Recoding Product Design Education: Visual coding for human machine interfaces

Conference Paper

Recoding Product Design Education: Visual Coding for Human Machine Interfaces

James Novak*, and Jennifer Loy

Griffith University, Australia

Abstract

This paper evaluates the impact of visual coding on the Industrial Design and 3D Design disciplines, in particular the role it plays in developing new products and services that would previously require interdisciplinary teams, or significant training beyond the scope of these disciplines into text-based coding and electrical engineering. The professional practice of designers working at the intersection of product design and coding is discussed, and design education evaluated in relation to the opportunities of electronics and visual coding. Quantitative research data is provided to support an argument that visual coding can enable designers to control their designs in new ways throughout the design and prototyping process.

Keywords: Visual programming language, industrial design, prototyping

Corresponding Author: James Novak; email: j.novak@griffith.edu.au

Academic Editor: Jennifer Loy

Received: 28 November 2016
Accepted: 4 December 2016
Published: 9 February 2017

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the DesTech Conference Committee.

OPEN ACCESS

1 Introduction to Visual Coding

Design is rapidly evolving with "the coming generation of designers see(ing) code as a kind of material just as a potter sees clay." (Sauter, 2011, p. 5) Visual Programming Languages (VPLs) are becoming increasingly integrated into conventional 3D Computer-Aided Design (CAD) software, specifically enabling industrial and 3D designers to engage with coding simultaneously with the development of the 3D form of a product system. This code, just like the 3D modelling environment, is visual and intuitive, appealing to designers' innate visual understanding of three-dimensional space and two-dimensional diagrams.

As a discourse, VPL is a field dating back to the nineteen eighty's, but is only newly finding momentum through practical applications as technologies and graphics catch up to the theory. In a very broad sense, Brad Myers' 1990 definition of VPL being "any system that allows the user to specify a program in a two-(or more)-dimensional fashion" (p. 98) still rings true, contrasting traditional text-based coding which is described as being one-dimensional in nature, and "not utilising the full power of the brain." (Myers, 1990, p. 100) To provide a visual comparison, figure 1 shows a basic piece of code that controls an LED flashing at one second intervals on an 'Arduino Uno' microcontroller which is a popular open-source tool for controlling electronic sensors and actuators. On the left is the traditional text-based code to control this action using the

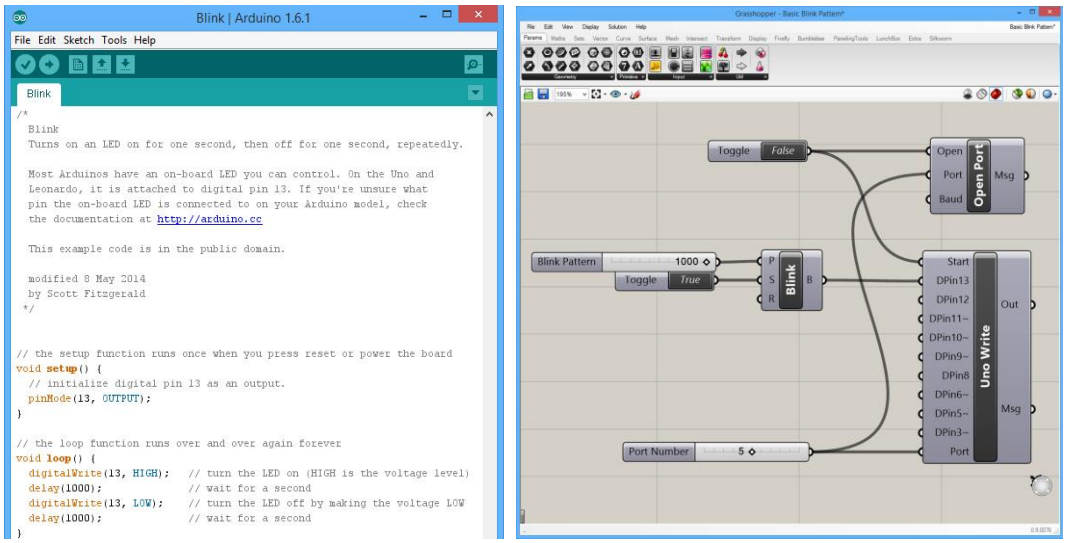

Figure 1: Text-based Arduino code for blinking LED on the left versus visual code in Grasshopper on the right 2015, Screen capture.

native Arduino software, while in the right panel is the same command within a program called Grasshopper, which is a plug-in for a CAD program called Rhino that uses a visual programming language. Both methods perform exactly the same task, however are visually quite different and engage the brain in different ways.

