How Open is Coding Pedagogy Really?

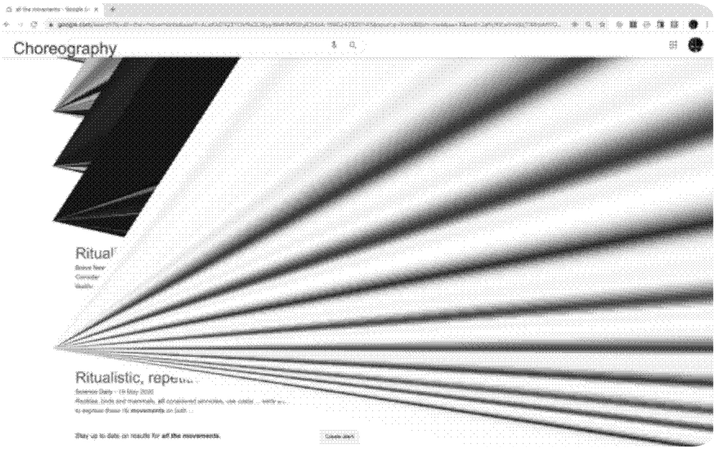
As previously covered in my findings, creative coding emerges as a pure and software-independent form of expression, operating as an open medium with additionally free learning hubs. While preparing designers for technical intricacies, the challenge lies in the lack of manual involvement and the prevailing IT stigma, hindering broader appeal. Addressing these concerns is crucial, alongside evaluating the accessibility and learnability of creative coding wisdom. How readily can designers adopt these methodologies to achieve greater technical autonomy?

Stig Møller Hansen, a Graphic Designer and Senior Associate Professor at DMJX, delves into the effective integration of programming into graphic design education in "Public Class Graphic_Design Implements Code { } // Yes, But How?" (140)*. Through a mixed-methods approach, Hansen identifies gaps in current practices, highlights learning style differences, and lastly, proposes a hands-on pedagogic method for a more effective assimilation of programming in graphic design education.*

Graphic Design Implements Code — Yes, But How?

Code as a creative medium and material holds a vast and yet barely explored potential and must be included in the disciplinary epistemology if graphic design is to stay current and relevant. As programming and graphic design continue their convergence and form an even stronger symbiotic relationship, it naturally calls for computation to be integrated into graphic designers' thinking and doing. By learning to program, graphic designers can become creators of unique and custom-made digital design tools and not just users of ready-made, industry-standard tools provided by the software industry. By learning to program, graphic designers can reap the benefits of adding computation to all stages of their workflow. By learning to program, graphic designers can acquire a new perspective and appreciation of their trade as they explicate their tacit knowledge. By learning to program, graphic designers can understand how software works and adapt the way they conceptualize their designs to fit the computational media. By learning to program, graphic designers can push the disciplinary boundaries forward as they infuse the essence of their trade with the power of technology. Finally, and perhaps most importantly, by learning to program, graphic designers can also learn when programming is not the answer to their needs and their computer is better left switched off. *(Hansen, 2019, p.107.)*

Programming should ideally be taught to graphic designers as a studio-based social activity practiced in a safe environment using relatable metaphors, familiar materials, visual examples, and live, interactive demonstrations. Courses should be planned utilizing a design-first perspective to allow the graphic design students to use their pre-existing, domain-specific skills to leverage their acquisition of programming knowledge and relate it to their vocational practice. Ideally, flipped classroom and blended learning instructional strategies should be used advantageously to move transferable knowledge out of the classroom, allowing instead for contact time being used to engage in plenary discussions, presentations, troubleshooting, and transfer of tacit knowledge through experiential, hands-on discovery learning. Assignments should be available in varying difficulties, solvable individually or in pairs/groups, provide a fixed goal, include premade assets, and limit the need for aesthetic decisions to ensure students stay focused on learning to program. *(Hansen, 2019, p.90.)*



(Chicau, Joana: Act 2 — live coding browser Javascript)

Screenshot of Joana Chicau performing live coded visuals on web console browser, writing in JavaScript naming functions after a choreographic vocabulary

How Should Creative Coding Be Taught?

