DressCode: Tools and Activities to Engage Youth in Algorithmic Craft

1st Author Name
Affiliation
Address
e-mail address
Optional phone number

2nd Author Name
Affiliation
Address
e-mail address
Optional phone number

3rd Author Name
Affiliation
Address
e-mail address
Optional phone number

ABSTRACT

Author Keywords
Guides; instructions; author's kit; conference publications; keywords should be separated by a semi-colon. Mandatory section to be included in your final version.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

See: http://www.acm.org/about/class/1998/ for more information and the full list of ACM classifiers and descriptors. Mandatory section to be included in your final version. On the submission page only the classifiers' letter-number combination will need to be entered.

INTRODUCTION

Computation is a driving force in our world. The recognition of the importance of computation to both national and international industry and advancement has resulted in the STEM Education Initiative, a national drive to engage young people in topics of math, science, engineering, and computer science [2]. Proposed K-12 computer science educational curriculums are aimed at teaching students the core components of computation and programing, as well as fostering sustained interest in a broad range of computer science professions [1]. These approaches explore digital applications of computer science that will appeal to young people, including game development, digital media manipulation, and web development. Parallel to the growth of youth STEM initiatives is the recognition of the educational potential of the Maker movement, which has produced a growing number of youth-oriented maker initiatives. These programs seek to create opportunities for young people to participate in hands-on learning through making [4], and conduct research on how hands-on-learning can be integrated with educational policy and practice [3]. Many youth maker initiatives are designed to connect directly to STEM educational objectives, by connecting making with principles in science, technology and math; however these initiatives also have great potential to connect with the arts. As researchers, we believe computation can serve as a powerful tool for creative disciplines. When applied to the arts, programming offers different paradigms for creation and new methods of problem solving. We see an opportunity to support young people in art-oriented programing by connecting computation to making. Specifically, we seek to support youth in the combined use of computational design and traditional arts and crafts for the production of functional and decorative physical artifacts. We use term *algorithmic craft* as a way of describing this collective practices.

Combining computational design and handcraft raises many interesting questions: What are the important design principles to consider when creating programming environments for physical design? How do we compellingly link textual code with visual designs, and what are the appropriate intersection points between textual manipulation and visual manipulation? What support is required to help people move back and forth from programming to building real objects in a way that is comfortable, expressive, and pleasurable? How can we remove the technical challenges involved in translating code into an object that can be successfully fabricated, but still support 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?

To explore these questions we developed a software tool called DressCode, to enable young novice programers to participate in computational design. We designed and implementation of a set of algorithmic craft activities in which high-school students used DressCode in conjunction with traditional forms of hand-craft to produce personal artifacts. Throughout our description and analysis of these experiences, we discuss design factors that enable young people to use computation in ways that emphasize pleasure, intellectual engagement, and utility in the service of personal expression.

ALGORITHMIC CRAFT

In order to illustrate the creative potential of algorithmic craft, it is useful to examine the key affordances of computational design and craft and how they can be combined into a single practice. In computational design, the abstract qualities of computer programming provide a powerful way of thinking about design. Similarly, computational design can be applied to any design domain, be it physical, visual, sonic, or interactive. Our focus focus in computational design is the process of using computational processes to create visual forms and patterns. Although individual applications of computational design often consider material properties that are relevant to a specific domain, computational design does not possess 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.

Affordances of Computational Design

- **Precision:** Computation supports high levels of numerical precision.
- **Visual Complexity:** Computational design allows for 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.
- **Generativity and Self-Similar Forms** Computation allows for the programmer to create algorithms that allow for the computer to autonomously produce unique and unexpected designs.
- **Parameterization:** Computation allows users to specify a set of degrees of freedom and constraints of a model and then adjust the values of the degrees of freedom while maintaining the constraints of the original model.
- **Documentation and remixing:** Computationally generated designs are generated by a program which can be modified by other designers, and serve as a form of documentation of the design process.

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.
- **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 the area of deviation. Because computational design is governed by a systematized ruleset, the methods of breaking these rules at arbitrary points are often unclear or tedious to implement.
- **Selecting a final design:** 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 difficult, especially if the decision is based on aesthetic criteria.

