"DRAWING WITH CODE": THE EXPERIENCE OF TEACHING CREATIVE CODING AS A SKILL FOR COMMUNICATION DESIGNERS

A. Benedetti, T. Elli, M. Mauri

Politecnico di Milano (ITALY)

Abstract

In this contribution we present a didactical framework aimed at teaching design students how to use coding in support to their design process. This framework has been developed during the last four years in a course named "Creative Coding", held in the Communication Design degree of Politecnico di Milano. The article firstly introduces the motivations that support the idea of teaching coding to design students, it specifies the approach to coding that is taken into account (creative coding) and it describes the strategies and the framework developed for teaching it. Lastly it presents and discusses results derived from the held editions, showing how the presented framework is capable of making design students conversant about coding and able to use this skill within a communication design project, with a beneficial boost in creativity and a better understanding of the feasibility of their ideas.

Keywords: creative coding, communication design, teaching methods.

1 INTRODUCTION

As communication designers, in the latest decades we have been seeing a constant rise of visual artefacts that are digital, dynamic and interactive. Such artefacts are made with an extensive use of digital technologies and present properties that are hardly to be found in traditional communication design products. For instance, they can automatically readapt their graphic layout according to the available space, they can automatically change their content according to certain underlying data, they can generate their own shape according to predefined logics and they can react to different inputs.

Traditional design methodologies are still the baseline for the realization of similar products, but designers have now to learn new ways to carry out their work: the ideal sequence 'Ideation – Concept – Design – Prepress – Production – Delivery' is even harder to be found in a contemporary design studio and these processes naturally move back and forth between ideation and production [1].

In other words, today we see that the interplay between creativity and digital technologies is more powerful and important than it used to be. Moreover, it can occur since the very early stages of the design process.

With no doubts, this technological interplay pushes designers to deal with a process that presents higher level of complexity, because it increases the amount of elements that are to be taken into account. However, it also opens up a world of new features that can be embedded into communication products. For example, graphics can benefit from the computational generation of images and written contents [2], typographic characters become variable and capable of rendering an undefined amount of weights and italic grades [3], corporate identities become generative [4], movies allow viewers to make decisions that affect told stories [5], [6].

1.1 Motivations for teaching coding

"Why should communication design students learn coding?" This question is bound to the matter of how design instructors should prepare their students to the complex process of designing products similar to the aforementioned ones. In order to do so, we believe it is beneficial to introduce coding to design students so for them to achieve awareness and be responsible about their future job.

As practitioners and researchers in the field of communication design we identified the following four points that motivate our assumption and explain why it is helpful to be conversant about coding.

First of all, we need to consider that designer have to collaborate with practitioners of different kinds. When designing for print a key figure is the typographer, when designing for screens, a key figure is the developer. Relationships between designers and developers can be hard and the hand-off of work between the two can be difficult too [7]. Being conversant about coding is certainly of use for designers, because it allows them to ask and address questions, to explain their work and to understand the one of developers. Furthermore, it even helps designers to create connections with other kind of practitioners, those ones that belong to fields in which code already permeated, like sciences, arts and economics. Therefore, our first motivation is a really pragmatic and dialogic.

Secondary, in the design of the aforementioned interactive artefacts we have to apply a form of procedural thinking [8], [9], [10, p. 17]. Depending on the context or on the fed data, visual artefacts may need to display a different number of items, their content may present different lengths or shapes and their positioning may depend on screen resolution. This is the case of social networks feeds, responsive websites, generative corporate identities or interactive data visualizations. In similar examples, designers work on the behaviour, the logics and the rules that underlie their visual outcomes [11, p. 236]. In other words, designers participate in the definition of a piece of software and knowing basilar coding patterns helps in better designing these products.

