**DressCode: Software to Support Algorithmic Craft (Better title needed...)**

**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/**](http://www.acm.org/about/class/1998/) for more in- formation 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 com- bination 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 inter- national industry and advancement has resulted in the STEM Education Initiative, a national drive to engage young peo- ple in topics of math, science, engineering, and computer science [2]. Proposed K-12 computer science educational curriculums are aimed at teaching students the core compo- nents of computation and programing, as well as fostering sustained interest in a broad range of computer science pro- fessions [1]. These approaches explore digital applications of computer science that will appeal to young people, in- cluding game development, digital media manipulation, and web development. Parallel to the growth of youth STEM ini- tiatives 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 cre- ate 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 connect- ing making with principles in science, technology and math;

Paste the appropriate copyright statement here. ACM now supports three different copyright statements:

*•* ACM copyright: ACM holds the copyright on the work. This is the historical ap- proach.

*•* License: The author(s) retain copyright, but ACM receives an exclusive publication license.

*•* Open Access: The author(s) wish to pay for the work to be open access. The addi- tional fee must be paid to ACM.

This text field is large enough to hold the appropriate release statement assuming it is single spaced.

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 ap- plied to the arts, programming offers different paradigms for creation and new methods of problem solving. We see an op- portunity 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 princi- ples to consider when creating programming environments for physical design? How do we compellingly link textual code with visual designs, and what are the appropriate inter- section points between textual manipulation and visual ma- nipulation? 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 translat- ing 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 pro- gramming 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 par- ticipate in computational design. We designed and imple- mentation of a set of algorithmic craft activities in which high-school students used DressCode in conjunction with tra- ditional 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 en- gagement, 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 interac-

tive. 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 struc- tures 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 al- lows for the programmer to create algorithms that allow for the computer to autonomously produce unique and un- expected 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 gener- ated designs are generated by a program which can be mod- ified by other designers, and serve as a form of documen- tation of the design process.

Computational design also contains a number of unique chal- lenges:

*•* **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 devi- ation. Because computational design is governed by a sys- tematized 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 his- tory, however forms of craft have endured, both as a recre- ational pursuit, and as set of valued artisanal practices. Below we describe the specific aspects that are integral to algorith-

*•* **Materiality:** Craft involves the manipulation of physical materials by hand, which is often an intuitive physical pro- cess. 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 con- ceptions 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 em- phasizes the importance of beauty in the form and or- namentation 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 prop- erties, while incorporating patterns and forms made possible through computational processes.Objects that emerge from a combined practice of digital fabrication and craft are of- ten 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 fabrica- tion is that it ensures that machine-produced artifacts retain their personal history. Finally combining computational de- sign 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 de- sign with the physical world of craft, some form of transition is needed between the digital and the physical. Digital fab- rication technology provides one method of connecting the abstraction of computational design and the material domain of craft by converting digital designs to physical forms. Digi- tal 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 pro- cesses Unlike most 3D printing technology, subtractive pro- cesses can work with a wide range of materials [**?**].

Time is an important quality of crafting. Handcrafted arti- facts often require a great deal of time to complete and in- volve repetitive tasks. Because the process of crafting is often pleasurable and contemplative, many people look for- ward 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 computa- tional forms to physical materials. These physical forms, in turn, can be shaped through craft processes, and imbue com- putational designs with individuality, utility, and temporal- ity. Several forms of digital fabrication machines are capa-

novice levels of handcraft. Because subtractive fabrication technologies work with materials like wood, paper, and fab- ric, 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 cut- ters 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 fabri- cation, 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 intellec- tual: learning the unique affordances of computation and ap- plying them in personally meaningful ways technical- chal- lenge of using programing for design Focused on software design for the technical, activity design for the perceptual