Affordances of Craft

The cultural connotations of craft have varied throughout history, however forms of craft have endured, both as a recreational pursuit, and as set of valued artisanal practices. Below we describe the specific aspects that are integral to algorithmic craft.

- **Materiality:** Craft involves the manipulation of physical materials by hand, which is often an intuitive physical process. The decisions made in the craft process are altered by the feel of working with the material.
- **Pleasure:** Fundamental to traditional conceptions of arts and crafts is the idea of pleasure in working with one's hands [5]. This emphasis on pleasure is retained in conceptions of professional and amateur craft practices today.
- **Unification of form and function:** Although not all craft is functional, many forms of traditional craft can applied the creation of useful objects. In addition craft often emphasizes the importance of beauty in the form and ornamentation of objects. Well crafted artifacts frequently demonstrate the successful unification of aesthetics and utility.
- **Craftsmanship:** Put in a description of craftsmanship

The potential of Algorithmic Craft

The combination of computational design and craft offers the opportunity to produce pieces that are connected to a distinct space, time and process, and that are shaped by material properties, while incorporating patterns and forms made possible through computational processes. 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. Combining computational design and craft also offers the opportunity to engage people with interest in craft in the use of computational tools.

Digital to Physical Translation

In order to connect the digital artifacts of computational design with the physical world of craft, some form of transition is needed between the digital and the physical. Digital fabrication technology provides one method of connecting the abstraction of computational design and the material domain of craft by converting digital designs to physical forms. Digital fabrication technology ranges from the additive processes of 3D printing, to the subtractive manufacturing techniques of laser cutting, CNC mlling and vinyl cutters. Subtractive fabrication technology often corresponds well with craft processes Unlike most 3D printing technology, subtractive processes can work with a wide range of materials [**?**].

Time is an important quality of crafting. Handcrafted artifacts often require a great deal of time to complete and involve repetitive tasks. Because the process of crafting is often pleasurable and contemplative, many people look forward to spending time productively engaged with their hands. In algorithmic craft, the role of digital fabrication is not to speed up craft, but to facilitate the translation of computational forms to physical materials. These physical forms, in turn, can be shaped through craft processes, and imbue computational designs with individuality, utility, and temporality. Several forms of digital fabrication machines are capable of producing forms that are suitable for both expert and novice levels of handcraft. 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 also can be used on cloth and paper. They 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 faster rate than 3D printers or milling machines, and permit a higher tolerance for error during the craft process because it is feasible to re-cut damaged parts. Inkjet printers can also function as a form of digital fabrication, as they can print on cloth, or transparencies used in screen printing...

Challenges in access and application

primary barriers being: perceptual: people don't know you can use programing for physical making and craft intellectual: learning the unique affordances of computation and applying them in personally meaningful ways technical- challenge of using programing for design Focused on software design for the technical, activity design for the perceptual

RELATED SOFTWARE TOOLS

Professional Tools

Entry-level Programming Environments

Scratch - variants (turtle art, design blocks)
Processing

Entry-level CAD Tools

Sketchup
Tinkercad
Autodesk 123D
Vector Graphics programs (Illustrator, Inkscape)

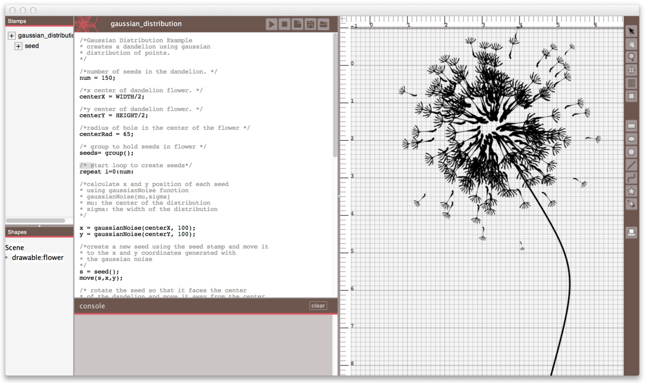
Novel accessible Fabrication and CAD tools

Sketchit Make it
Spirogirator
Codeable Objects

Visual feedback is important. Combining two tools is challenging for novices. Using processing in a fabrication context is challenging.

