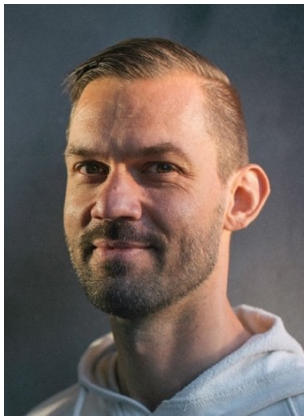
Building a Digital Textbook around Video Worked Examples for Intermediate Level Programming Courses in Design Schools (Work in Progress)

Assoc. Prof. Stig Møller Hansen, PhD
Center for Design & Visual Communication
Danish School of Media and Journalism, Copenhagen, Denmark
www.stigmollerhansen.dk
e-mail: smh@dmjx.dk



Abstract

This work-in-progress paper reports on the preliminary research that has informed the development of an interactive digital textbook for use in intermediate level programming courses in design schools.

The audience of the textbook is design students with prior programming experience and familiarity with basic programming concepts, subsidiarily design school educators who will be teaching programming.

In its current form, the textbook consists of projects presented as Video Worked Examples with subgoal labeling and supplementary activities that support an increasing progression in learning levels.

This paper contributes useful insights on decision-making processes and the choice of theoretical and methodological foundations for the textbook. These insights can be useful to others who wish to fill the shortage that currently exists in the availability of educational materials for design students with intermediate level programming skills.

Key Words

design education, programming, instructional design, digital textbooks

1. Introduction

Generation Zers - people born after 1996 - are now entering design school classrooms. As true digital natives, their expectations of educational materials are very different from previous generations. Studies [1, 27, 32] show that Gen Zers often don't read traditional textbooks, avoid buying them, and consider them boring and

acquire new knowledge from on-demand learning videos [24], and expect digital textbooks to include interactive activities such as quizzes, puzzles, dynamic visualizations, and problem-solving activities.

Design students acquire tacit knowledge through studio-based learning and verbal guidance, but must typically also acquire the design discipline's foundational and core explicit knowledge through self-study; an activity where textbooks have traditionally been a key pedagogical resource. However, technological advances coupled with Gen Z students' preferred learning methods - further accelerated by the recent COVID-19 pandemic's shift to remote teaching - have seen many design schools abandon traditional textbooks (paper-based physical books) in favour of so-called *videobooks* (video-based online books), a format that Granitz et al. [12] estimates, will likely become more common.

Code has long proven its potential as an expressive, malleable medium, and programming courses are increasingly to be found in design school curricula. These courses prepare design students for a future job market where their ability to work with design as the result of computational processes becomes an essential part of their occupational toolset, skillset, knowledge set and mindset. Programming courses are requested by design students, their educators, but also by the design industry, which increasingly sees clients asking for code-driven design products.

While there is no shortage of instructional materials dealing with code and art, few of these address the trade-specific issues designers must adhere to. In their search for instructional materials that both cater to Gen Zers' preferences for video-based interactive content, and focus on teaching programming adapted for use in a design context, design school educators currently look in vain. Some capable educators produce their own proprietary programming courseware, but the fruits of their time-consuming labour rarely reach beyond their own school to benefit other educators who, for a variety of reasons, are unable to make their own teaching material.

Collectively, the scenarios described above raise the research question driving this paper: *How can instructional materials for intermediate level programming courses in design schools be designed to help improve motivation, engagement, knowledge transfer and retention, and overall experience for a generation of design students who have a strained relationship with traditional printed textbooks?*

2. Initial Considerations

2.1 Topic

The textbook's topic is programming, specifically programming in a visual context. In this field, a large number of frameworks and programming languages exist, e.g. Processing, p5, openFrameworks, Cinder, Nodes, OpenRNDR, Nannou, thi.ng, and TouchDesigner to name few. The textbook will use Processing, a robust, well-documented and widely used Java-based programming environment, developed specifically with the textbook's audience (see section 2.2) in mind. Processing adheres to the philosophy of "low threshold, high ceiling, wide walls" [26], and its clutter-free interface and dependency-free environment make it ideal for an audience of informal non-CS programmers. A drawback of Processing is that it cannot be embedded in a browser-based textbook. To mitigate this, the textbook relies on p5.js, a javascript port of Processing, to embed interactive versions of the textbook's projects.