The third motivation lies in the interplay between coding and the creative process. Coding isn't necessary a tedious and boring act, instead it can be a playful, iterative and serendipitous experience: it can help designers to discover new possibilities of expressions, to test assumptions and to envision new ideas. Many designers already deploy coding in their everyday practice, and they see it as a tool that enables generative thinking. Thanks to it they can create and evaluate visual shapes and interactions that can hardly be done by hand with traditional tools, with the ultimate benefit of conceiving more innovative products [1], [11, p. 244], [12], [13].

Lastly, coding is not just for engineers, differently from what social constructs make us to believe [14]. As many other practitioners, designers have the possibility to appropriate it [12] or rather they ought to do it. The reason behind it, is that the development of software is bound to coding, and software strongly influences contemporary culture and products, design practice included [10], [15]. Designers have the ethical role to fully understand the products they help to develop and they must watch over the instrumentalization and the imbalance that they can bring into play [16].

The above list could be extended depending on one's subfield or approach, providing even more specific motivations for designers to learn coding. However, one of the reasons behind the introduction of the course object of this article, it is that in our experience education about coding is fragmented. This generates the risk that many students get intimidated and avoid it.

Fortunately, this doesn't apply to all of them. In fact, during the four years of teaching over which we developed the framework that we are presenting, we've always seen that students manifested a growing interest in coding and we always had a very good and heterogeneous attendance. Gladly, curiosity of students has been winning against the idea that coding is difficult and boring when in fact it can be something about creativity.

2 CREATIVE CODING

Programming has existed for long before computers: some recognize programmed devices in the early history, like the pegged cylinder used to mechanically transfer information between devices, used also in modern times in automated musical instruments [17]. Coding, on the other hand, can be seen as a step in the programming process related only to computers: with coding we can rapidly refer to predefined specifications, data structures and high-level conceptualizations, and create computer programs.

With the introduction of computing machines, programming was finally formalized in the span of decades with a "formal, rule-driven, hierarchical approach" [14] that would work best in the development of scientific machines first, and the deployment of digital products then. However, computer and technologies shifted from the status of mere instruments, to new materials for personal expression, where less formalized forms of coding can be used as means to convey an emotional and expressive message [13].

In this context, coding, that was a specific practice in the computer programming discipline, can be expanded as a practice on its own that is based on the translation of concepts, ideas, tasks and instructions into a series of computer readable actions through a process of trials and errors and in a high variety of contexts, not just software production.

In the early 2000s, the growing interest in this approach of 'coding for creativity' initiated the creation of libraries and programming languages that simplify the act of coding and corroborate the creative process of programmers, artists and designers that are using them [18]. Among these products, we can mention VVVV [17], Processing [12], openFrameworks [18] and other systems characterized by a simplified syntax.

These libraries and programs have become the de-facto tools to anyone who wants to approach creative coding, and they continue to co-exist thanks to their different degree of complexity presented to coders, along with varying functionalities and focuses.

Creative Coding approaches allow to create digital experiences, installations and interactive physical products with different vocations, from recreational and artistic, to discursive and pragmatic. Our teaching framework is intended to be a stepping stone towards a way to extend the reach of Creative Coding to cover the design process without only focusing on artistic expression but also to problem solving related to design as a practice.

3 STRUCTURING A TEACHING FRAMEWORK FOR CREATIVE CODING

3.1 Learning goals

The overall learning goal of the Creative Coding course is to teach design students how to use coding in a design process. This goal is achieved in a process that is three-folded.

The very first step is to introduce students to the idea of procedural thinking [9]. In doing so, the pragmatical aspects related to the creation of communication artefacts are deconstructed and translated into algorithms, namely instructions that computers can execute. Being familiar with procedural thinking is the baseline necessary to turn Creative Coding into an asset for the design process.

Secondly, students are presented with scenarios in which coding plays the role of creative medium. Instead of focusing on designing hypothetical digital artifacts that are deployed by developers, students are faced with the task of creating expressive artifacts directly with code. Coding is framed as a useful asset in various scenarios:

1 *Generative artworks.* Students usually execute static artifacts, like posters or videos, and they design specific versions of them that are curated and finite. Instead, coding can be used to create artworks that respond to randomicity and/or user input. This scenario changes "the designer['s role] from being a performer of tasks to being a conductor, effectively orchestrating the decision-making process of the computer" [21].