2 Changing Role of the Designer

The argument in this paper is that VPLs, such as Grasshopper, have developed to the point that they are forcing a blurring of the boundaries between the traditionally distinct disciplines of design, programming and electrical engineering, and so significantly changing practice and design outcomes. A single designer is now able to model, prototype, test and even manufacture complex electronics or applications using visual tools and workflows potentially within a single piece of software, as in the case of Rhino with the Grasshopper add-on. An example of this type of project is 'MyPen,' published in the proceedings of the 2015 Drawing International Brisbane conference (Novak, 2015). In this project the design of a 3D printable, customised pen is developed in the virtual environment with customisation completed in Rhino using pressure sensors and a webcam in the real world. This weaves together the development of ergonomic, aesthetic and electronic elements, allowing them to be created simultaneously. The graphical tools of Grasshopper turn the complex process into logical blocks of code which flow from one to the next, sharing much in common with the mind maps and flow diagrams that designers commonly use throughout the design process. The ability of a designer to learn a tool like Grasshopper is therefore increased, because of the similarities to practices already employed. Rather than needing to learn a completely new skill - or

in the case of traditional coding, a completely new language – the designer can draw parallels with previous experience and discipline knowledge to engage with electronics during design development.

This capacity to streamline the learning process and quickly adopt a VPL like Grasshopper has been similarly observed by Gabriela Celani and Carlos Vaz (2012), who compare programming methods used in Architecture and in the education of Architecture students. "The comparison between textual and visual programming languages showed that the later can lead to better results with novice architecture students. However, without any textual programming knowledge, applications are restricted to parametric explorations" (Celani & Vaz, 2012, p. 135). António Leitão and Luís Santos arrived at a similar conclusion in a 2011 study, finding that "modern TPLs (Textual Programming Languages) with user-friendly IDEs (Integrated Development Environments) can be much easier to program and understand than the older ones, and they can surpass recent VPLs, especially in complex tasks" (Leitão & Santos, 2011, p. 556). Debate around the effectiveness of VPLs continues, however the MyPen example demonstrates how VPLs have matured to allow for an enhanced complexity compared to that available in 2011-2012, evolving like most technologies at a rapid pace and providing designers with the tools to control multiple elements of the design process.

To further evidence the spread of VPLs and rise in their popularity, Scratch (https://scratch.mit.edu/) and App Inventor (http://appinventor.mit.edu/), both visual coding environments, have grown out of research from Massachusetts Institute of Technology (MIT), to become successful cloud-based products. Scratch provides a programming space targeted at children and teenagers to encourage an interest in coding, although "a sizeable group of adults participates as well," (Resnick et al., 2009) and uses visual blocks that resemble puzzle pieces to connect pieces of code together and build games, interactive stories etc. App Inventor provides a similar workspace for users to create Android applications. At this time, over sixteen million projects have been shared on the Scratch website (https://scratch.mit.edu/statistics/) with over twelve million registered users, indicating that VPLs have moved out of experimental research into mainstream adoption as genuine programming methods.

It is important to make the distinction between these technological approaches and text-based coding that text will likely maintain its status as the form most suitable for managing large interconnected systems of electronics, as described by Celani and Vaz (2012), or Leitão and Santos (2011) due to its ability to cope with more complex programming and problem solving. The creators of Scratch also acknowledge this, stating that "for some Scratchers, especially those who want to pursue a career in programming or computer science, it is important to move on to other languages" (Resnick et al., 2009, p. 66). However the significance of VPLs for industrial and 3D designers is that they empower the simultaneous development of an application, interactive component, or electronics with the physical product, whether it is within the same piece of software like Rhino, or separate from the CAD file in the case of Scratch or App In-

ventor. Designers are not necessarily interested in becoming programmers as their role is broader, encompassing ergonomics, user experience, aesthetics, form, manufacturing constraints etc. However, VPLs provide designers with workflows akin to those of other visual practices common to the design process, and thus maintains their freedom to remain creative, whilst engaging more fully with the interaction design integral to their product. Using visual representations of programming allows for rapid design iteration, testing and prototyping, which may result in everything necessary to manufacture the end-use product, or may then be passed to collaborative programmers or electrical engineers to finalise for production. This would traditionally occur much earlier in the design process, with multiple stakeholders working individually on components before bringing them together, and can sometimes result in disparities between elements when the original vision is not properly understood and executed by all parties. By empowering designers to develop all facets of a product, at least to a prototyping level, it is foreseeable that design problems may be more successfully solved, or indeed re-imagined through the unique lens of a designer.