2.2 Audience

The primary audience for the textbook is students at design schools. More specifically, design schools with programs focused on visual communication, e.g. Commercial Arts, Graphic Design, and Visual Communication. The secondary audience is educators at these schools. Finally, the textbook is also relevant for self-taught designers, graduate designers wishing to further their education, young people training to apply to design studies and hobby designers.

An important distinction to make is that designers are not artists. The textbook aims to fill the gap in a market where there are numerous titles focused on making art through code, but all neglect to address the many issues of the design trade, which the artist does not need to worry about: consistency, identity, homogeneity, identification, staying on brand, conveying the right message, legibility, functionality, applicability, reproducibility, flexibility, adaptability, technical limitations, and client feedback [15].

2.3 Aim

The textbook is not an introductory course to programming. Instead, the textbook aims to show design students with a basic understanding of programming how to apply this knowledge in solving design domain-specific problems.

Introductory programming courses focus on teaching basic programmatic concepts, and rarely discuss the output's further use. The textbook deliberately aims to raise the focus on the applicability and usefulness of the output.

The overall goal of the textbook is to act as an effective *learning object* described by Kay & Knaack [16] as: *"[...] an interactive web-based tool designed to enhance, amplify and guide learning, [...] a readily accessible, easy-to-learn, concept-focussed tool [...]. Good learning objects a require students to construct and manipulate information, provide rich feedback and interactive illustrations, help students understand abstract ideas with concrete representations, and support important student weaknesses such as limited working memory, difficulty in retrieving long term memory, and ineffective learning strategies."*

2.4 Level

There is no shortage of learning resources covering general programming for beginners. However, once design students have mastered basic programming concepts, many report a lack of intermediate-level learning materials. This shortage becomes even more pronounced as they look for learning resources that focus on design-specific problems. This implies a gap between learning materials aimed at all-purpose beginners and subject-specific experienced learners. The textbook aim to situate itself in this sparsely occupied space.

Consequently, the textbook assumes prior knowledge of basic programmatic concepts, which the student must acquire through other resources. This is pointed out in the textbook's preface as well as in the list of prerequisites. The textbook's list of resources also suggests learning materials that are suitable for creating the knowledge base needed to work on the textbook's projects.

2.5 Tone of Voice

The textbook is aimed at design students who are generally unfamiliar with the tools, platforms and terminology used by professional developers. When designers follow programming tutorials made by

"clone a Github repo," "open a terminal," or "install with npm" distract and impede their motivation to learn. To meet design students at eye level, the textbook takes its cue from design students' design-centric conceptual world and IT skill level, and use informal, non-technical language. For example, code examples will be provided as packaged zip archives rather than repo's on GitHub, use of the terminal is reduced to an absolute minimum, and installation of third-party libraries and tools is primarily done through Processing's graphical interface.

2.6 Technical Platform

A number of actors (e.g. Runestone, OpenDSA, zyBooks) offer tools that enable educators to create interactive digital textbooks. However, these are predominantly aimed at use in computer science and STEM education. Recently, the concept of *computational notebooks* (e.g. Jupyter, RNotebook, Observable) has gained traction [28]. These are documents that can be read like a regular book chapter and executed like a computer program. While such notebooks are very flexible and could be tailored for use in design courses, their novelty and unfamiliar structural paradigm often confuses students, causing them to spend an inappropriate amount of time just understanding the structure and operation of the notebook.

Other services exist that make it easy to develop tutorials in an environment specifically tailored to designers. One such example is OpenProcessing, which allows step-by-step tutorials to be written and potentially linked to courses also hosted on the platform. Despite its many features, OpenProcessing does not have the flexibility to enable the implementation of the textbook as desired.