2 *Sketching.* By sketching, we mean the idea of rapidly testing and exploring rough ideas to be later refined or scrapped. Code can become a reliable way to quickly test a concept, just like a sketch on paper can be. Through a continuous negotiation between the designer and code [14], sketches can later turn in full-fledged digital products.

3 *Prototyping*. Closely related to sketching, prototyping is the most common scenario in which coding can be used as a creative skill by designers. Just like product designers would produce prototypes of products in general, communication designers have the need to produce prototypes of their interactive products to test and refine them.

Finally, students are introduced to the practice of documentation. This activity helps in de-mystifying the black box approach of code [14] peculiar of top-down practices of development, with a moment of recollection and explanation of the process that enables students to shape awareness of what they produced.

3.2 Content Syllabus

Being design students, the course starts from a practice known and close to them: drawing. This ploy allows to show directly how this practice can be applied to their work and to link coding to something they already know. Design students are indeed trained as practitioners, where design is mainly learnt by doing it. Furthermore, this approach is beneficial because it allows them to see almost immediately the results of their actions.

The parallel with drawing allows us to introduce procedural thinking as a way to formally describe all the actions needed to create a drawing. The first exercise is to list all the actions that they usually would do to reach a specific result using a vector editor software: drawing curves, areas, choosing colors, stroke weights, and ordering all the elements. For each action we identify which functions are available in P5.js and the first assignment is to use them to reproduce a given drawing using only code.

In the second lesson 'for loops' and 'if statements' are introduced to automate repetitive tasks. If in the first lesson the writing of code was a less efficient way to obtain the same result, in the second lesson students can start to understand the advantages of computational approaches to coding.

In the third lesson the scope is further expanded by introducing functions, objects and classes. Students can see that coding itself is also a design activity [14], which requires an evaluation of the goal and of the multiple solutions to achieve it.

In the fourth lesson the expansion process continues under a different angle: providing to students the context of the code they are writing. In the first three lessons, thanks to tools like the P5 web editor, or the P5 plug-in for Atom, students just focused on writing the code and running it. In this lesson is introduced the fact that systems like P5 and Processing are that simple because they hide a huge work already done and the concept of 'library' is introduced: code written by others to make your life easier.

With the fifth and sixth lesson we focus on input devices, and on how to use them as an interface between users and artefacts. In the lessons all the ways to provide signals to the sketch are covered, from traditional interactions (mouse and keyboard) to webcam, microphone, movement sensors.

The seventh and eight lessons further context is provided by showing how the code they are writing can have a different role if used on the client side or on the server side. In this way it is possible to explain them the basic terms and concepts useful to understand how a web application works, and the physical infrastructure enabling them.

The last two lessons are used to deepen specific topics, namely geolocation and 3D graphics. In the lesson related to geolocation the basics of cartographic representation are provided, and how to use them in interactive projects also using the GPS sensor of mobile phones. Finally, while it is out of the scope of the course to deepen 3D graphics, we provide a starting point useful for the student to move autonomously.

Concluding, the lessons are a "zoom out" movement from "drawing with code" to "code and design" on two main axes. The first axis is technological, moving from the single action to the software/hardware infrastructure who enables it. The second is related to the design process, from using code for design to designing code.

3.3 Teaching framework

Inspired by already existing coding courses and practices [21]–[23] the presented framework is based on four main driving principles: a pragmatic and visual technology, hands-on lessons, the use of scoped coding and a group activity.

3.3.1 Pragmatic and visual technology

Working with design students it's important to realize the necessity of a programming library that immediately shows the results of what it's being coded.

The choice fell on web technologies due to communication designers familiarity with the environment and, more specifically, on the JavaScript library P5js. Based on Processing, this library is specifically created to be easily learned and to give immediate feedbacks of what is being coded.

