Why Teach Coding to Art & Design Students?

A survey and literature review exploring the topic / potential of teaching programming to art & design students

study initially conducted 2017-18; re-edited 2020-21

Table of Contents

1 Abstract / Summary

1.1 Introduction

The skill of programming is increasingly used in creative practices. Where this is the case, either the final work itself derived from programming or self-created digital tools were involved in the process of creating the work. Acknowledging this persistent trend, it seems timely to think about if there is a place for teaching programming in curriculums of creative study subjects that are not inherently involved in digital production (e.g. Fine Arts, Graphic- and Product-Design vs. Interactive- or Game-Design).

1.2 Aim

The study aims to gather arguments* in relation to teaching programming to students in the creative fields, to present these arguments in a structured way and to make them accessible to relevant stakeholders.

Foremost this addresses the questions of in how far teaching how to code is relevant for art and design disciplines in higher education and which implications can be expected if doing so.

* with *arguments* are meant reasoned views, positions and opinions on the subject

1.3 Methods

A literature review and an email survey were conducted. A method was tested to evaluate literature sources formally for assumed relevance by visually analysing word-cloud representations of the most frequent words. All literature sources and survey responses were contextually analysed whereby interpretative, summarising comments on relevant text passages were formed and categorised. The study uses a qualitative research method with open coding scheme, i.e. a strategy where categories are not pre-defined, but derive from the process of analysing the sources.

1.4 Results

A considerable amount of arguments was extracted and classified. More than 130 arguments are grouped in 10 main categories – some of those further divided into several sub-categories. In addition to listing the arguments in enumerated, tabular form, a visual representation of the concluding categories was constructed as a mind-map inspired diagram, which serves as a comprehensive overview of the issues addressed.

1.5 Conclusion

The majority of arguments are in favour of teaching programming – this for various reasons, though with a focus on calling for empowerment by widening the spectrum of possibilities in creative expression and for keeping up with developments in an increasingly technology-influenced society, both practically and conceptually.

But also critical voices came to attention – mostly concerned about an implicit thrive in commodification of creative endeavours (i.e. enforcing negative socio-economical trends), the danger of de-personification, a possible constringence of tools / techniques (if leading to a narrowly specialised focus), and last but not least, questioning the level of relevance in aesthetic expression.

The study identified challenges that would need to be addressed on the way to higher integration

into the curriculum, especially if it were considered to be made compulsory. Nevertheless, no objections were raised to teaching programming as an optional part or specialisation area of studies in the creative fields.

2 Personal Motivation to Conduct the Study

For more than decade, I worked as an interactive designer and creative programmer, predominantly developing commercial interactive installations and websites. Increasingly drawn to the visual arts, I decided to study again. After finishing a Master in Fine Art and whilst establishing my artistic practice, I worked at the University of the Arts London (UAL), where I was helping students developing their skills with creative computer applications (mostly Adobe Suite's Photoshop, Illustrator, Indesign & Premiere).

Although not working within the realm of digital art in my artistic practice (that means my artwork is usually not in digital form), I used these applications, as well: for retouching photos and scans of artworks, creating portfolios, sketching out concepts, ...

Learning that it is possible to *script* these applications*, and then how to do it, opened up new perspectives to me. (*this means that almost everything that a user can do manually within these applications, can also be executed within the applications by running a computer program – here called 'script'). Creating imagery / (interactive) art by the use of programming was not new to me, though it first dawned on me that I could build my own digital tools that would help me

- to automate some repetitive tasks in those environments that I find tedious to execute manually
- with expressing myself artistically through generative visualisations of pre-defined concepts / systems – including the kind, that make use of by-chance decisions.

For me it was relatively easy to access the offered technology as I could already write code in several different programming languages and am familiar with the involved strategies, including how to find resources and help on the internet – which is important especially when starting out within a new programming environment.

Obviously, not every artist works in the same way – and for sure not like I do. Nevertheless, I thought about that if I benefit from the skills described above, what are the chances that others acquiring these skills would benefit, as well? ('What is potentially a useful skill' is a general question in contemporary art; considered contestants include, but are by far not limited to ... drawing, painting, photographing, filming, welding, pottery, media-analysis, [il-]logical thinking ... computation is just one of many themes with potential in a long list, though, indeed quite a universally helpful one.)