A code-first approach is used by most design educators to inform and guide the planning of their Creative Coding courses. This entails course structures originally intended for a nondesign audience being transferred to graphic design students in a form that has only been moderately adapted. I was interested in investigating whether there were any notable differences in the learning style profile of graphic design students and students within technical fields. A quantitative analysis of Felder-Soloman Learning Style Index (ILS®) *(141)* questionnaires, completed by 77 graphic design students, showed that their learning style profile differed noticeably from that of students in technical disciplines. This implicitly underlines the need to develop customized programming courses and accompanying instructional methods for use in design schools. In response, the following specific recommendations were made:

- Teaching materials, demonstrations, and assignments must be highly visual.
- or metaphors tied to the graphic design students' pre-existing, domain-specific knowledge must be used to help them develop mental models of abstract programming constructs
- Live coding and visual demonstrations should be used whenever possible.
- Exercises and assignments should be designed to have a fixed goal but allow for different ways of arriving at a solution either through experimentation or through stepwise instructions.

- Educators must be careful to provide the big picture and relate it to the students' previous knowledge before going into details.
- Teaching must actively engage the students; however, initiatives that call for reflective activities should also be integrated.
- Students should be given the option to work either in pairs or alone, depending on their personal preference.
- Students should be given objects to assist their exploratory discovery; this might be visual specimens, premade code snippets, or a set of digital assets to use.
- Providing additional explanatory tutorials and demonstrations, preferably in the form of video/animations, is highly encouraged.
- Other suggested activities to support graphic designers in building cognitive models of programming constructs are verbal walk-throughs of algorithms, tests that involve math and logic, quizzes, and code-related puzzles.

The hypothesis for this specific research question was that contemporary Creative Coding courses aimed at graphic designers are not taught in the most optimal way and do not account for how they learn. I consider this hypothesis to be (partially) supported. *(Hansen, 2019, pp. 88–89.)*

Tim Roedenbroeker — Interview with PAGE Online

Tim Rodenbröker (142)*, whom I've also previously mentioned in my findings, serves as a prime example of accessible creative coding pedagogy "done right", through his significant contributions to the blogosphere of various designers enganged in coding processes. The German graphic designer turned creative coder launched his e-learning platform* (143) *in 2020, after years of teaching his methodologies at various universities. An interview conducted by Lena Simonis, on behalf of Page Online* (144)*, effectively captures the essence of his teaching philosophy.*

You are currently developing a learning platform for processing and creative coding, how did this come about?

TR

There are a number of reasons for this. Code is the material from which innovations are made. Designers benefit enormously from understanding how digital products work under the hood. Furthermore, learning to program means training systemic thinking and that is a superpower! So programming not only helps with the development of digital products, it also trains the human brain in an efficient way.

How do you prepare the courses?

TR

Each course tells its own little story. I don't want to give too much away yet, but my aim is to pick up the users emotionally. For each lesson, there is a video lasting around five minutes, a detailed text with code examples and additional download links and references to internal and external tutorials if you want to delve deeper into a topic.

What thoughts have you given to the learning concept?

TR

The first and fundamental task in developing such a program is to answer the question of "why". Why should a communication designer deal with this topic? My thoughts have been revolving around this question since the beginning. That's how I find out what's really relevant. And that is then the foundation stone for a course. All courses are based on a curriculum that starts from scratch. Pure designers are best advised to start with the basic course, while people from the world of programming can go straight into courses with a design focus. "Programming Posters" *(145)*, for example. The course reveals a whole range of tricks, not only about programming, but also about creativity and aesthetics. There's a piece of paper next to me with eleven course ideas on it, all of which expand the curriculum. That's what I want to do over the next few years.