As other frameworks oriented to Creative Coding, it has "a focus on making coding accessible and inclusive for artists, designers, educators, beginners" [23].

With P5 it is extremely simple to describe actions and strategies used to produce a certain output, so it is extremely useful for emphasizing the framing of "drawing with code" that we advance in this teaching framework.

3.3.2 Lectures, students' presentations and hands-on sessions

Lectures are structured to encourage interaction as much as possible. Frontal lessons are used to introduce functions, coding constructs and technicalities of the library. These theoretical instructions are accompanied by live coding sessions that help see in real time the results of what's being coded.

Live coding and hands-on sessions are meant to engage also with students doubts and curiosities about the topic of the lecture. It's also a place where, if they feel comfortable, they can also quickly test their own ideas, knowing that an instructor can be available to help them if they don't understand specific issues.

During lectures, groups of students are also required to present their own research about creative coding as a practice, by compiling a list of case studies that showcase creative use of coding in expressive media. These moments are a way to make ideas circulate in the classroom, by also asking students to identify which kind of technologies were used in the case studies presented.

The case studies are then stored as inspirations. Their collection is part of the effort of contributing to the sentiment of being part of a community of people that share knowledge, insights and help between them.

3.3.3 Scoped coding exercises

Students are asked to test and sediment the notions acquired during the lessons in weekly coding exercises. These assignments are scoped to the concepts introduced in the lesson, and they are evaluated on two criteria: a technical goal and an expressive goal. The technical goal refers to the capability of correctly applying the newly acquired notion in a small project: for example, when accessing the gyroscope and accelerometer of a mobile device, the technical goal would be to correctly calling the related functions. The expressive goal, on the other hand, taps more in the creative part of coding: using the same example as for the first goal, students could be requested to use the gyroscope and accelerometer to recreate digitally a physical object using the device's orientation.

This kind of scoped exercise creates a playground where students don't feel constricted in their assignment, leaving room for exploration of how they can use the same coding principles to achieve different communication goals.

In addition to weekly home assignments, a similar format is applied to an ex-tempore exam that students need to attend for their final grade. This exam, that we call "Hackathon", is a way to directly test skills acquired by students and it is framed around a yearly theme. The evaluation method is similar to the one applied to assignments: a series of technical requirements are enumerated to the students, and a plus is given if students were able to express a personal point of view or a particular concept.

3.3.4 Group project

The course ends with a group project that asks groups of students to tackle a design problem. Every project primarily focuses on a theme given by instructors, that has to be interpreted expressively by the group. The expected outcome is a prototype that presents a clear communication goal and a clear context of use. Indeed, the focus of this activity is to confront students with the need of designing code towards a determined communication goal. Documentation is compulsory and must include descriptions of design process, implementation and technical solutions adopted. This pedagogical effort is fundamental in organizing the knowledge developed in producing the prototype [24] and necessary for exposing the design decisions made during the creative process.

Lastly, this documentation nurtures the circulation of ideas and for this reason it is embedded in the GitHub repositories that will host the source code.

4 RESULTS AND VALIDATION OF THE FRAMEWORK

4.1 Group Projects

We present here a selection of projects done during the four editions of the course, with a brief description and the link to code and prototype.

From 2016 and 2018, the theme provided by instructors had been "Out of scale". With this theme, students were challenged with an experimentation on computer interactions Such interactions had to be designed to explain and nearly create a perception of certain phenomena that are very hard to be experienced, because of their immeasurable dimension or their elusive nature.

This theme was strongly inspired by projects that made a strong use of interaction for explanatory aims, such as “If the Moon were only 1 pixel” [25], “Vax! A game about epidemic prevention” [26] and “Sortie en mer. Drowning simulator” [27]. The reason why instructors picked this theme is that it combines traditional skills required in communication design (like the ability to clearly and efficiently cast a message) with more technical ones.