I came to the conclusion that it would be irresponsible to withhold the opportunities from other people and developed an ambition to offer courses and workshops to students in creative fields who are interested in the topic of creating their own digital tools.

But, since the world is not just me, I wondered how different stakeholders think about if art & design students should (get offered to) learn how to code, and what their reasoning would be. Am I overly biased in only seeing advantages? What are possible negative implications and what the challenges to introducing programming into the curriculums of creative study programmes?

In the course of a self-initiated project as part of my PGCert Academic Practice studies, I took the opportunity to trying to find answers by looking for argumentation from people involved in teaching, programming, working artistically / creatively or, more often than not, in several of the mentioned.

3 Methodology

3.1 Overview of involved research processes

The task of compiling a list of a-priori unknown, descriptive data (and corresponding grouping through classifiers) asks for a qualitative research method. In order to gather the searched for informative data, suitable sources have to be identified and dissected by extracting the desired information, either in form of quoted text passages or in form of summerised interpretations of relevant text passages.

Structuring the found information through categorisation, i.e. further abstraction, helps to establish an overview of the researched field and eases navigating through the space of found data in a top-down approach (from the general to the specific).

3.2 Data Collection Methods

3.2.1 Literature Review

A study should start with researching existing literature, that is, to find sources that are able to establish the context, to see what other researchers have found out and if there are any results which the own research effort can build upon. To my surprise, nothing of what I term 'classical academic literature' (books and research papers in english language) that relates education in the creative fields with digital tools or programming could be found by me in the academic libraries.

A conducted online research for suitable sources to consider for a literature review resulted in putting the focus on examining a selection of texts that relate to the topic of 'programming digital creative tools' in a wider sense. Attempts were made to find a statistical selection process for the suitability of sources, based on keyword extraction / periodicity of words (see separate outline in Appendix: 6.3 Relevance Testing of Literature Sources by Word-Cloud Analysis). Identified as suitable sources where:

- articles, interviews (online magazines, blogs)
- descriptions of university courses / MOOC programmes
- educational content guidelines (e.g. issued by UK government)
- documentaries (case studies, including videos)

In this study, the literature review was not used as secondary research, but directly as primary source for findings that add to the collection of arguments. In this respect one could partially ascribe this research project a meta-study character.

3.2.2 Email Survey

A well established method for qualitative research is that of a survey, meaning the process of personally gathering information from other people. Due to a limited timeframe and focus on collecting arguments from experts as diverse as possible, the preferable method for survey data collection in form of interviews was dismissed in favor of a survey in form of an open-ended questionnaire via email. A broad spectrum of stakeholders was invited to reply to the single question “Why teach coding to art & design students?”, either in writing or by selecting or fabricating image material. The question was intended to serve as a 'stimulant', a starting point to elaborate on the topic and to obtain a wide spectrum of personal feedback. (see Appendix: 6.4 Email Survey Invitation Text)