DRESSCODE SOFTWARE DESIGN

DressCode is a 2D computational design tool aimed at young programers. The primary objectives in the design of DressCode were to reduce the technical challenges of computational design, to foster independent and deliberate design decisions, and to assist in the creation of designs that were viable for craft applications. Although the name of the tool reflects an emphasis on wearables and fashion, as an application DressCode is general purpose, and can be applied to many forms of algorithmic craft. The software is comprised of the following elements: A custom programing language and drawing application programing interface (API), an integrated development environment, and a graphic user interface design tool.



```
/*Gaussian Distribution Example
* creates a dandelion using gaussian
* distribution of points.
*/

/*number of seeds in the dandelion. */
num = 150;

/*x center of dandelion flower. */
centerX = WIDTH/2;

/*y center of dandelion flower. */
centerY = HEIGHT/2;

/*radius of hole in the center of the flower */
centerRad = 65;

/* group to hold seeds in flower */
seeds= group();

/* start loop to create seeds*/
repeat i=0:num:

/*calculate x and y position of each seed
* using gaussianNoise function
* gaussianNoise(mu,sigma)
* mu: the center of the distribution
* sigma: the width of the distribution
*/

x = gaussianNoise(centerX, 100);
y = gaussianNoise(centerY, 100);

/*create a new seed using the seed stamp and move it
* to the x and y coordinates generated with
* the gaussian noise
*/
s = seed();
move(s,x,y);

/* rotate the seed so that it faces the center
* of the dandelion and move it away from the center
```

Figure 1. The DressCode software

Interface

The interface of DressCode is divided into two sections: a design panel on the left and a coding panel on the right. The design panel approximates many of the features of a digital graphic design tool, with rulers, a grid, a drawing board, and a graphic tool menu. The drawing board defines 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 by selecting the drawing board icon in the tool menu. Similarly, the pan and zoom tools allow the user to navigate around the drawing board. The print tool opens a dialog that allows the user to export their current design in a vector format for output through digital fabrication.

The coding panel contains a primary window for entering text, and an output console for print output and error reporting. It also contains two additional panels, a declarative listing view, and the Stamp menu. Both of these panels are discussed in detail in the Graphic Tools section. A toolbar on top of the coding panel allows the user to save their existing program, open a program, or run their current program. When a program is run, the resulting design is displayed in the design panel. The DressCode programming language is interpreted with semantic functionality that is simulated through a Java-based library. For most programs, the interpretation process is instantaneous; however, some programs with complex operations require several seconds to be executed. Early in the development process, we experimented with automatic interpretation, wherein the design would be automatically updated as changes were made to the code. Early user testing demonstrated that this process produced a great deal of frustration, and people explicitly requested control over running their program and updating their design.

Programming Language and Drawing API

DressCode features a custom textual programing language with native 2D drawing functionality[1]. The language supports conventional datatypes including numbers, strings, booleans, and lists, and also supports drawing primitive datatypes such as ellipses, rectangles and polygons, as well as groups of primitives. The language also contains support for basic expressions, as well as conditionals, loops and user-defined functions. DressCode is dynamically typed in order to reduce syntactic challenges for people new to textual programing. DressCode determines how to treat the data assigned to an identifier based on context and identifiers can be assigned to datatypes that differ from their original assignment at any point.

The DressCode drawing API is s formulated on an Object Oriented Programming (OOP) paradigm which enables users to create and manipulate collections of geometric primitives including points, lines, curves, polygons, ellipses, rectangles and imported SVG primitives. Shape primitives are initialized by calling the appropriate primitive method, and passing it a set of parameters designating the initial appearance of the primitive. For all primitives besides lines and curves, the first two parameters designate the coordinates of origin point of the primitive, and each primitive is drawn in the design panel relative to this central origin point. There is no "draw" method for DressCode. Instead all primitives are automatically drawn in the order of their initialization. This was designed to make it as easy and immediate as possible to have a design appear on the screen. Objects can be selectively hidden with the hide method.