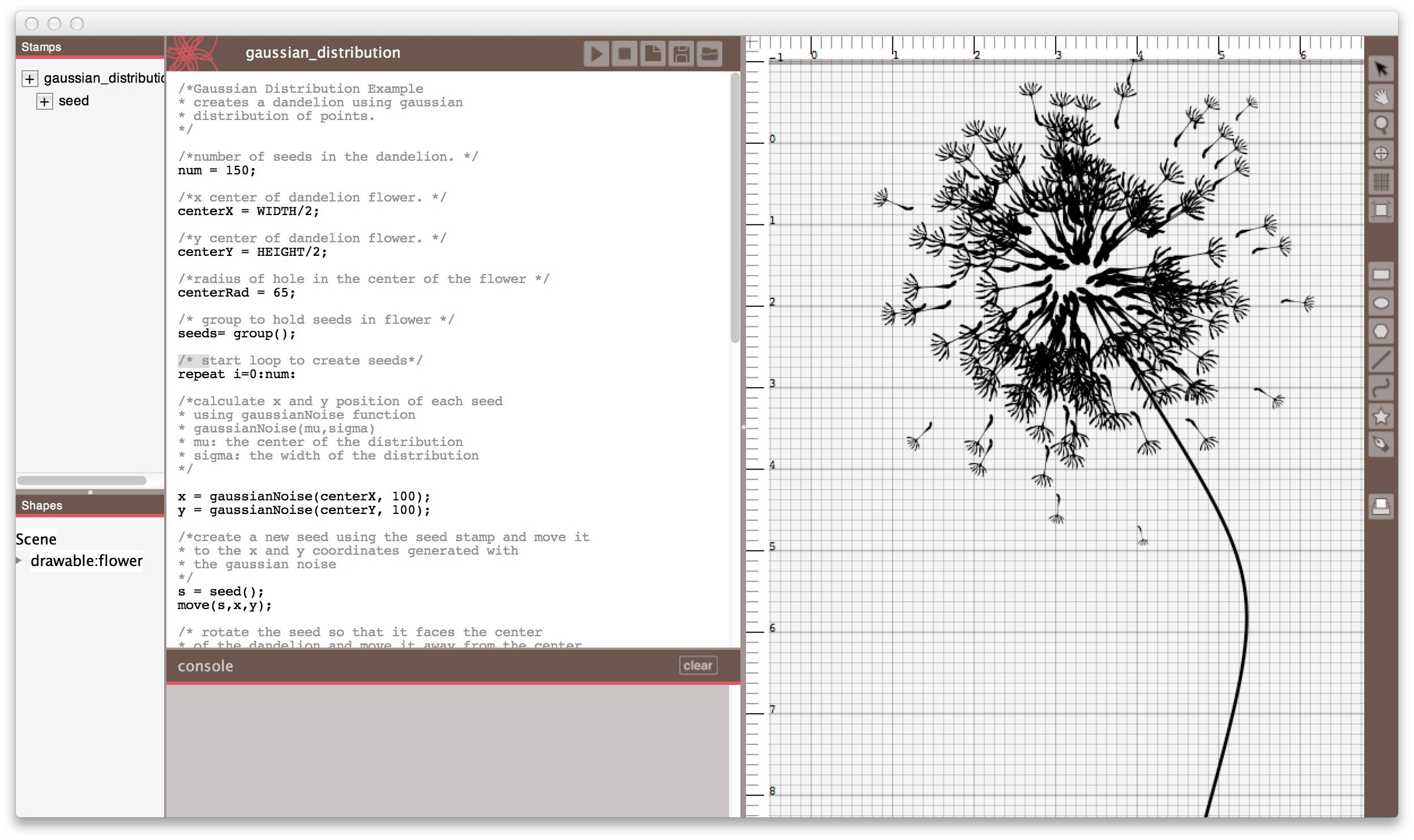
**RELATED SOFTWARE TOOLS Professional Tools**

**Entry-level Programming Environments**

**Entry-level CAD Tools**

**DRESSCODE SOFTWARE DESIGN**

DressCode is a 2D computational design tool aimed at novice programers. The primary objectives in the design of Dress- Code were to reduce the technical challenges of computa- tional design, to foster independent and deliberate design de- cisions, and to assist in the creation of designs that were vi- able for craft applications. Although the name of the tool reflects an emphasis on wearables and fashion, as an appli- cation 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 inte- grated development environment, and a graphic user interface design tool.



**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 coding panel contains a primary window for entering text, and an output console for print output and error reporting. 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 dis- played in the design panel. The DressCode programming lan- guage is interpreted with semantic functionality that is simu- lated through a Java-based library. For most programs, the interpretation process is instantaneous; however, some pro- grams with complex operations require several seconds to be executed.