In 2019 instructors switched to a new theme: “Common Spaces”. Students were required to design and prototype an interactive, digital and collective experience. The theme revolved around the role of the designer as an intermediary: each project had to identify a context of use for a small application that more users would use in real time or asynchronously. Taking into account the simultaneous possibilities of interaction of many individuals, the theme is also a good exercise of procedural thinking.

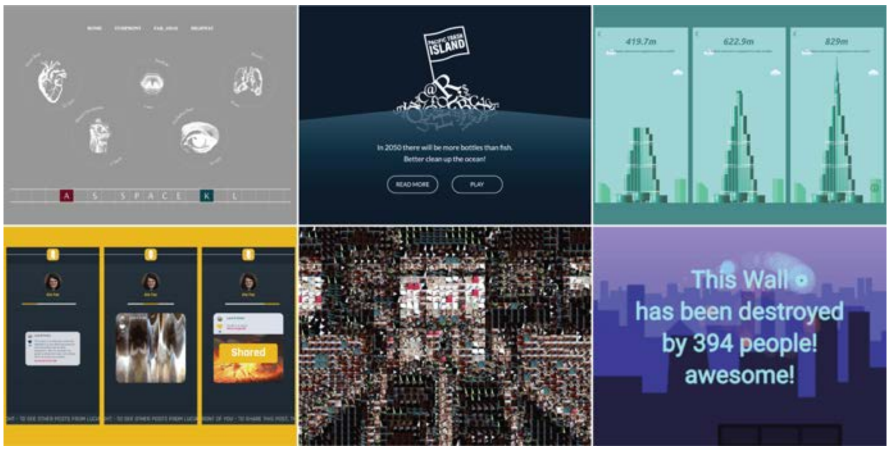

Figure 1. Screen captures from projects, ordered from top-left: 4.1.1 Anybody, 4.1.2 Pacific Thrash Island, 4.1.3 Straight Up, 4.1.4 The Unsocial, 4.1.5 Aletheia Phto Mosaic, 4.1.6 Break the wall

4.1.1 Anybody – Symphony (A.Y. 2016-2017)

Repository: github.com/estestest/anybody

Prototype: estestest.github.io/anybody/SYMPHONIA/index1.html

The interactive experience is a fascinating journey through human body, that challenges the viewer with controlling the rhythms of many unconscious movements that our body performs all at once, like heartbeat, blood circulation, eyelashes beat or breath. How hard would it be to actively pursue all of them? Find it out testing the prototype.

4.1.2 Pacific Thrash Island (A.Y. 2017-2018)

Repository: github.com/9roupnine/PacificTrashIsland

Prototype: 9roupnine.github.io/PacificTrashIsland

“It is estimated that in 2050 there will be more bottles than fish in the Ocean”. How much garbage is that and how hard would it be to clean it up? This interactive experience challenges the viewer with the task of collecting plastic scattered in the sea water. As it happens in reality, the appearance of new waste makes the task even harder, or rather, impossible.

4.1.3 Straight Up (A.Y. 2017-2018)

Repository: github.com/nucleartriad/straightup

Prototype: nucleartriad.github.io/straightup

The human mind perceives horizontal and vertical distances in different ways, indeed “we find a 300-meter tower to be enormously high, although 300-metres along the street is seen as a trifling distance” [28, p. 25]. What happens if vertical becomes horizontal? This prototype gives viewers the possibility to experience heights horizontally and walk all along their actual length taking advantage of the GPS location capabilities of regular smartphones.

4.1.4 The Unsocial (A.Y. 2018-2019)

Repository: github.com/drawwithcode/2018-group-work-group-05

Prototype: drawwithcode.github.io/2018-group-work-group-05

This project is one of the most speculative design outcomes of the course. Would we make the same use of social networks if their interfaces required unusual and strange interactions? Thanks to smartphones sensors, users can experience this weird situation in which they have to jump to like a post, walk to scroll over different posts, loudly dictate to write a status and throw the mobile to share posts.