3 Re-coding Designers

As the digital revolution increasingly impacts on industrial practices and production systems around the world, there is a growing imperative for industrial and product design students to be educated in digital technologies and their disruptive influence. On a practical level, this means extending their portfolio of skills to include technologies such as design for digital fabrication, the generation of data using scanning technologies and its manipulation in preparation for product development, and, as discussed in this paper, the integration of an understanding of the development of electronics and applications during the design process. However, preparing students for professional practice post digital revolution involves more than adding to their skills base and their understanding of designing for digital technologies; it involves helping students rethink the traditional roles of the designer and explore the changing digital landscape. This is not only a challenge for the students, but for faculty, who require professional development in both skills and the changing paradigm around product and industrial design education.

The re-coding of designers begins with changes to the curriculum. Whilst faculty may need to be encouraged to re-skill in this area, millennium students (born at the turn of the century), appear to be embracing the accessibility that digital technologies provide. The rise in the maker society, described by Chris Anderson (2014), has been echoed in the reconnection of design students with making, based on computer numerically controlled (CNC) technology. First year students are now able to laser cut, CNC router and 3D print prototypes, to complement traditional model making. This has led to a renewed enthusiasm for workshop practice (Loy, 2014) based on digital technolo-

gies. This shift provides a good basis for changes to thinking and practices necessary for the adoption of VPL by designers. This adoption should increase the integration of electronics into design process, and should make the changes to the curriculum more easily accepted and help faculty to further the development of digitally enabled design practice.

A study of the introduction of visual programming language (VPL) into the design curriculum was conducted by Celani and Vaz (2012). They discovered that VPLs were well-received by Architecture students who were unfamiliar with traditional coding. Navarro-Prieto and Cañas also found that "spreadsheet programmers develop a data flow mental representation even in the easiest tasks, while C programmers seemed to have more problems acquiring and using this information" (2001, p. 822). Whilst there are many contradictory arguments about the usefulness and scalability of VPLs, there appears to be a general consensus that they are adopted and understood far quicker than text-based languages amongst new learners of programming, and in particular learners in visual disciplines, such as Architecture and Design.

This was demonstrated in research based in a new course studied at Griffith University in 2015, called Human Machine Interfaces. This course was core for Industrial Design students. The research was on providing a small group pilot study for the viability of the adoption of VPL by Industrial Design students to inform future curriculum planning. During this twelve-week course, students were introduced to both a text based programming language (Arduino), and a visual based programming language (Grasshopper for Rhino with Firefly). Both methods were taught side-by-side, as illustrated in figure 1, for the first three weeks of the course. Students were asked to build Arduino circuits of increasing complexity through this time, and walked through the coding process step-by-step for both methods. Students were given opportunities to experiment along the way and encouraged to push the boundaries of what they could do with the technology as the projects evolved. At the end of the three-week period, an anonymous survey was completed by the students to gain an insight into their experiences of both methods.

The results in figure 2 highlight that students considered the VPL of Grasshopper significantly easier and more intuitive to learn compared with the Arduino IDE. These are averages for the twelve students who participated in the small group survey, and some students identified as having previously worked on Arduino coding in other classes, which could impact the scores. This is a pilot study only, and more comprehensive data would be needed to for conclusive data. However, it is interesting to look at the final survey question that asked:

> *Imagine you are designing a new product that will be used to measure the distance to an object. Your design needs to give a numeric value of the distance, as well as use LED's that flash faster as an object gets closer. Your prototype might look something like [image of an Arduino prototype]. Given your current skills, which software would you choose to develop the working prototype?*