The design panel approximates many of the features of a dig- ital graphic design tool, with rulers, a grid, and a drawing board. The drawing board defines the coordinate system ref- erenced by the drawing API with the upper left hand corner corresponding to (0,0) in cartesian coordinates. Users can re- size the drawing board and set the units to inches or millime- ters at any point during the design process. The print button opens a dialog that allows the user to export their current de- sign in a vector format for output through digital fabrication.

Early in the development process, we experimented with au- tomatic interpretation, wherein the design would be automat- ically 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 run- ning their program and updating their design.

**Programming Language and Drawing API**

DressCode features a custom textual programing language with native 2D drawing functionality. 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 re- duce syntactic challenges for people new to textual program- ing. 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 organized around the cre- ation and transformation of shape primitives including points, lines, curves, polygons, ellipses, rectangles and imported SVG shapes.There is no “draw” method for DressCode. In- stead all primitives are automatically drawn in the display panel in the order of their initialization. This was designed to make it as easy as possible to have one’s design appear on the screen. If a user does not want a primitive to appear on the screen, they can hide it with the hide method.

All primitives can be modified through a set of transforma- tion methods that allow them to be rotated, scaled or moved. Transformations on a primitive are performed by assigning an

1 //repeat statement repeat i=0:10:

3 ellipse(0,i\*10,10,10); //draws a vertical row of 10 ellipses

**Primary Study (Screen Printing) Screen Printing Results ANALYSIS AND DISCUSSION**

end

method with the identifier. By using the transformation meth- ods to manipulate primitives in a structured manner, it is pos- sible to generate complex and interesting designs from sim- ple forms. Through these methods, DressCode was intended to support the affordances of computational design, specifi- cally precision, visual complexity, generativity and stylistic abstraction, in a way that was feasible for new practitioners.

DressCode also contains a set of Polygon boolean transfor- mation methods that allow shape primitives to be combined in various ways. Polygon booleans are widely used in CAD ap- plications and are essential for many forms of digital fabrica- tion because they allow for complex configurations of shapes to be combined into effective toolpaths. DressCode contains methods for performing unions, intersections, differences and either-or intersections between any two or more shapes. It is also possible to clip a group of shapes within the bounds of a single shape to serve as a form of clipping mask. They can also expand strokes to filled polygon paths, enabling the translation of line art to a form that will maintain its appear- ance when fabricated. By positioning these operations as pri- mary components of the DressCode, we attempted to create a programming language that produced forms that were ap- plicable to fabrication as a direct result of the design process.

1 .

We chose to create a textual programing language for Dress- Code because we believe textual programing is well suited to transparent representations of computational design algo- rithms. Simultaneously, we sought to reduce the challenges of first-time text programming by providing intuitive forms of interaction and feedback through the graphic drawing tools.

**Graphic Drawing Tools ACTIVITY DESIGN Materials**

**Background Code Instruction Craft Instruction Critiques**

**USER STUDIES Methodology**

**Preliminary Study (Bracelets) Bracelet Results**

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

[**php?title=Drawing\_API\_Reference**](http://jacobsj.scripts.mit.edu/dresscode/index.php?title=Drawing_API_Reference)

**Generative Design Practices for Physical Objects**

Discuss success / challenges of Stamp tool, GUI drawing and transformation tools

Embracing of Benefits of Computational Design by practi- tioners

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 program- ming, design. Material importance

**Opportunities in Craft and Code Hybridization**

Digital Feedback mechanisms vs social feedback mecha- nisms (critique)

Starting with craft- Desire for enhanced drawing functional- ity

Open question of what the resultant product is. (screen? arti- fact? program?)

**RECOMMENDATIONS FOR FUTURE DESIGN Transparency**

*•* **Tools should support design techniques that demon-**

**strate 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 program- mers in mind.**

**Literacy**

*•* **Tools should reduce the technical challenges of fabrica- tion.**

*•* **Tools should be supported through fabrication and craft-specific documentation.**

*•* **Activities should encourage discussion and group cri- tiques of work, and when possible, use in-person facili- tation 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 Reso- nance**

*•* **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/**](http://m2l.indiana.edu/), 2013. (Accessed: 09/12/2013).

4. Maker education initiative.

[**http://http://makered.org/about/**](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.