To deliberately avoid encapsulating the textbook's content in proprietary systems and to circumvent their inherent limitations in terms of layout and functionality, the textbook is instead based on widely available open web technologies.

2.7 Publishing

As pointed out in this paper's introduction, Gen Zers do not find printed books appealing og engaging. Their preference for on-demand video-based learning material and their expectation of being able to interact with the content make a compelling argument for publishing the textbook on a digital medium. Furthermore, given that the topic of the textbook is programming, it seems logical to publish it on a platform that supports the insertion and execution of code directly into the body of text. Finally, given the pace of which technologies are changing, printed books containing code listings are virtually outdated by the time they are published.

Publishing the book through a publisher will entail a transfer of ownership and rights. This will seriously hinder or block the intended audience's access to the textbook. Hence, the textbook will be a self-published, online-based, continuously updated and expanded open work that can be accessed for free and shared without financial or copyright restrictions imposed by third parties.

2.8 Structuring Approach

Writing a textbook on programming for designers requires a constant interdisciplinary negotiation between two distinct disciplinary fields (visual design and programming, respectively) and a balancing of the inherent dichotomy present in their individual didactic concerns for learning progression.

The overall approach to selecting and structuring the textbook content is *design-first* rather than *code-first* [14]. This approach favors a focus on exploratory generation of visual design through code, and deemphasizes the importance of "correct" coding practices; programming is understood a means rather than an end in itself.

Further, emphasis is placed on relating the programming activity to the design students' subject area. A driving force behind the textbook has been the intention that it should appear as a book about programming written *by* designers *for* designers.

2.9 Material

Due to the pedagogical approach chosen for the textbook (see section 3.1), the main material consists of pre-existing commercial products in the field of visual communication. Products have been chosen that are particularly suitable for the deconstruction/reconstruction method, as they exhibit some kind of underlying structure that can be reconstructed and rebuilt.

A key criterion in curating the material has been to illustrate how visual communication products throughout history have also applied systemic and procedural principles. By deliberately tracing a direct line from the works of mid-century designers to today's generation of *creative coders*, the point is made that the use of algorithms (concrete or abstract) to create visual communication is neither new nor dependent on the use of computers. For this reason, priority is given to describing the background and context of each project. Primarily to connect and anchor it in the design student's studies, but also with the expectation that a backstory will contribute positively to raising motivation and engagement associated to completing the project.

Equally important has been that the collective range of material chosen for the textbook reflects the paradigm shift from analog to digital media and the consequent shift in emerging visual aesthetics only explorable through code. This involves techniques like glitch art, ASCII art, cellular automata, emergence, L-systems, fractals, self- organizing systems, evolutionary design, and drawing using data feeds. A study by Hansen [14] highlighted how these techniques were often absent in design school programming courses, and this textbook seeks to remedy that.

3. Theoretical and Methodological Foundation

The textbook's theoretical and methodological foundation incorporates and interweaves a wide range of overlapping theories.

3.1 Deconstruction/reconstruction

As its pedagogical method, the textbook employs an adapted version of deconstruction/reconstruction [13]. This method is based is based on constructionist learning theory, which prescribes a focus on experiential learning, and the use of familiar objects as *objects-to-think* with [23].

Pre-existing visual design product are used as starting points. Their structure and underlying design principles are deconstructed, formalized, and then recreated using code. This approach invokes the use of computational thinking principles and lends itself well to the use of both worked examples, sub-goals and dissemination through video.

3.2 Computational Thinking (CT)

Computational Thinking [5], a concept coined by Papert [23], later reintroduced and popularized by Wing [33], is a

problem-solving technique that aims to formulate the solution to a given problem in a way that can be carried out by a computer. According to Csizmadia et al. [7] the characteristics that define computational thinking are Algorithmic thinking, Decomposition, Generalisation (Patterns), Abstraction, and Evaluation.