4.1.5 Aletheia Photo Mosaic (A.Y. 2019-2020)

Repository: github.com/drawwithcode/2019-group-10-1

Prototype: drawwithcode.github.io/2019-group-10-1/

Album launches proved to be an event to be designed and orchestrated. In this example, students worked on how the experience of discovering new music can be shared by individuals through the web. In the prototype, users can take a picture and contribute to a shared mosaic of photos that, in the end, create the album cover of a newly released album.

4.1.6 Break the wall! (A.Y. 2019-2020)

Repository: github.com/drawwithcode/2019-group-08-1

Prototype: ccbreakthewall.herokuapp.com

In this example students worked on the topic of ‘walls’, intended as all of those situations where separations are created among human beings. The Wall of Berlin, the Mexico border wall, the Brexit, the Italian immigration policy issues are among the triggers behind this project. After entering the room, players are separated by a wall and perceive their presence thanks to sound. To break this situation, they have to break the wall by working together, showing that “the more people get separated, the more they want to be together”.

4.1.7 Results

Beside the capability to deploy code in the design process and to overcome technical challenges, by exploring the group projects it appears as self-evident how the adoption of coding resulted in a boost of creativity for designers, given the high level of innovation and originality reached in each exercitation.

All the other group projects and individual assignments are made available on the website drawwithcode.github.io [29].

4.2 Survey

The described outcomes are certainly of use to esteem the teaching framework in terms of pragmatic achievements. However, there are other important aspects that escape this form of evaluation and that are useful to fine-tune the teaching framework in relation to our initial motivations and learning goals.

For the purpose of this paper, we invited students to participate to a survey, asking them open-ended questions. The survey didn’t focus on the methodology discussed in this paper, rather it was structured to let the students reflect on the impact that this course may have on their career.

Questions:

1 Did you have any knowledge about any form of programming before the course?

2 How did you develop your previous knowledge of coding, if any?

3 What are the main challenges you faced during the CC course?

4 Following the Creative Coding course, have you acquired skills useful for a designer?

5 What's the most useful thing you learned during the Creative Coding course?

6 Do you believe that the developed skills influenced your way of designing? How?

7 Did you use the skills acquired during the course in other occasions?

4.2.1 Survey Results

The survey had a total of 54 participants. Most of them had a basic knowledge of coding, but a considerate number of them had no previous knowledge before attending the course. People who were already familiar with code attended other courses in Politecnico di Milano, or in other external courses. Most of them, however, were interested in the field and learned coding by themselves before the course.

From the survey, many participants confirmed that the course was useful to get an idea of how developers work, and that working with technology directly helped them to have a more aware design process:

> *"Now I design an interface with more awareness of the back-end work and probably I could interact better with the developers"*

> *"[…] My projects' scope opened after the course, and I used my acquire skills in various occasions."*

After the course, the large majority of participants said they used knowledge and skills acquired during the course in other instances of their careers, many in university courses and others in a professional context.

Some of the answers highlighted that students incorporated procedural thinking in their workflow, and they realized the potential of using code to automate repetitive tasks and to automatically generate artworks with it. Some of them explained that coding helps them to quickly explore ideas that otherwise would require much more time to implement.

> *"You learn that some repetitive tasks […] can be done just by writing few lines of code […]. This saves you a lot of time and enables you to try more things than before."*

> *"In particular, […] I learned to look for the quickest way to achieve the results I'm looking for."*

Not all the survey's participants, however, found that coding influenced in any particular manner the way they design, either because they already incorporated some form of procedural thinking, or because they did not develop a deeper affinity with the practice.

> *"I've chosen the subject because I already design in kind of an algorithmic way."*

> *"At the moment I think coding is not what I want to focus on for my future career"*

Many participants mentioned how coding opened up new creative and expressive paths. After surmounting the initial "language barrier", students started to exploit the specificity of the coding practice to their advantage.