All primitives in DressCode can be modified through two kinds transformation methods, Geometric and Stylistic. Geometric transformations allow for primitives to be rotated, scaled or moved or combined with other primitives. All geometric transformations are performed relative to the origin of the primitive, unless otherwise specified. Rotation statements for example have an optional third parameter which specifies a point to rotate the primitive around. Stylistic transformations affect the appearance of a primitive, and can modify properties like fill and stroke color, stroke weight, and presence or absence of fill and stroke. Transformations on a primitive are performed by assigning an identifier to the primitive, and then calling the transformation method with the identifier. By using the transformation methods to manipulate primitives in a structured manner, it is possible to generate complex and interesting designs from simple forms. Through these methods, DressCode was intended to support the affordances of computational design, specifically precision, visual complexity, generativity and stylistic abstraction, in a way that was feasible for new practitioners.

As mentioned, DressCode also contains a set of transformation methods that allow primitive primitives to be combined with one-another via polygon boolean operations. Polygon

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

booleans are widely used in CAD applications and are essential for many forms of digital fabrication because they allow for complex configurations of primitives to be combined into effective toolpaths. DressCode contains methods for performing unions, intersections, differences and either-or intersections between any two or more primitives. They can also expand strokes to filled polygon paths, enabling the translation of line art to a form that will maintain its appearance when fabricated. By positioning these operations as primary components of the DressCode, we attempted to create a programming language that produced forms that were applicable to fabrication as a direct result of the design process.

As a method of organizing sets of primitives, DressCode contains a group datatype, which essentially is a specialized list for primitives. Groups can be initialized with a starting set of child primitives, or they can be initialized as empty, and then later have primitives added (or removed), from them. Groups automatically maintain an origin that is the average of all their children's origins. Geometric transformations performed on groups will relatively effect all of the primitives within the group; for example, if a group of polygons is moved to 0,0, then the origin of the group will become 0,0, with all polygons in the group being drawn relative to that new origin point. Groups can be multi-dimensional, and contain other groups. Stylistic transformations applied to groups will also be applied to each child. Groups also facilitate more advanced transformations. For example, it is possible to clip a group of primitives within the bounds of a single primitive to serve as a form of clipping mask, and a union can automatically be performed on all objects in a group with the merge method.

Random and Noise Methods, Property Methods, Unit Methods.

We chose to create a textual programing language for DressCode because we believe textual programing is well suited to transparent representations of computational design algorithms. In recognition of the challenges of first-time text programming, we also focused on creating intuitive forms of interaction and feedback to correspond with the DressCode language. These are inherent in the graphic drawing tools, which we discuss in the next section.

Graphic Tools

The drawing and manipulation tools in DressCode are designed to allow users to create and modify elements of their design through graphic selection. Collectively, the drawing functionality of the tools is similar to the functionality of many existing forms of 2D vector graphics software. The tools are distinguished from other graphics software tools because they maintain a direct symmetry with the DressCode

[1]A full specification of the DressCode language and API is available at: **http://jacobsj.scripts.mit.edu/dresscode/index.php?title=Drawing_API_Reference**

programming language. Each tool correspond directly to a method in the drawing API. More importantly, the use of each tool automatically generates a corresponding textual statement in a user's program. This enables elements that are created graphically, to be immediately manipulated through textual programing, and is designed to encourage a natural flow between graphic drawing and textual programing throughout the design process. The toolset includes regular primitive creation tools (ellipse, rectangle, polygon, line curve), and an svg import tool. There is also a pen tool that allows for the creation of irregular forms. Use of the pen tool generates a list of points in the text program, and statement initializing a polygon with the list. Because of the symmetry between the DressCode graphic tools and the DressCode language, there is no functionality hidden from the user in the process of modifying elements graphically. The program that results from a graphically modified design is human readable, and can be frictionlessly shared with other users, or copied and pasted into other DressCode programs.

(figure:2). In addition to the primitive generation tools, there is also a selection tool, which allows for individual primitives and groups to be manually selected with the cursor, and moved to different points on the drawing board. When a primitive is moved for the first time, a textual move statement is inserted into the user's program, with the identifier for the moved primitive as the first argument. If the moved primitive does not have an identifier, the primitive declaration is wrapped by a move statement. For all subsequent moves of that primitive with the move tool, the inserted move statement is updated to reflect the new coordinates of the primitive.

Figure 2. Graphically created polygon, rectangle, and irregular polygon, and corresponding automatically-generated code. The irregular polygon has just been moved with the selection tool.