Although the textbook only explicitly mentions Computational Thinking in the educator's guide, its approaches are used to complete the textbook's projects. This provides students with the opportunity to develop computational literacy and apply it to their own subject domain. The solutions proposed in the projects' Worked Examples are all generalisations/abstractions that can be (re)used to produce a multitude of variations of the original project.

3.3 Cognitive Load Theory (CLT)

Cognitive Load Theory [6, 30] deals with the way we process information. CLT describes how information is held in our working memory until it has been sufficiently processed, after which it is transferred to our long-term memory. The capacity of our working memory is very limited and when too much information is presented at once, it becomes overwhelmed resulting in a loss of information.

Considering the ease by which attention-grabbing elements can be added to a digital textbook, it is easy to inadvertently exceed the processing capacity of a student's sensory and working memory; a situation referred to by Mayer & Moreno as *cognitive overload* [20]. Informing and influencing the design of the textbook are Mayer & Moreno's five types of overload scenarios in multimedia instruction, and their accompanying methods for reducing the cognitive load.

3.4 Worked Examples (WE)

One way to reduce learners' cognitive load is by using Worked Examples. Worked Examples is an instructional method that provides an "expert" solution as a blueprint for solving a specific problem that the learner can read, understand and adapt.

Learners who encounter Worked Examples in the context of practice-related problems *"are more likely to develop and assimilate strategies"* for solving similar problems [3]. Known drawbacks of Worked Examples is that learners tend to focus on incidental rather than fundamental features, and that Worked Examples do not inherently promote deep processing of concepts [9].

The literature points to a distinction between *process-oriented* and *product-oriented* Worked Examples [29]. Process-oriented Worked Examples show how a specific solution was reached, while product-oriented Worked Examples show one possible solution. Process-oriented Worked Examples have been shown to be best suited for novice learners, while intermediate learners benefit more from product-oriented Worked Examples from which they can infer the rationale [11].

Taking into account the textbook's aim to address intermediate learners, a product-oriented use of Worked Examples has been applied. This approach, which does not focus on the repetitive production of one particular solution, resonates with the textbook's desire to allow the projects to act as springboards for the learner's own interpretations and developments.

3.5 Sub-goal Labeling

Studies [8, 19, 22] have found that student's problem-solving performance of Worked Examples can be improved

through the use of sub-goal labels. A strategy predominantly used in STEM fields, sub-goal labels use action-based instructional phrases (e.g. "declare variables" or "calculate the sum") to help learner's deconstruct problems, recognize the structure of a procedure, mentally organize information, and self-explain the examples [19]. In programming exercises, sub-goals are typically be added using comments, but thoughtful considerations of both their placement and wording are citical for their effectiveness.

3.6 Video Tutorials

Today, video-based tutorials are widely recognised as a powerful pedagogical tool in online teaching activities [31] and favoured over traditional textbooks by Gen Zers [24]. Video tutorials present knowledge in an attractive and engaging way, stimulate learners' attention, motivating them to participate and help improving learning outcomes [34].

Most recently, the shift towards flipped classrooms and the COVID-19 pandemic has sparked the creation of a host of new courses in which Worked Examples are supported by video. This relatively unstudied constellation requires more investigation, but educators report enthusiasm for the format among students [17]. In the absence of guidelines specifically addressing the production of videos for worked examples, the textbook instead draws on eight principles of video production for software training proposed by van der Meij & van der Meij [21]. However, their seventh principle, "keep the videos short," has been purposely overridden. A survey conducted among the textbook's target audience, revealed that 51% preferred videos of 30-45 minutes duration with a detailed review of code written from scratch with each step and decision explained verbally. Timeline navigation of a video's segments is offered in the form of links that carry the same labels as the project's sub-goals.

