ABSTRACT

The combination of computational design and digital fabrication offers many exciting possibilities for art, design, and creative expression. We seek to make computational design accessible by developing tools that allow novices to use programming and digital fabrication to produce personal and functional objects. In this paper, we describe our development of Codeable Objects, a preliminary computational-design programming tool developed to work in conjunction with digital-fabrication machines. We also present our evaluation of the tool based on a set of user studies in which people built computationally generated crafts, clothing, and accessories. These studies illuminated the viability (and challenges) of engaging novice programmers through design and digital fabrication, and provide a platform for future work in developing programming tools to support personal expression.

Author Keywords
Software; Accessibility; Art and Craft; Computational Design.

ACM Classification Keywords
D.2.2 Design Tools and Techniques; J.6 Computer-aided engineering (computer aided design).

INTRODUCTION

The increase in accessibility, diversity, and functionality of modern computer systems has made programming useful in nearly all realms of human study and advancement. The development of digital-fabrication technology has further increased this utility by providing more immediate transitions between the digital and physical realms. Art and design are two domains that offer exciting possibilities when combined with programming and digital fabrication. Unfortunately, use of programming as a medium for art and design, especially by young adults and amateurs, is limited. Many individuals consider computer programming to be a highly specialized, difficult, and inaccessible activity that only has relevance as a career path rather than as a mode of personal expression [12]. Despite this perception, programming has the potential to correspond well with traditional, physical art-making practices [7]. By finding ways to connect computation to the design and production of personally relevant physical objects, it is possible to engage novice practitioners in creative programming. The combination of digital-fabrication technologies with computational design serves as one such connection.

Computational design is the practice of using programming to create and modify form, structure, and ornamentation. We distinguish computational design from other realms of programming in that decisions in computational design primarily determine the aesthetics of an object rather than its behavior. Computational design offers a number of benefits that can extend traditional design techniques. These benefits include:

• Precision and automation: Computation affords high levels of precision and allows for automation of repetitive tasks, enabling the rapid development and transformation of complex patterns and structures.

• Generativity and randomness: Computation allows for the programmer to design algorithms which when run, allow for the computer to autonomously produce unique and often unexpected designs.

• Parameterization: Computation allows users to specify a set of degrees of freedom and constraints of a model and then adjust the values of the degrees of freedom while maintaining the constraints of the original model [10].

The conjunction of computational design and digital fabrication has the potential to allow individuals to use programming to express their aesthetic concerns in the creation of objects. Fabrication devices can work with a wide range of materials and support traditional construction practices and interests. In addition, fabrication machines are rapidly decreasing in price and increasing in availability [6]. These machines can extend traditional forms of fabrication by allowing for new levels of automated complexity, precision, and fabrication speed in the production of one-off physical objects [4]. Yet, despite the opportunities afforded by merging these two technologies, the combined practice of computational design and digital

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright © 2013 ACM 978-1-4503-1899-0/13/04...$15.00.
fabrication is primarily limited to the professional sector at this time. To begin to address this limitation, we have developed a prototype programming tool entitled Codeable Objects, created to aid novice users in the combined use of computational design and digital fabrication. In this paper, we discuss the format of Codeable Objects and the insights gained through a set of workshops where people used our tool in conjunction with digital-fabrication machines and traditional crafting methods to produce personally meaningful artifacts.

RELATED WORK
Three primary categories of existing software tools directly relate to our study of computational design and digital fabrication for novices: professional computational-design tools, novice-oriented programming environments, and novice-oriented computer-aided-design (CAD) tools. While certain qualities are shared between these categories, several key distinctions exist between each group.

Professional Computational-Design Tools
A couple of forms of professional computational-design tools exist. Foremost, many popular graphic-user-interface (GUI) CAD applications include a feature that allows the user to automate certain elements of the program through scripting or 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. This scripting is usually omitted from the primary menus and interfaces of the applications that feature it.

There are also professional tools that are explicitly developed for computational design. The most prominent example is Grasshopper, a third-party add-on for the Rhinoceros 3D modeling tool. Grasshopper is a data-flow programming environment that lets users combine multiple C# scripts with a variety of other modules to create and adjust 3D models in Rhino.