It is also possible to select primitives in the declarative view panel. The declarative view contains a listing of all primitives in the current design, listed by their type (polygon, line, ellipse etc.) and identifier. Child primitives of groups are shown as nested elements below the group. When a primitive is selected in the declarative view, the primitive is simultaneously selected (and highlighted) in the design view, and the line where the primitive was last modified in the textual program is highlighted (figure:**??**). This declarative view functionality is designed to provide visual feedback on how elements of a design connect to the a user's program, and provide a practical selection technique for selecting and modifying individual children within a complex group.

Figure 3. Declarative view with selected primitive

Finally, DressCode contains functionality to help people organize their code in the form of *stamps*: auto-generated functions that return shape primitives. Two forms of stamps can be created: dynamic and static. Dynamic stamps are created by selecting a portion of code in a user's program. and then selecting the "create stamp from selected code" option from the menu. A dynamic stamp will package the selected code in a function with a name specified by the user, and automatically return any primitives that are generated by the selected code in the last line of the function. Static stamps are created by graphically selecting a primitive or group of primitives with either the selection tool or the declarative view, and selecting the "create static stamp from selected objects" option. Similar to dynamic stamps, static stamps are a function with a user-defined name that return a copy of the primitive or group that is selected. Static stamps differ in that they contain an explicit representation of the selected primitives. Static stamps were created to assist in the process of generative design. For example, if you create a random collection of shapes using a random or noise method, the next time the program is run, a different random collection will be generated. If a user wishes to maintain a specific instance of a generative design, they can create a static stamp from it, which will contain methods describing that discreet representation of shapes. Stamps are listed the stamp menu and can be added to a user's primary program by selecting the "+" icon next to each stamp. The functionality of both static and dynamic stamps can be modified by the user, by selecting the stamp from the menu, which will display the stamp code in the primary code window in an editable format.

ACTIVITY DESIGN

Domain-specific software is an important component of engaging youth in algorithmic craft; however it is equally important to appropriately design the context in which the software is applied. In this section, we discuss the primary components of our activity design, for which DressCode was the primary design tool. Because algorithmic craft combines aspects of computation, design and craft, we focused on an activity structure that would emphasize key components from each of these domains through three forms: learning, creation and reflection.

Learning

Instruction in our algorithmic craft activities was divided between imparting conceptual understanding of the creative opportunities of algorithmic craft, and teaching practical skills necessary to act upon this knowledge. We presented participants with a conceptual overview of the core principles described in the Algorithmic Craft section, by sharing examples of professional projects that demonstrated these properties. We made a particular effort to link the properties of computational design to more familiar contexts, for example comparing parameterization to the process of creating a size-variable dress pattern for sewing, or using a coin flipping and drawing activity to demonstrate the properties of randomness and generativity. To provide participants with the required technical knowledge for computational design, we conducted guided activities in programing in DressCode. For the programing lessons, we limited instruction to a specific design principle, such as radial symmetry or a random walk, and assisted participants through the process of creating their own algorithms that expressed this principle. For the craft skills, the instructors demonstrated the basic techniques of the craft activity, such as how to pull ink through a screen for screen printing, or how to set snaps into leather, and then gave participants the opportunity to try the process themselves.

Creation

Because one of the primary objectives for algorithmic craft activities is to have participants emerge with a completed artifact, we devoted the majority of activities to time for the participants to develop their designs and projects. Following each learning portion, we provided participants with time to experiment with the concepts or techniques that were covered, and develop their own interpretations. We also conducted longer-term design sessions that lasted 1-3 hours each and gave participants the opportunity for open-ended computational design. These sessions were supplemented by providing participants with DressCode examples which they could reference, modify or remix, quick syntax-reference sheets and an online code reference. Instructors were also present to help with programing questions, or offer design suggestions. The physical craft materials were made available to participants during the computational design sessions, so they could take their properties into account as they created their design.

In the crafting and fabrication creation portions we emphasized hands-on practice and experimentation. Abundant test materials and extra supplies were provided to encourage participants to try crafting techniques until they were comfortable with the process. For activities that relied on digital fabrication, participants were given instruction on the use of the fabrication machines and then operated them independently under instructor supervision. We also encouraged participants to work collaboratively during the construction process, having groups of two or three people work to complete one individual's artifact, and then transition to working on the other's project.

