stroke to a filled polygon with dimensions that match the weight of the stroke. The boolean methods were added for multiple reasons. Polygon boolean operations play an important role in CAD design for fabrication, because they allow for the creation of vector paths that appropriate for subtractive manufacturing processes.

In particular, the merge method can function as a catch-all method for preparing designs for fabrication, and the expand method is useful for converting line drawings to paths that will retain their appearance when cut. In prior workshops, we relied on illustrator to prepare file paths for manufacturing, which was difficult, time-consuming and a hindrance to the computational design process. By integrating boolean operations as a part of the programming language in DressCode, it not only prevents users from having to rely on multiple software tools to design, but allows booleans to be used in a computational context, facilitating the creation of a whole new range of forms with simple primitives.

Both the syntax and functionality of the boolean operations have been modified significantly since the first version of DressCode. In early versions, the majority of the booleans were expressed through mathematical notation, rather than with built in functions (e.g. ellipse1 + ellipse2 would perform a union). After some of our early user testing however, we depreciated this syntax, because many testers felt that mathematical notation was not intuitive beyond the union and difference operations. In addition, the mathematical notation required that all booleans be set equal to an identifier (e.g. ellipse1 + ellipse2 = e3), in order to evaluate as a valid statement. The new method syntax still returns a resulting boolean, but can also be called without a corresponding assignment statement. Semantically, we originally had the boolean statements operate in a destructive manner as is the case in many graphic drawing programs; union(ellipse1, ellipse2) would destroy the original two ellipses and set the identifier ellipse1 equal to the resulting boolean. This was confusing however, because in a imperative program, the programmer could still reference the destroyed ellipse2. What we opted for instead, was a non-destructive boolean method wherein union(ellipse1,ellipse2) would return a third shape, which would automatically be drawn on the screen, and could be assigned to an identifier as needed. The original ellipse1 and ellipse2 would be hidden, but could both still be referenced without conflict, and could be revealed again on the canvas with the show() method.

Additional Methods: Aside from methods for shape generation and transformation, DressCode features a set of getter methods that return information about a shape primitive, including its location, origin point, width and height, as well as a method to determine the Euclidean distance between two objects. There are also a set of methods that enable more advanced mathematical operations, including trigonometric functions, mapping, rounding and random number generation. The random method posses an interesting questions for implantation, as it generates different numbers each time a a program is run. This can be a useful design feature, allowing for the generation of numerous patterns with random variations. Figure B-8 demonstrates variations of an algorithm that produces randomly symmetrical stripe patterns. It can also become problematic however, when the user wishes to keep the same randomly generated number for each interpretation of their program.

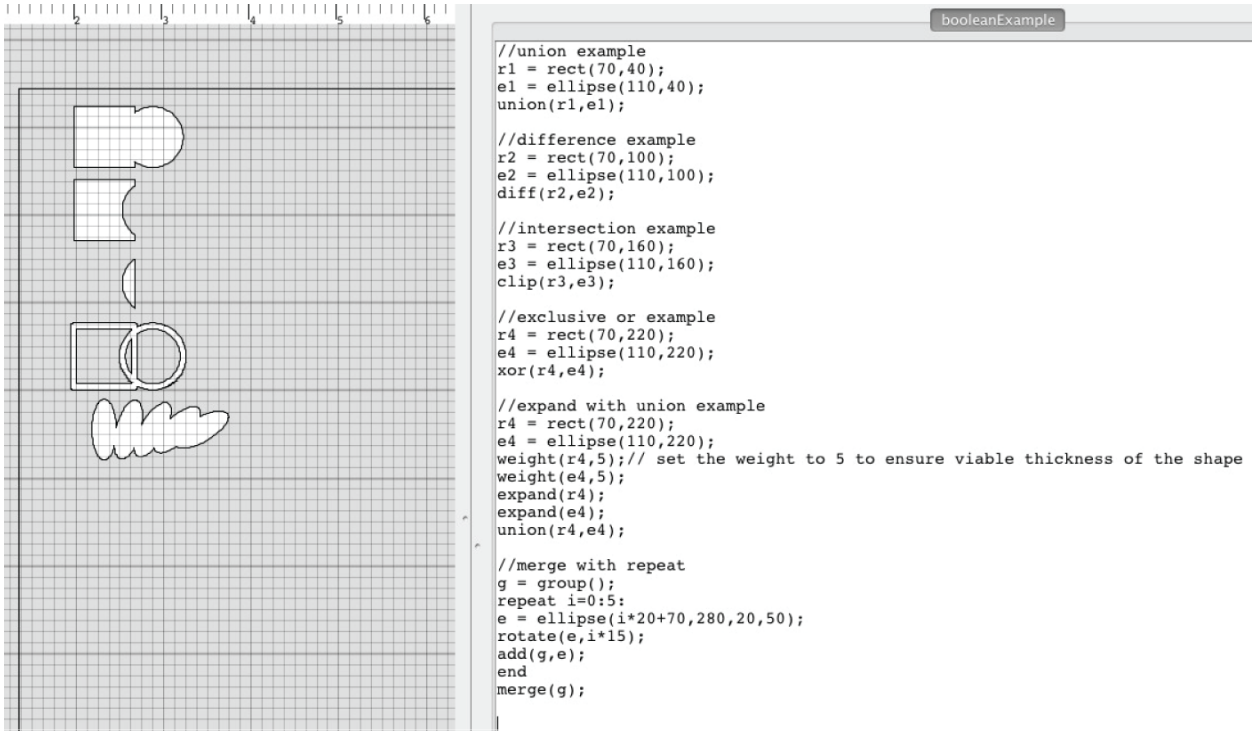


Figure B-7: Polygon boolean operations

Finally, we recently added a set of methods that allow the specification of numerical values in inches, centimeters and millimeters, or in the current units of the drawing board. These methods are helpful when sizing and transforming primitives to real world measurements rather than pixel values.

Constants: DressCode has a small set of constants to assist with design. WIDTH and HEIGHT correspond to the width and height of the current drawing board. PI corresponds to the numerical Pi value. Lastly, there are a set of color constants; RED, GREEN, BLUE, PURPLE, PINK, etc., that can be used to define the color of objects with fill and stroke methods. Custom colors can also be specified with 0-255 RGB values.

Figure B-8: Stripe pattern variations using the DressCode random method

Bibliography

[1] How to make a laser cut lamp. `http://www.instructables.com/id/How-to-make-a-laser-cut-lamp`, November 2007. (Accessed: 05/25/2013).

[2] About — iris van herpen. `http://www.irisvanherpen.com/about#collections`, 2013. (Accessed: 05/29/2013).

[3] M. De Berg. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin,Germany, 2010. Third Edition.

[4] G. H. Hardy. A mathematician's apology. 1940.

[5] Resnick M. and Silverman B. Some reflections on designing construction kits for kids. In *Proceedings of the 2005 Conference on Interaction Design and Children*, 2005.

[6] LUST Reas N., McWilliams C. *Form and Code: In Design, Art and Architecture, A Guide to Computational Aesthetics*. Princeton Architectural Press, 2010.