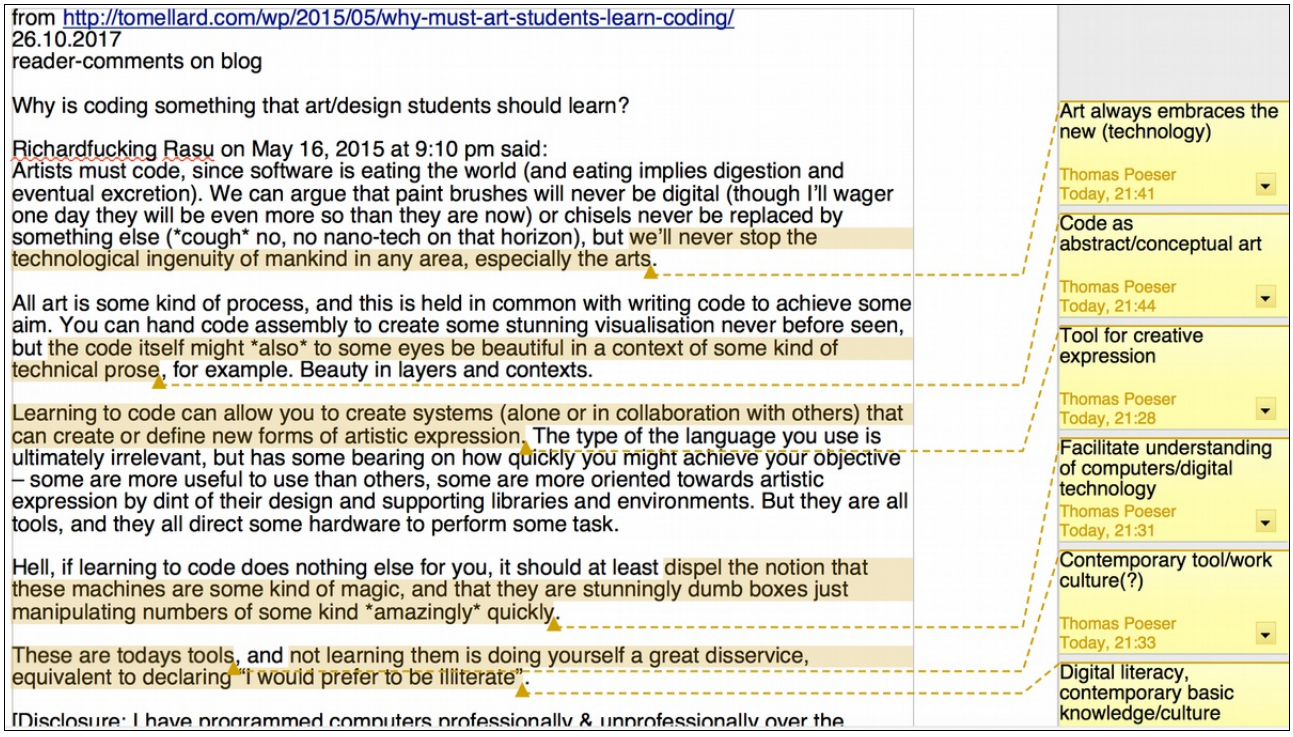
Sampling

The selection of potential participants derived from examining topic-related backgrounds (educators / course planners and / or practitioners with or without programming experience) and were then chosen by level of accessibility, that is, predominantly considering the ease of establishing contact and the estimated response probability of people invited (a.k.a. 'convenience sample' [1]). Specifically of interest in terms of contributions that were expected to contain the most valuable information and meaningful data (and hence serving the purpose) were

- heads of art- and design-related study programs
- course leaders, educators, lecturers
- creative practitioners (professional artists, designers, architects)

3.3 Contextual Content Analysis of Sources

The actual extraction of meaningful information from the collection of source texts started by marking relevant text passages, summarising longer passages and commenting those in an interpretative manner as remarks within the text file.



Example of commenting text passages

A script was used to collect the remarks and combine them in a tabulated data set (.csv format file).

For the evaluation of image material (e.g. the alternative email-response in form of an image), a visual research method was used. Similar to the approach in the 'Photovoice' method that is used mainly in context of social community voicing [2], images were interpreted in the context of the source selection topic of 'programming digital creative tools' or the survey question 'Why teach coding to art & design students?', respectively.

With help of another script all gathered remarks were imported into Adobe Illustrator as individually movable text-blocks to facilitate structuring / grouping in an intuitive way, similar to creating a collage by arranging a variety of clippings taken from preexisting materials.

3.4 Structuring by Applying an Open Classification Coding Scheme

Searching for possible arguments of a qualitative character means that they are to be considered unknown to the researcher beforehand. The study attempts to structure these findings by categorisation. Since a pre-evaluation definition of possible categories beyond general terms like 'pro / contra' arguments would not support the intention of trying to keep the research methodology free from bias, the structuring categories are to be considered unknown to the researcher beforehand, as well. This implies an open coding scheme where themes develop from the growing collection of arguments, that is, the categories are emerging whilst in the process of analysing the sources.

This methodology involves the foremost subjective process of finding and naming connections between descriptive summaries of statements found in the sources (a.k.a. 'coding' the data – i.e. to assign labels to the marked data).

Every newly established category asks for a review of previously worked through text in a repetitive manner. Only after completing the compilation of a list of derived categories (and sub-categories), a final classification of each argument is possible.