Reflection

At each stage of creation during an activity, we added an opportunity for reflection. Participants were introduced to the process of a design critique at the start of an activity, and then engaged in group critiques following each design and craft stage. For the critiques, we provided participants with aesthetic and practical criteria to guide their feedback. This ranged from composition and visual flow of a design, the level of complexity, the wearability of a design (how it would look when worn), and how compatible a design would be with the crafting process. Following the conclusion of an activity, participants shared their completed artifacts and reflected on their design process, and their experience combining the software tools and craft practices.

Although we discuss learning, creation and reflection separately, the boundaries between each of these components were blurred through the activity as a whole. There were also opportunities for each component to occur multiple times throughout the activity, and their ordering was deliberate. For example, we engaged participants in a learning activity on craft practices, and directly followed with an open computational design session, in order allow them to take craft knowledge into account for their digital design.

USER STUDIES

We evaluated DressCode and our activity structure through one preliminary study and one long-term evaluation. Both workshops focused on using DressCode to produce wearable artifacts. The preliminary study consisted of a one-day workshop in which participants designed and fabricated leather wrist-cuffs. The long-term evaluation was conducted through a four-day workshop, divided between two consecutive weekends, in which participants created computational designs and screen-printed them onto t-shirts. The preliminarily study used a version of DressCode that was absent of the graphic drawing tools, stamps and the declarative design view, whereas the long term study included the a version of DressCode with these features. Comparison of the experiences of the participants between the two workshops therefore provides feedback on the effects of introducing the graphic tools.

Evaluation Methods

The participants' experiences were evaluated through written surveys, interviews and group discussions. The surveys were aimed at understanding participants prior attitudes towards programming, design and craft, their interest in and attitudes toward programming and design, and their engagement in and enjoyment of the workshops. Pre-workshop surveys focused on participants previous experience and attitudes towards programing, craft and design, how personally useful they felt programing to be, and how good of a tool they found it for creativity, personal expression. They also asked students to describe their views on design, programing and craft practices as they related to referencing examples, learning from other's work. 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 design process, their opinion of the success of their projects and their experience using DressCode. Interim surveys were also administered during the primary workshop following the first two days, and questioned the participants on experiences with DressCode programing language and graphic tools and how the combination of features hindered or supported the design process.

In-person interviews were conducted with the participants in the preliminary workshop. These interviews lasted an average of 10-20 minutes and were audio recorded and transcribed. During the interviews, the participants were asked to describe their artifact, and talk about the process of conceptualizing, designing, and producing it. In the primary study, participants were engaged in three group discussions at the start, middle and end of the workshop which were similarly recorded and transcribed. The discussions examined participants' design process, experience using DressCode in combination with craft practices, and ideas they had for modifying or augmenting future activities and software tools. In both the interviews and discussions, participants were also 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 and discussion responses and project outcomes were analyzed to determine how they connected to the primary research questions we associated with algorithmic craft in the introduction. We also used this information to identify repeated and notable elements of participants experiences, and formulate a set of criteria for future development in algorithmic craft.

Preliminary Study: Wrist cuffs

The wrist cuff workshop was conducted among 10 young adults, aged 15-17, 20% male and 80% female. Of those surveyed, three participants had prior experience with Scratch, and two had worked briefly with the Arduino programming environment in a prior workshop. All 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 programming on their own; however, the majority indicated they were interested in learning more about the process.

During the workshop, participants used DressCode to design a pattern for a wrist cuff, laser-cut their patterns into leather and assembled the finished wrist cuff by hand. The computational design in the preliminary study was structured around radial symmetry, and the majority of the resulting artifacts featured patterns that were derived from radially symmetric structures. Participants wrote their own radial symmetry algorithms, and were then provided with a template in DressCode that automatically clipped their designs to the dimensions of a cuff sized to their wrist. Participants were given two hours to design, and 3 hours to fabricate and assemble their piece.