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

Novice Oriented CAD tools
Some developers have recognized the benefits of helping novices to participate in CAD and have created more accessible CAD environments. TinkerCad is an online CAD environment for 3D modeling that provides a simplified set of methods for manipulating and combining basic 3D primitives. TinkerCad allows users to export their designs to the .stl format and is compatible with 3D printing [15]. kidCad is another CAD environment designed for 3D printing and allows users to design 2.5D forms in a process similar to imprinting objects in clay [5].

Other novice-oriented CAD tools use the more intuitive process of sketching as a means of design. Sketch It, Make It is a 2D CAD tool that allows users to constrain their designs through gestures made using a digital drawing tablet [14]. 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 [17].

There are also examples of domain-specific novice-oriented CAD tools. SketchChair allows users to design their own chair by sketching with a computer stylus [13]. 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 [8].

Novice-Oriented Programming Environments
In addition to beginner CAD tools, a number of tools and applications have been created to introduce inexperienced programmers to the realm of computer science.

Logo, a computational drawing program, serves as the seminal novice programming language founded on principles of constructionism and embodiment [9]. The Scratch visual programming language is a notable successor to Logo, and allows users to create interactive projects by combining command blocks rather than writing textual code [11]. Alice is another programming environment that relies on visual programming, but is targeted towards an older user group than Scratch [3]. Turtle Art [16] and Design Blocks [2] are two visual programming languages inspired by Logo that are designed specifically for visual composition. Processing is a text-based programming environment designed for easy learning, and directed toward artists, designers, and inexperienced programmers [10].

Logo, Turtle Art, Design Blocks, and Processing facilitate computational drawing and, therefore, can be viewed as computational-design environments. There remains a gap, however, between novice-oriented programming environments and the novice-oriented CAD tools. With the exception of FlatCAD, none of the examples listed address the space between computational design and CAD for digital fabrication. Codeable Objects was developed to examine strategies to address this space by supporting novice programmers in a simple, general computational-
design environment that allows them to work closely and effectively with digital-fabrication tools. Furthermore, Codeable Objects is designed to support aesthetic design choices in addition to functional and structural design, distinguishing it from FlatCAD. This aesthetic orientation is highlighted by our user studies, in which we explored novel application domains, most notably fashion.

**Requirements for Computational-Design Tools for Novices**

**Essential Qualities and Evaluation Criteria**

Based on prior work in combining design and art with digital technology [1], we hypothesized that the following functional properties are necessary for an effective novice computational-design and digital-fabrication tool:

- Novice-oriented programing syntax that contains a simplified and limited set of programing methods.
- Prioritization of visualization and simulation in order to provide users with ample visual feedback to inform their programming decisions.
- Simple workflow from software to fabrication to facilitate the transition from programming to physical output.
- Free and open-source software tools to afford high levels of access to casual users.

In addition to integrating these properties, we generated a set of evaluation criteria for any prospective computational-design toolset for novices. We believe a successful tool should produce the following results:

- Allow users to successfully create physical artifacts.
- Enable the production of durable and useable artifacts.
- Afford a wide degree of variation in design and expression.
- Enable people to understand the functionality and utility of the programs they write.
- Allow users to create objects and designs they would have difficulty generating with conventional techniques.
- Engender in users a positive, enjoyable experience.
- Foster a sense of confidence.

The remainder of this paper describes our development of a prototype computational-design tool for novices containing the functional properties we describe. We then discuss the evaluation of this tool using the above criteria.

**The Codeable Objects Programing Library**

Codeable Objects is a programming environment that was developed as a Java-based library for Processing. We chose to build Codeable Objects on top of the Processing environment because it enabled a rapid cycle of development, user testing, and re-design.

**Library Structure**

The 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. To use the library, a user imports it into the Processing environment and then writes and compiles code using the Processing editor. Codeable Objects allows users to define and manipulate basic geometric primitives such as Points, Lines, Curves and Polygons. These primitives can then be collected within Pattern and Shape objects—structures designed to capture surface decoration and 2D structure, respectively—to form increasingly complex designs.

![Figure 1: Codeable Objects Workflow](image)

Codeable 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 programing approach. The structure of Codeable Objects enables users to simultaneously apply transformations to all of the elements in a collection that make up a complex pattern or shape. It is also possible to import scalar vector graphics files (SVGs) to incorporate pre-drawn designs as elements within a pattern or as a container for existing patterns.