3.7 Animations and Interactive Figures

Where video captures images of the external world, animations are the product of deliberate construction processes such as drawing. Animation's ability to show temporal change directly and explicitly holds great educational potential, but used incorrectly, animations can be directly inhibitory to learning and concentration and lead to cognitive overload (see section 3.3). Lowe and Schnotz's *Animation Processing Model* [18] suggests five principles for designing animations in educational materials. These principles guide the production of animations for the textbook.

Interactive figures are also used throughout the textbook. These differ from animations by being non-linear. An example of such an interactive figure is a unit circle, where the learner can explore the relationship between sine and cosine. Each project's Worked Example is also embedded as a prominent interactive figure. Initially with the purpose of enticing students to engage with the project, but subsequently also to act as a teaching vehicle that visually reveals the behaviour of the Worked Example.

3.8 Active Code Widgets

Active code widgets are interactive components on a web page. Generally, they appear as a split view containing a code editor and an output area. Pre-entered code is usually present by default in the code editor, where it can be manipulated, executed and reloaded. Although popular among students and found in many digital textbooks and tutorials on programming, there is still insufficient research on the educational value of active code widgets.

A study by Ericson et al. [10] on the use of active code widgets among CS students saw more execution of code than editing. It can be speculated that the willingness to edit code will be even lower among design students.

Another important factor to consider is the risk of causing syntax errors, which may potentially discourage students from engaging with the active code widgets, demotivate them or even impede their learning.

Complicating the use of active code widgets in the textbook, furthermore, is the fact that there are no browser-based active code widgets for the Processing language. Although there are several p5.js widgets, the subtle yet present nuances between the two languages are thought to increase learners' cognitive load.

At the time of writing, it has not been decided whether active code widgets should be included in the textbook, and the advantages/disadvantages of doing so need to be further investigated.

4. Structure

The textbook consists of three sections: Preface, Projects, and Postscript.

Preface describes the motivation for the book, its target audience, learning outcomes, and necessary prerequisites for using the book. Also included is a reading guide that describes the textbook's structure, project structure, instructional elements (e.g. video worked examples, interactive illustrations, code widgets, galleries, assignments etc), and formatting of the texts' various content elements.

Postscript include glossary, author biography, colophon and a large collection of links to related resources. A dedicated section also includes a series of action-oriented heuristics [15] specifically directed at educators who teach programming to design students.

4.1 Project Structure

In an effort to reduce student cognitive load, all textbook projects share the same structure. This establishes a homogeneous, familiar and consistent reading experience.

The project structure is based on the five learning levels of Petersen's Psychomotoric taxonomy [25]: observe, imitate, train, adapt, and design. Petersen's taxonomy is particularly useful for describing learning levels in practical subjects fields, as opposed to Bloom's revised taxonomy [2] which is typically used in the humanities, and Bigg's SOLO taxonomy [4] which is used in STEM fields.

Table 1 describes the prototype structure of a project. Sections are listed as they appear on the page (top to bottom). Note how the activities move from lowest to highest psychomotor level.

5. Further Work

At the time of writing, the textbook is still under development. At this stage, no conclusions can therefore be drawn on the research question posed at the beginning of the paper.

A number of pre-launch activities are planned to qualify the textbook once there is a sufficient mass of content for testing. A workshop with 6-10 design students will test, discuss and evaluate the proposed structure of the project pages (see section 4.1). An observational test of the textbook applied to a class in a simulated teaching situation followed by a plenary interview with the students will provide insight into