> *"Before the course JavaScript was like "Arabic" to me."*

> *"It does, it gives you one more (powerful) tool to express your creativity and to realize cool stuff!"*

> *"Unlock my creativity trough coding."*

> *"I tend to think about using the tools learned in this course when I face a design problem."*

However, others were not able to get into the creative aspect of coding, either from a lack of understanding of the concepts introduced in class, or some other reasons. One of the main challenges faced by a participant was *'being creative with code'.* This could be tied to the fast-paced nature of the course, which takes place in just a semester. Or, again:

> *"Understanding the logic, and it was a too fast speed for me. […] I stopped focusing on the coding part and focused more on the idea."*

> *"Learning how to solve coding problems when the things turned more complicated with the final project. This was also because I believe one-semester course is not enough to get deeper in the mind-set of the coding language."*

Moreover, participants found the visual and pragmatic nature of p5.js a good way to start coding, but some of them stated that p5.js introduces a series of abstractions that make it harder to understand.

Presumably, this is due to the high degree of self-teaching that some students had before the course that focused on JavaScript, that is structured in a quite different way from p5.js itself.

> *“Knowing [p5.js] proved to be useful also when dealing with other programming languages”*
>
> *“Having to deal with a new language and its specific rules, which were different from what I was used to. […] I found myself just "copying" the code without knowing what it really meant.”*
>
> *“P5.js has lots of weaknesses that make it unsuitable for professional project.”*

Assignments proved to be a valuable framework to test and sediment skills acquired during the course, but proved to be overwhelming for some participants.

> *“The weekly assignment was time consuming from time to time, but challenging in a good way.”*
>
> *“[The main challenge for me were] the assignments with a brand new concept every week.”*

The practice of documentation highlighted the usefulness of open source resources that can be found online. However, it doesn’t seem to have been contemplated in the survey answers.

> *“If I can't do what I got in my mind I can always look online and find a way to achieve that.”*

5 CONCLUSIONS

In this contribution we presented a framework aimed at teaching “creative coding” to communication design students. Through this framework students are expected to learn procedural thinking, to become conversant about coding and to assimilate it within their design workflows.

The framework relies on the selection of a technology that helps students in easing the learning curve by providing immediate visual feedbacks: p5js. Consequently, the framework presents lectures that alternate groups’ presentations, frontal lessons and live coding sessions. Students can test and experiment with acquired notions in weekly assignments and in a final group activity that gives space to expression, exploration and further learning.

The issued survey shows that the teaching framework is effective in making students conversant about coding, however it also shows that it can be improved in many aspects, such as the frequency of the weekly assignments.

By using free and open materials, the presented framework can be easily replicated, allowing any design instructor to teach Creative Coding. In its replicability, the framework is flexible and scalable: instructors have the possibility to shorten or to extend the syllabus depending on specific contexts and they can address with custom themes its overall goal, that is to teach design practitioners how to use coding within their process, with a problem-solving intent.

The presented framework provides students and instructors with the basic knowledge to achieve such high goal. The work presented in this contribution represents a first step to understand how coding gets assimilated by designers and how it gets pragmatically deployed within their workflows. It also offers a setting in which it is possible to do research on this specific aspect of design education and practice.

ACKNOWLEDGEMENTS

Authors would like to thank all the students who participated to the survey and to the Creative Coding course, as well as the School of Design at Politecnico di Milano for the didactic opportunity.

REFERENCES

[1] itsnicethat.com, “Three creatives tell us why coding is crucial to contemporary design,” Jan. 22, 2018.

[2] A. Benedetti, “Generative×Data-driven Posters,” *DensityDesign*, Mar. 17, 2020. https://densitydesign.org/2020/03/openrndr-generative-posters/ (accessed May 05, 2020).

[3] A. Shamir, “Feature-Based Design of Fonts Using Constraints,” *Electron. Publ. Artist. Imaging Digit. Typogr.*, vol. 1375, Dec. 1998, doi: 10.1007/BFb0053265.

[4] F. Guida, “Generative Visual Identities. New Scenarios in Corporate Identity.,” 2014, doi: 10.13140/2.1.1200.0329.

[5] J. Matthews, “Netflix Review: Bandersnatch.” Jan. 18, 2019, [Online]. Available: https://digitalcollections.dordt.edu/faculty_work/1031.