Users are presented with a 2D preview of their designs when they compile their code. Codeable 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 Codeable Objects can be fabricated on essentially any x-y axis tool. 3D structures can be created by assembling
fabricated pieces. Figure 1 demonstrates the workflow from code to a finished object.

The Codeable 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. Finally, the library contains several domain-constrained examples with more sophisticated preview features that directly support the generation of decorated 3D objects from flat surfaces. For example, for the lamp design workshop we discuss below, Codeable Objects automatically generated press-fit notches to allow laser-cut wooden pieces to be fitted together.

This preliminary version of Codeable Objects was designed to iteratively explore the creative potential of computational design and digital fabrication. This version has allowed us to explore a novel and tremendously promising area; it enabled us to conduct the user studies we discuss in the remainder of the paper. These experiences, in turn, provided us with valuable insight into the features that a complete programming environment specifically designed to enable novices to design and build physical objects should include, and illuminated the unique affective, intellectual, and expressive potential of computational design.

USER STUDIES
We evaluated Codeable Objects through one preliminary study and one long-term evaluation. The preliminary study consisted of a one-day lamp workshop in which participants designed and built lamps. The long-term evaluation was conducted through a two-week fashion workshop in which participants created garments and accessories.

Methodology
The workshops were evaluated through before and after surveys, interviews, and photographs of participant projects. The surveys were aimed at understanding participants’ previous experience in programming and design, their interest in and attitudes toward programming and design (before and after the workshops), and their engagement in and enjoyment of the workshops.

Pre-workshop surveys focused on participants’ previous experience and attitudes. They also asked students to describe their opinions about how programming and craft could be combined, and how they felt programming could extend or limit creativity. Post-workshop surveys 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.

In-person interviews were conducted with the participants in the fashion 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 garments. They were asked to describe what they enjoyed, what was difficult for them, and what they felt they had learned through this process.

Survey results, verbal-interview responses and project outcomes were then analyzed to determine if the essential qualities outlined in the requirements section (above) were achieved. We also used this information to identify recurring and prominent themes in participants’ experiences.

Preliminary Study: Computationally Designed Lamps
The first evaluation of Codeable Objects was conducted with a group of nine graduate students who engaged in a six-hour workshop. According to self-reported pre-workshop survey data, all but one of the participants were intermediate to experienced programmers.1

During the workshop, participants engaged in the design and fabrication of a laser-cut lamp through the use of a set of domain-constrained examples from the Codeable Objects library. By altering the code in the examples, or writing their own code with the library methods, users could generate a wide variety of lamp forms and patterns. When compiled, the lamp examples displayed a specialized preview containing a 3D model of a user’s design, visualizations of the 2D tool paths that would be used to cut the parts on the laser cutter, and a rendering of the current lampshade decorations. After using the library to design their lamps, participants were provided with access to materials and training on the laser cutter in order to create their finished pieces.

---

1 Because of their prior expertise, the responses of the participants in the
Lamp Workshop Results
Users in the lamp workshop experimented extensively with the library to produce a wide range of forms and patterns for their lamps. Although the physical assembly process required additional time for some participants, all participants indicated on the survey that they were able to complete a finished product to their satisfaction. The physical objects produced were 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 (Fig 2).

Long-Term Evaluation: Computational Fashion
The long-term evaluation 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\(^2\) and used the full version of the Codeable Objects software. Participants were given 10 days to conceptualize and construct a garment using a combination of computational design, digital fabrication, and traditional sewing and crafting. The second study was more open than the first; 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 Codeable 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 (Fig 3).

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 Codeable Objects and generating their own programs from scratch. They used these programs to create a design for a wooden bracelet (Fig. 3), which was then laser cut and assembled. After these two initiation activities, these participants were asked to conceive their own garments and provided with the resources to design, prototype, and craft finished garments.

Fashion Workshop Results
Participants in the fashion workshop were successful in using programing 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 programing methods to produce a diverse set of final products (Fig 4). 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.

On the post survey, when asked if they “were able to complete a finished project to their satisfaction,” 100% of the fashion participants responded yes. 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 fashion workshop indicated their interest in crafting increased after the workshop, as did their enjoyment of the design process. Eighty-eight percent of the participants in the fashion workshop indicated that they felt more comfortable programing after the workshop than before.