Section	Content	Psychomotoric level
Title	Project title	-
Teaser	Short text that explains the gist of the project	-
Project Info	Visual representation + facts (title, client, designer, product type, year)	-
What we'll be making	Interactive p5.js implementation of the Processing project. It serves to draw the student in, but is also an exploratory learning tool that can help students visually understand the interplay between the interactive example and the constituents of the underlying code that drives it.	Observe
Video Worked Example	Video screencast of live coding session in which the project is built from scratch. The project's sub-goals are listed as clickable timecodes to provide easy navigation of the video.	Observe
Download Project Files	Downloads Processing project (.pdez) with complete annotated and sub-goal labeled Worked Example + associated assets and libraries to the student's local computer. Depending on their browser configuration, the project will automatically open in Processing when the download is complete.	Imitate
Textual Description	A brief summation of the activities in the video. The wording of all paragraph headlines matches the project's sub-goals. All paragraph headlines contain a "watch in video" link that leads to the video's explanation of the chosen sub-goal. The textual description is enhanced with animations, interactive figures, executable code listings, and other non-static supporting content.	Imitate
Source Code Listing	Listing of the project's full and final source code with annotations and sub-goals still present in the code.	-
Variations	A gallery exhibiting the visual diversity of products made with the project's code. Serves mainly to motivate and draw the student in.	(Observe)
Explore & Expand	Suggestions for further development of the final project code.	Train
Recreate These	Similar examples using the demonstrated principle, but with a different visual appearance.	Train, Adapt
Assignment	Fictional design brief for an imaginary client posed as a conventional design brief. Its purpose is to allow the student to practice unaided application of the programmatic principle demonstrated in the project. The student can individually undertake the assignment or it can be set as a plenary assignment by the educator.	Adapt, Design
Related Materials	Links to additional relevant resources (e.g., websites, tutorials, videos, books, podcasts, code repositories, papers)	-

Figure 1: Schematic overview of project pages' sections

their actual and perceived use of the textbook.

Following adjustments based on the collected data, the book containing a partial subset of the planned 111 projects will be made publicly available mid-2022 at programmingforgraphicdesigners.com. Completion of the textbook's planned 111 projects is currently scheduled for the end of 2023. Hereafter the books is expected to be further extended and more projects added.

Post-launch, the use of the textbook will be tracked and recorded in log files, from which a number of metrics can be extracted and contribute valuable insights that will be continuously used to improve the textbook. Later, experiences on the textbook as a teaching tool will be collected from design school educators who have used it in their courses.

Once the textbook has seen wider use, a more comprehensive analysis of its effectiveness as an instructional material can be made by examining the performance metrics mentioned in this paper's research question, obtained by students in courses where the textbook has been used, compared to similar prior courses featuring other instructional materials.

References

[1] 65% of Students Skip Required Textbook Purchases, How Well Do They Do in College? - Top Hat: (2019). https://tophat.com/blog/65-of-students-skip-required-textbook/. Accessed: 2021-11-04.

[2] Anderson, L.W. and Krathwohl, D.R. (2001). A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom's Taxonomy. Addison Wesley Longman.

[3] Atkinson, R.K. et al. (2000). Learning from examples: Instructional principles from the worked examples research. Review of Educational Research. 70, 2, 181–214.

[4] Biggs, J.B. and Collis, K.F. (1982). Evaluating the Quality of Learning : The SOLO Taxonomy (structure of the observed learning outcome). Academic Press.

[5] Cansu, F.K. and Cansu, S.K. (2019). An Overview of Computational Thinking. International Journal of Computer Science Education in Schools. 3, 1, 17–30.

[6] Chandler, P. et al. (1991). Cognitive Load Theory and the Format of Instruction Cognitive Load Theory and the Format of Instruction. Cognition and Instruction. 8, 4, 293–332.

[7] Csizmadia, A. et al. (2015). Computational thinking A guide for teachers. Computing At School. October 2018, 18.

[8] Decker, A. et al. (2020). Using Subgoal Labeling in Teaching Introductory Programming. J. Comput. Sci. Coll. 35, 8, 249–251.

[9] Eiriksdottir, E. and Catrambone, R. (2011). Procedural instructions, principles, and examples: How to structure instructions for procedural tasks to enhance performance, learning, and transfer. Human Factors. 53, 6, 749–770.

[10] Ericson, B. et al. (2015). Usability and usage of interactive features in an online ebook for cs teachers. ACM International Conference Proceeding Series. 09-11-Nove, 111–120.