[6] N. Elnahla, “Black Mirror: Bandersnatch and how Netflix manipulates us, the new gods,” *Consum. Mark. Cult.*, pp. 1–6, Aug. 2019, doi: 10.1080/10253866.2019.1653288.

[7] J. Walny *et al.*, *Data Changes Everything: Challenges and Opportunities in Data Visualization Design Handoff*. 2019.

[8] S. F. Larsen, “Procedural Thinking, Programming, and Computer Use,” in *Intelligent Decision Support in Process Environments*, vol. 21, E. Hollnagel, G. Mancini, and D. D. Woods, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 145–150.

[9] M. Mateas, “Procedural Literacy: Educating the New Media Practitioner,” in *Beyond Fun: Serious Games and Media*, Pittsburgh, PA, USA: ETC Press, 2008, pp. 67–83.

[10] C. Reas, C. McWilliams, J. Barendse, and N. LUST (Hague, *Form+code in design, art, and architecture*. New York: Princeton Architectural Press, 2010.

[11] S. Heller and V. Vienne, *Becoming a Graphic & Digital Designer. A Guide to Careers in Design*, 5th ed. Wiley, 2015.

[12] C. Reas and B. Fry, “Processing: programming for the media arts,” *AI Soc.*, vol. 20, no. 4, pp. 526–538, May 2006, doi: 10.1007/s00146-006-0050-9.

[13] J. Maeda, *Creative code*. New York, N.Y: Thames & Hudson, 2004.

[14] S. Turkle and S. Papert, “Epistemological Pluralism: Styles and Voices within the Computer Culture,” p. 32, 1990.

[15] L. Manovich, *Software culture*. Milano: Olivares, 2010.

[16] J. Maeda, *How to Speak Machine. Computational Thinking for the Rest of Us*. Portfolio, 2019.

[17] Joreg, M. Wolf, S. Gregor, and S. Oschatz, *VVVV*. vvvv group.

[18] Z. Lieberman, T. Watson, and A. Castro, *OpenFrameworks*. 2018.

[19] H. Bohnacker, B. Gross, J. Laub, C. Lazzeroni, and M. Frohling, *Generative design: visualize, program, and create with JavaScript in p5.js*. New York: Princeton Architectural Press, 2018.

[20] D. Shiffman, “The Coding Train,” *The Coding Train*. https://thecodingtrain.com/ (accessed May 02, 2020).

[21] L. McCarthy, C. Reas, and B. Fry, *Make: Getting started with p5.js ; making interactive graphics in JavaScript and Processing*. San Francisco, CA: Maker Media, 2016.

[22] R. S. Geiger, N. Varoquaux, C. Mazel-Cabasse, and C. Holdgraf, “The Types, Roles, and Practices of Documentation in Data Analytics Open Source Software Libraries: A Collaborative Ethnography of Documentation Work,” *Comput. Support. Coop. Work CSCW*, vol. 27, no. 3–6, pp. 767–802, Dec. 2018, doi: 10.1007/s10606-018-9333-1.

[23] J. Worth, “If the Moon were only 1 pixel. A tediusly accurate scale model of the Solar System,” *https://joshworth.com/*, 2014. https://joshworth.com/dev/pixelspace/pixelspace_solarsystem.html (accessed May 01, 2020).

[24] Salathé Group, “Vax! A game about epidemic prevention,” 2014. https://vax.herokuapp.com/ (accessed May 01, 2020).

[25] G. Cotten, “Sortie en mer,” *Sortie en mer*, 2014. https://www.youtube.com/watch?v=0Uu_N1c7E2M&feature=emb_title.

[26] A. Frutiger, *Signs and symbols: their design and meaning*. New York: Van Nostrand Reinhold, 1989.

[27] M. Mauri, T. Elli, and A. Benedetti, “Draw with Code,” *Draw With Code*. https://drawwithcode.github.io/ (accessed May 02, 2020).