---

\(^{2}\) See [http://nuvastudio.org/](http://nuvastudio.org/)

**Figure 3: Resultant bracelets and scarves**

**ANALYSIS AND DISCUSSION**
The analysis of the outcomes in conjunction with the experiences and feedback of the workshop participants revealed several interesting patterns. We discovered that the
A combination of computational design and fabrication can actively support the expression of personal identity in a positive setting. It can also foster feelings of confidence in programming and support aesthetic and technological literacy. The projects from the workshops demonstrate unique aesthetics and suggest new opportunities for casual and novice practitioners of art, craft, and design. The workshops also 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 workshops promoted a sustained engagement in programming. The evidence of these outcomes and their implications are discussed at length in the following sections.

Self Identification as a Designer and Programmer
Before the start of the fashion workshop, the participants’ knowledge of potential programming applications was vague. When asked in the pre-workshop surveys how they thought programming 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 J]

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]

Surprisingly, the more experienced programmers in the lamp workshop also exhibited limited knowledge of computational design prior to the start of the workshop. They considered programming to be a 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]

Figure 4: Completed Garments (Clockwise from top left: samurai dress, octagon dress, flag pants, viral sweatshirt)
Following both workshops, participants demonstrated improved understanding of the benefits of using programming as a tool for physical construction:

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 post 2]

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]

Participants in the lamp workshop also described in the post-workshop survey 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 post 4]

I love the idea of being able to combine my interest in programming for creative expressions. [Lamp participant post 6]

The participants in the fashion workshop expressed dramatically different views on programming after the completion of their final projects. 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 novice programmers in the fashion workshop demonstrated a new understanding of their ability as a programmer. From the interviews:

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 inaccessible and irrelevant. The overall feelings of engagement and empowerment fostered by these experiences indicate that computational-design tools for novices could serve as a powerful way to positively change people’s understanding of the relevance and applications of programming, while fostering technological and aesthetic literacy in the process.

**Computational Complexity from Simple Tools**

Computational design can provide numerous potential benefits to the realm of conventional art and design. Through the use of Codeable Objects, participants in the workshops developed and implemented techniques that took advantage of these benefits. The participants in the fashion workshop, in particular, used design approaches that effectively leveraged a variety of the advantages of computational design. 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. Lastly, the viral shirt, demonstrated an optimal 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.

Aside from conceiving of and applying computational-design approaches, participants consistently articulated an awareness of the rationale behind the methods they were using. One of the participants in 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 I]

Participants also demonstrated an understanding of specific programing 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 implement computational approaches supports the growth of technical literacy as described in the preceding section. Beyond this literacy, the ability of novice programmers to inventively and effectively combine personal craft with diverse computational approaches suggests the potential for the growth of exciting new set of aesthetics and techniques in both fields. The breadth of aesthetics in the products produced by these two initial workshops hints at the creative potential offered by continued work in this domain.

Tensions in Computational Design
Parallel to their effective use of programing, participants also put forth detailed critiques of the programing process, which brought into focus concerns about the practice of computational design itself. One participant articulated the absence of an intuitive design practice in programing by explaining the difference between the creative processes of writing a piece of music versus programing a visual design. He described his musical process as one in which he begins with a rough melody, and then gradually adds to that element until a more complete and cohesive theme results.

Usually [the music] changes because usually I just have an idea of some sort of note thing, like really simple and then from there I kind of in the moment think of what kind of loops I want to make, trying to make a theme for myself.

Programing a design, he said, was less intuitive:

For code I usually have some end thing that I want, like a design, you think of what you want it to look like, and then from there I try to think of parameters you would try to set up to produce a kind of outcome that would be similar. These two processes feel pretty different to me... For code, I usually think about what I want in the very end and how to get to that, but for music I think about it as I go. [Fashion participant O]

Procedural intuition is an important part of the creative practice. There remains a continued debate about the effect the structural restrictions of programing have on the process of creative intuition. The statement from the participant in the fashion workshop highlights the essence of this critique in the context of his own experiences. Contrasting critiques emerged from participants in the lamp workshop. One participant reacted against classifying generative processes 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." Programing, 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 post 6]

This second comment addresses the concern that 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 designer’s role in the deciding process is often ambiguous. The fact that this comment and the one preceding it are in conflict with one another is particularly revealing because they highlight a debate about the role of conscious design and the restriction of intuitive creativity. User insights like these demonstrate that novice-oriented tools in the right setting have the potential to promote a critical discussion of the overall practice. This critical discussion demonstrates that participants were not only engaged in the activity and learning about the technical functionality of the tools they were using, but also developed a deep understanding of the creative implications of computational design.