[11] van Gog, T. et al. (2008). Effects of studying sequences of process-oriented and product-oriented worked examples on troubleshooting transfer efficiency. Learning and Instruction. 18, 3, 211–222.

[12] Granitz, N. et al. (2021). Textbooks for the YouTube generation? A case study on the shift from text to video. Journal of Education for Business. 96, 5, 299–307.

[13] Hansen, S.M. (2017). Deconstruction/Reconstruction: A Pedagogic Method for Teaching Programming to Graphic Designers. Proceedings of the 20h Generative Art

Conference GA2017 (Ravenna, Italy, 2017).

[14] Hansen, S.M. (2018). Mapping Creative Coding Courses: Towards Bespoke Programming Curricula in Graphic Design Education.

[15] Hansen, S.M. (2019). public class Graphic_Design implements Code { // yes, but how? }: an investigation towards bespoke Creative Coding programming courses in graphic design education. Aarhus University, Faculty of Arts, School of Culture an.

[16] Kay, R.H. and Knaack, L. (2009). Analysing the Effectiveness of Learning Objects for Secondary School Science Classrooms. Journal of Educational Multimedia and Hypermedia. 18, 1, 113–135.

[17] Koretsky, M.D. (2020). Re-flipping in the remote classroom: The surprising uptake of video-recorded worked examples. Journal of Chemical Education. 97, 9, 2754–2759.

[18] Lowe, R.K. and Schnotz, W. (2014). Animation principles in multimedia learning. The Cambridge Handbook of Multimedia Learning, Second Edition. 513–546.

[19] Margulieux, L.E. and Catrambone, R. (2016). Improving problem solving with subgoal labels in expository text and worked examples. Learning and Instruction. 42, 58–71.

[20] Mayer, R.E. and Moreno, R. (2003). Nine Ways to Reduce Cognitive Load in Multimedia Learning. Educational Psychologist. 38, 1, 43–52.

[21] van der Meij, H. and van der Meij, J. (2013). Eight guidelines for the design of instructional videos for software training. Technical Communication. 60, 3, 205–228.

[22] Morrison, B.B. et al. (2015). Subgoals, context, and worked examples in learning computing problem solving. ICER 2015 - Proceedings of the 2015 ACM Conference on International Computing Education Research. April 2016, 21–30.

[23] Papert, S. (1980). Mindstorms. Basic Books Inc.

[24] Pearson (2018). Beyond millennials: The next generation of learners. Global Research & Insights. August, 1–21.

[25] Petersen, K.H. (2015). Mål og taksonomier – i praksis. Fleksibel læring. 3, 3–5.

[26] Resnick, M. et al. (2005). Design principles for tools to support creative thinking. Proceedings of the Workshop on Creativity Support Tools.

[27] Seemiller, C. and Grace, M. (2017). Generation Z: Educating and Engaging the Next Generation of Students. About Campus: Enriching the Student Learning Experience. 22, 3, 21–26.

[28] Singer, J. (2020). Notes on notebooks: Is Jupyter the bringer of jollity? Onward! 2020 - Proceedings of the 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Co-located with SPLASH 2020. April, 180–186.

[29] Sweller, J. et al. (2019). Cognitive Architecture and Instructional Design: 20 Years Later. Educational Psychology Review. 31, 2, 261–292.

[30] Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. Cognitive Science. 12, 2, 257–285.

[31] Tarquini, G. and McDorman, R.E. (2019). Video tutorials: An expanding audiovisual genre. Journal of Specialised Translation. 32, 146–170.

[32] Vafeas, M. (2013). Attitudes Toward, and Use of, Textbooks Among Marketing Undergraduates: An Exploratory Study. Journal of Marketing Education. 35, 3, 245–258.

[33] Wing, J.M. (2006). Computational thinking. Communications of the ACM. 49, 3, 33–35.

[34] Zhang, D. et al. (2006). Instructional video in e-learning: Assessing the impact of interactive video on learning effectiveness. Information and Management. 43, 1, 15–27.