Preliminary Results

Each participant in the preliminary workshop was successfully able to use DressCode to produce a unique leather cuff. The design approaches and aesthetics of participants varied greatly. Some cuffs were geometric, some were floral, and many styles were difficult to link back to the original radial algorithm. Each participant was able to write their own radial symmetry algorithm in the initial lesson and produce a variety of patterns with it. During the design portion, several participants extended the radial symmetry algorithm with additional design elements, however most participants designs relied exclusively on structures that were possible with radial symmetry. Eight of the participants said they planned to wear the bracelets they created, and two indicated that they planned to give them as gifts. A comparison of the pre-and post-workshop surveys demonstrated increased comfort with programming as a result of the activity. After the workshop, more participants indicated that they thought programming was a good tool for personal expression and design. All participants stated that they believed they could use programming to make things that were beautiful, whereas several had disagreed with this statement before the workshop. All participants unanimously identified the programing portion as the most difficult component of the workshop, however the majority said they were interested in continuing to learn more about computational design for craft applications.

Primary Study: Screen Printing

The screen printing workshop was conducted among 7 young adults, aged 13-17, and one older participant, aged 21. Three participants were male and five were female. Participants for the workshop were deliberately selected to represent a range of programing and design experience. Two participants, (one being the oldest participant), were relatively experienced programers, having worked extensively with java, python and lua. Two participants considered themselves intermediate to novice programers with basic experience in Scratch, HTML, Python or App Inventor. Two other participants had prior exposure to Scratch, but had not worked with it extensively. One participant had no programing experience, but was interested in learning. Participants also varied on their prior experience with digital design software. Two indicated that they had no prior experience with design software, two had previously used the graphics editor in Scratch, three had used Adobe Illustrator or Photoshop, and one person was shown Solidworks in a summer course, but had limited experience with it. The majority of participants indicated they had a variety of prior crafting experience, including making jewelry, basic woodworking, sculpture and origami. One person had screen printed before using vinyl cut stencils. Interestingly, one participant also identified soldering a electronic synthesizer kit as a craft activity. Only one participant stated that they had no prior craft experience.

Because the screen printing workshop occurred over a four-day period, it provided the opportunity to explore the computational design process in DressCode in greater depth, and allowed for the participants to engage in a more complex craft process: photo-emulsion based screen printing. Prior to the workshop, we had participants select t-shirts in a size and color of their preference. The first session of the workshop, we introduced participants to the concept of generative design, and demonstrated techniques for incorporating random elements into a pattern or graphic. We introduced participants to textual programing in DressCode by guiding them through writing their own random walk algorithms that produced patterns composed of lines. Following this, we explained the functionality of the graphic drawing tools, and had participants modify their random walk patterns by incorporating shapes they created graphically. The following session we discussed the general process of screen printing and demonstrated several other forms of generative design first we provided them with images of sample designs and had them work in groups to write a set of instructions that they thought would re-produce that design. As a group we discussed their results, and then demonstrated examples in DressCode that would produce similar results to the sample designs (figure:4). Participants were then provided with the design task of creating a computational design to screen print onto a t-shirt they would want to wear, and given 3 hours of open design time with the option of modifying the examples we provided, working off of the original random walk pattern they had created the first day, generating a design from scratch, or working with an instructor to come up with additional design ideas. The following session participants prepared their screens for screen printing, by stretching the screen material over the frames in groups and applying the photo emulsion to their screens. Afterwards, participants were given 2.5 hours of additional open time to complete their designs. Participants laser-printed their finished designs on transparencies, which the instructors exposed on the screens overnight (figure:5). The final session, participants practiced printing with their designs on test materials, and then printed onto their t-shirts.

Figure 4. Sample Designs and corresponding examples in DressCode

Figure 5. Screen printing process

Screen Printing Results

Each participant in the primary workshop was successfully able to use DressCode to produce a design for their t-shirt. The design approaches among the participants varied. Two participants created designs that were derived from their random walk algorithms, one participant created a pattern by combining two of the example patterns and making adjustments, and one participant worked solely by modifying an example. Other participants designed patterns independent of the examples, including a generative landscape, a geometric spiraling pattern and a complex radial pattern comprised of overlapping lines. The screen printing process was extremely popular among the participants. Each person was successful in creating their screen, transferring their design to the screen and printing to the shirt, although one participant had to re-expose his screen because of errors in applying the emulsion. Several participants not only printed to the provided shirts, but also brought in additional garments to print on for friends and family. The participants requested to keep their screens following the workshop, and stated on the survey that they planned to continue making prints for themselves and others. All participants indicated that they planned to wear their shirts, and two participants contacted us via email following the workshop thanking us for the experience, and requesting tips on how to properly care for their garments.