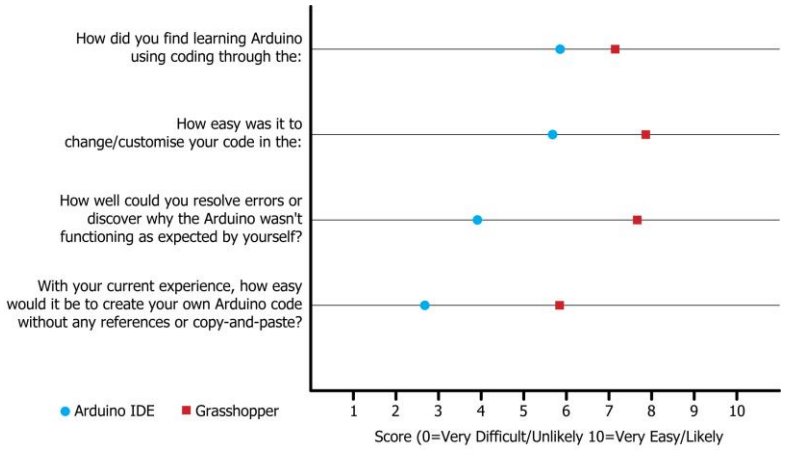

Figure 2: Programming Arduino Using the Arduino IDE Versus Grasshopper Survey Results – Human Machine Interfaces students 2015, Table.

In response, one hundred percent of the class chose to use Grasshopper. This indicates that even for students coming into the course with some previous experience of a textual coding interface, the VPL was preferred for solving problems and making a prototype in this scenario. The outcomes of this pilot exercise align with the results of Celani and Vaz's case study where "the use of the visual programming language was more successful in terms of students' enthusiasm, the complexity of the designs developed, and the understanding of computational design concepts," (2012, p. 133) Similar to the students of Architecture in Celani and Vaz's research (2012), the outcomes for the pilot study with Industrial Design students suggests that VPLs could suit the inherent visual skills of Industrial Designers over a text based interfaces. Based on the observations of students' Human Machine Interface practical projects, VPLs allow these type of students to more independently problem solve and generate more complex electronic and interface solutions for their prototypes more quickly.

4 Conclusion

Visual Programming Languages continue to advance at a rapid pace. There is a need for research in this growing field in order to understand the opportunities and limitations of evolved VPLs in relation to text-based programming. VPLs have the potential to support creative freedom in new ways and the capability of the programs is now allowing for more complex physical products enabled by electronic and parametric systems. Text based coding may continue to suit the creation of advanced software or electronic systems, but for designers, VPLs are arguably extending their ability to develop multiple facets of a project before handing control over to collaborators, adding value to the contribution of the designer and potentially changing the nature of the outcomes

created. It is a research area that has the potential to provide insight into the changing role of the designer post digital revolution.

References

C. Anderson, *Makers: The new Industrial Revolution*, Crown Business, New York, US, (2014).

G. Celani, and C. Vaz, CAD Scripting And Visual Programming Languages For Implementing Computational Design Concepts: A Comparison From A Pedagogical Point Of View, *International Journal of Architectural Computing*, **10**, no. 1, 121–138, (2012), 10.1260/1478-0771.10.1.121.

A. Leitão, and L. Santos, Programming Languages for Generative Design - Visual or Textual? Paper presented at the Respecting Fragile Places, 29th eCAADe Conference, University of Ljubljana, Slovenia, (2011).

J. Loy, eLearning and eMaking: 3D Printing Blurring the Digital and the Physical, *Education Sciences*, **4**, 108–121, (2014), 10.3390/educsci4010108.

B. A. Myers, Taxonomies of visual programming and program visualization, *Journal of Visual Languages & Computing*, **1**, no. 1, 97–123, (1990), 10.1016/S1045-926X(05)80036-9.

R. Navarro-Prieto, and J. J. Canas, Are visual programming languages better? The role of imagery in program comprehension, *International Journal of Human-Computer Studies*, **54**, no. 6, 799–829, (2001), 10.1006/ijhc.2000.0465.

J. Novak, *Drawing the Pen: From Physical to Digital and Back Again*. Paper presented at the Drawing International Brisbane, Brisbane, Australia. http://static1.squarespace.com/static/55779bbce4b004acf1e1479d/t/56aef970b6aa60cdf1c253d6/1454307702608/JAMES+NOVAK_DRAWING+THE+PEN_DIB2015.pdf, (2015).

M. Resnick, J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, and Y. Kafai, Scratch: programming for all, *Communications of the ACM*, **52**, 60–67, (2009), 10.1145/1592761.1592779.

J. Sauter, Preface - A Touch..., in *A Touch of Code - Interactive Installations and Experiences*, Gestalten, Berlin, 5–7, (2011).