The Programing-Prototyping Relationship
The duration of the fashion workshop allowed for an examination of role of prototyping in the design process 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 many 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 programing realm.

In the case of the octagon dress, (Fig. 4), 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. For the participants in the workshop, direct physical engagement with the materials became an important input to the decisions they made in their programs. The majority of the participants indicated that the physical prototyping process was both important and enjoyable:

*I thought it was fun coming up with a bunch of ideas that were completely different and then just picking one out.* [Fashion Participant M]

*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 post 6]

The central role of prototyping in the creative process strongly suggests that future computational-design tools for novices should endeavor to support iterative prototyping. In particular, features that can intelligently respond to the limitations and affordances of specific physical materials and crafting techniques could better inform the design process of the user.

**Desire to Continue Making**

One major indicator of success is willingness to further engage in an activity. Participants from the fashion workshop talked extensively about what they would like to make in future computational-design workshops, 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.” All but one of the 17 participants indicated on the post-workshop survey that they would like to continue combining craft and fashion after the workshop. In an interview one fashion participant stated:

*I would want to program again after this workshop. I’d want to make more designs.* [Fashion participant M]

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]

These responses indicate that these methods can produce independent involvement and enthusiasm for computational design. They also show that computational design affords a means to decorate personal belongings. These factors have the potential to promote sustained participation in programming as a method of personal creative construction.

**LIMITATIONS AND FUTURE DIRECTIONS**

While the workshops highlighted many positive affordances of computational design, we also encountered challenges.

**Syntactic Challenges**

Participants in the fashion workshop struggled with 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 sense 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]

Writing code for the first time is a difficult task; however, the high levels of frustration registered by the participants suggest that the further simplification of the syntax of the language may be necessary for a novice-friendly tool. The success of visual programing languages like Scratch suggest a compelling direction for future software.

**Real-Time Feedback**

Another consistent frustration experienced by participants of both workshops was the lack of real-time feedback. The lamp workshop participants in particular were frustrated by the necessity of continually having to compile their programs after making changes in order to see the results of their code edits:

*It was frustrating in the Processing code to have to update parameters and reload the program every time. It would be nice to see in real-time how changing these parameters affects the output.* [Lamp participant post 2]

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. Our reliance on the Processing programing environment made the incorporation of real-time compilation infeasible; however, the feedback from the workshop participants indicates that this would be a worthwhile avenue to explore because it would allow users to receive instant visual feedback from changes in their programming. In addition, a programing system with real-time feedback may 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.

**Using Digital-Fabrication Tools**

Participants in both workshops were enthusiastic about the opportunity to use digital-fabrication tools, like the laser cutter:

*The laser cutter*...*is something that I think has really in the last couple of years went from being something that no one could afford to something that’s starting to pop up in more accessible places and it’s really exciting that it’s becoming more accessible to everyone.*

[Fashion participant S]

This enthusiasm was often tempered by wait times because there was only one laser cutter available to a large group of users. The fabrication process for the lamp workshop was especially time consuming for a day-long workshop. It would be productive to explore other methods of fabrication that allow for more than one device to be available to users. We are experimenting with using vinyl cutters and ink-jet printers as affordable output mechanisms for the software with promising results. An ideal environment for supporting computational design should contain different fabrication machines with a range of speeds. The workflow for creating and uploading digital files to these machines should be as consistent as possible. Traditional fabrication tools like sewing machines, hot-glue guns, scissors, and paper cutters should be made available and introduced with the digital-fabrication process.

**CONCLUSION**

Computational design and digital fabrication offer many compelling opportunities for personal creative expression through programming. Codeable Objects provides one preliminary format to support novice programmers in this practice. Participants in workshops who used Codeable Objects were able to create personal items that were beautiful, functional, and personally meaningful by writing their own programming code. The participants emerged with an awareness of the field of computational design, specific understanding of some of the primary elements of programming, and a desire to continue using programming to build their own objects. Greater engagement in this process can be produced by building better tools that minimize the challenges of programming syntax while supporting the physical prototyping process. The continued exploration of this field has the potential to expand the creative applications of programming and support emerging programmers in new forms of art and design.

**REFERENCES**