From survey data and group discussions, 3 participants found the screen printing portion to be most difficult, 3 found the programing to be the most difficult, and one was equally challenged by both portions. An evaluation of the pre, mid and post-workshop surveys demonstrated that following the screen printing, participants attitudes towards programing, design and craft, changed from what they were after the computational design sessions. In the majority of cases, this change was positive; participants indicated greater interest in learning programing in the future, a stronger belief that programing was a tool that they could create things they would use in their daily life, and greater comfort in programing on their own following the craft activity. Six of the eight participants stated that they liked their design better after printing it, and all participants stated that the process of screen printing had given them ideas for additional designs they would like to create with DressCode. For a minority of participants however, the difficulty of the craft portion seemed to result in a slightly diminished interest in combining craft and programing, and a diminished interest in craft in general. Participants were positive about the graphic tools, although they requested that their functionality be extended to incorporate a greater range of transformation methods (rotation, scaling, and boolean operations). All 8 participants said they would be interested in using DressCode for another activity, and the majority indicated that they would like to combine computational design with 3D printing.

ANALYSIS AND DISCUSSION

Generative Design Practices for Physical Objects

Discuss success / challenges of Stamp tool, GUI drawing and transformation tools
Embracing of Benefits of Computational Design by practitioners
Emergence of narrative in random structures

Forms of knowledge Sharing in Algorithmic Fabrications

Levels of design- tinkering with examples to building from scratch
Sharing code and sharing craft artifacts
Teaching physical techniques
Blurring the boundaries between technical programing advice and objective design criticism.

Physical and Digital Notions of Value

Conceptions of Difficulty
Digital and Physical Craftsmanship
diversity of experience (things people found to be difficult vs enjoyable)
Added value of code-derived artifact.
Added value as the result of physical labor.

Personal and Cultural Relevance in Algorithmic Craft

Aesthetics- enabled through code and graphic manipulation
cultural appropriateness of aesthetics (gender, youth)
Utility- changes in understanding of applications of programming, design. Material importance

Opportunities in Craft and Code Hybridization

Digital Feedback mechanisms vs social feedback mechanisms (critique)
Starting with craft- Desire for enhanced drawing functionality
We also asked if they would have preferred to create designs for their screens by drawing them by hand, rather than through computational design. Two participants expressed interests in trying a non-computational approach; however most participants stated that they appreciated being able to randomly place components of their design.
Open question of what the resultant product is. (screen? artifact? program?)

RECOMMENDATIONS FOR FUTURE DESIGN

Transparency

- **Tools should support design techniques that demonstrate the benefits of computational design.**
- **The software interface should prioritize interaction through programming.**
- **Programming languages should have a design-oriented programming syntax, developed with novice programmers in mind.**

Literacy

- **Tools should reduce the technical challenges of fabrication.**
- **Tools should be supported through fabrication and craft-specific documentation.**
- **Activities should encourage discussion and group critiques of work, and when possible, use in-person facilitation and guidance.**

Flexibility

- **Activities should blend domain specificity with creative openness.**
- **Software tools should be free and open-source.**

Specificity

- **Materials matter.**
- **Select approaches with Craft and Computational Resonance**
- **Consider the cultural implications of algorithmic craft experiences.**

CONCLUSION

REFERENCES

1. *A Model Curriculum for K-12 Computer Science:Final Report of the ACM K-12 Task Force*, second edition ed. Association for Computing Machinery, 2003.
2. Report To The President. Prepare and Inspire: K-12 Education In Science, Technology, Engineering and Math (STEM) for America's Future. Tech. rep., Executive Office of the President, 09 2010.
3. Make to learn. `http://m2l.indiana.edu/`, 2013. (Accessed: 09/12/2013).
4. Maker education initiative. `http://http://makered.org/about/`, 2013. (Accessed: 09/12/2013).
5. McCullough, M. *Abstracting Craft: The Practiced Digital Hand*. The MIT Press, Cambridge, Massachusetts, 1996.
6. Rosner, D., and Ryokai, K. Reflections on craft: probing the creative process of everyday knitters. In *Proceedings of the seventh ACM conference on Creativity and cognition*, C&#38;C '09, ACM (New York, NY, USA, 2009), 195–204.