Are you also planning a community, forums, chats? How does the feedback channel work?

TR

I have already founded or manage a few Facebook groups; for example the Generative Design Research Network *(146)* with over 4000 members, or Crazy Cool Developers *(147)* for people who are interested in the web, user interfaces and brutalism. We also organize Creative Coding Days *(148)*, hackathons where we meet for a week with ten to 15 people from different parts of Europe at a Dutch artist's country house and everyone codes something and exchanges ideas. There should also be a forum for all questions about my courses. I'm still thinking about the best place to host it. Maybe it will also be a Slack channel. We'll see.

What exactly are the first two courses about?

TR

The first course is all about creative coding essentials *(149)* and is very broadly based. It's about the basics of programming for creative people. The second course is called "Programming Posters" *(150)*, which is all about developing ideas for interactive posters in a very applied way. Historically, the poster has gone through three media phases: first printed, then animated and now interactive - in other words, a completely new medium. The course teaches you to scrutinize the potential of the various media and to see what you can do with them if you involve programming.

Where do you see the potential of Processing in media design?

TR

I would ask the question slightly differently, because Processing is only a very rudimentary tool. But if you ask how generative design can be used in the development of (print) media, I see endless potential: design automation means that layouts can suddenly be developed based on data sets. But tools can also be developed that are suitable for designing any type of medium. The possibilities are so great that they give me a headache.

So is Processing the solution to all our problems or does it also have shortcomings?

TR

No, absolutely not. You have to look at it this way: Processing is just a relatively archaic tool. It forces you to think things through and question them. Creative coding is a school of thought and Processing is a suitable tool for solving the tasks of this school of thought. There are designers and agencies that use Processing to build large enterprise-level apps, scalable design tools for big brands. I also do something like this on a small scale for my clients from time to time, but it has to be said quite clearly that the development of an extensive app takes an extremely long time and many supposedly obvious things cannot be implemented. This requires excellent project management and even better communication between all stakeholders. This is extremely dangerous as a freelancer! Processing projects can escalate completely. The scalability of Processing apps is only feasible with enormous effort and a great deal of know-how.

On the other hand, Processing is very suitable for small projects! Be it a design tool for record covers, playful data visualizations or generative 3D graphics. The sometimes random results from a Processing project are perfect for more experimental, open customer projects.

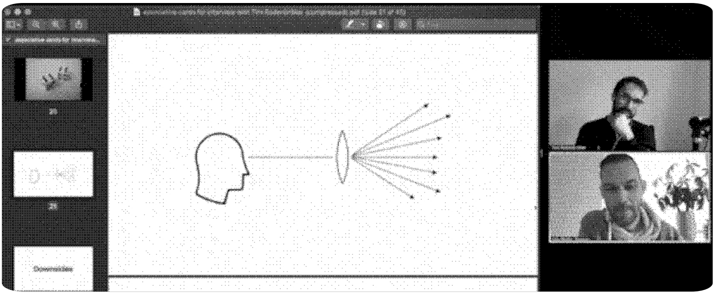
You gave a presentation at ADC Düsseldorf at the end of November 2019 about your "Programming Posters" course. What was the core message?

TR

I talked about the communication between designers and developers. The lack of equality and the lack of competence on both sides. As I mentioned above, I believe that we need a new type of designer who works at the intersection of design and technology and bridges the gap. I call this type the "creative technologist". Basically, it was about letting go of outdated principles in the training of designers. I would like to see people approach each other in order to work together more effectively. And that more universities would teach how to survive as a designer in an increasingly digital world.

(Zeta, Nahuel: Flor de Fuego at a live coding visual performance in La Plata, Argentina)

A live coding visual performance using Hydra by Flor de Fuego in La Plata, Argentina



(Rodenbröker, Tim: A conversation with Stig Mǿller Hansen)

Tim Rodenbröker and Stig Mǿller Hansen talking about creative coding teaching