Summarised step-by-step process:

1) find arguments (views, positions, opinions) throughout all sources
—> description in short phrases, keywords

2) find structure / pattern in descriptions
—> conceptual categories (and maybe sub-categories)

3) categorise occurrences of arguments (classification)
    3.1) possible surfacing of new (sub-)categories
    3.2) possible re-structuring of system of categories
    —> re-categorisation process

3.5 Ethical Considerations

In order to encourage an honest response by survey participants (i.e. them not having to fear negative consequences / repercussions / retributions for their stated opinions) strict measures were taken and communicated to be able to guarantee data protection to survey participants:

- Names of contributors are not publicised.
- Survey replies are securely stored, all data from survey contributions are anonymised.
- Participants have the possibility to withdraw their consent to using the information provided with their survey contribution.

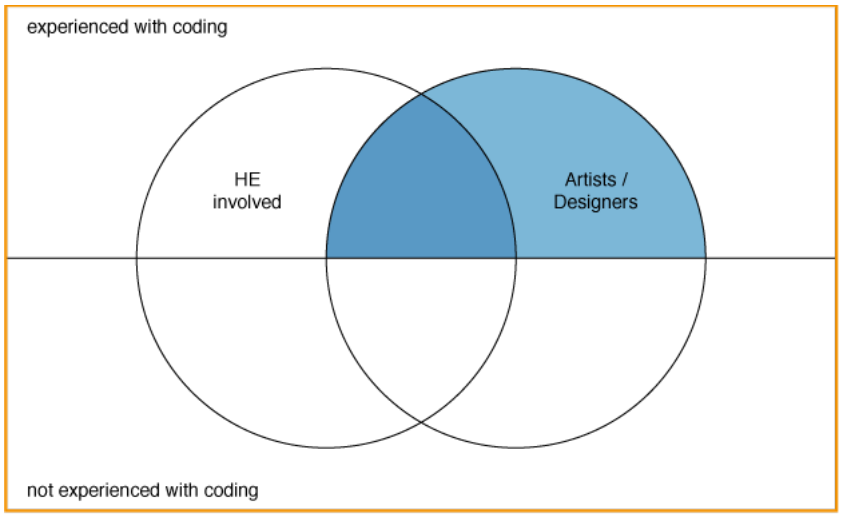
In support of democratising research results, all findings are accessible to the public – temporarily on a blog platform provided by the University of the Arts London, and after the re-edit on one of my personal, publicly accessible websites. The outcomes of the study will be used to support my efforts to offer courses and workshops.

4 Results

4.1 Context of Sources

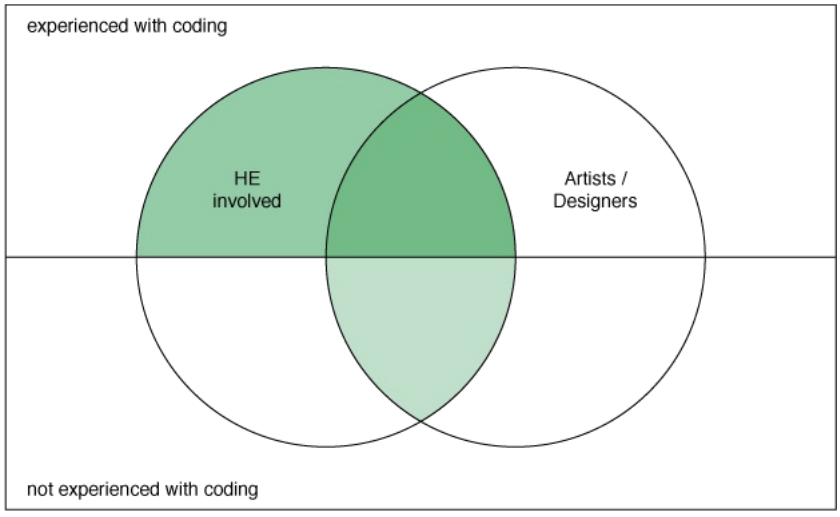
In order to better judge the findings of this study, an overview of the origins of the sources in terms of the relevant backgrounds of authors, interviewees and survey participants might be of interest.

Whilst the interviews published online are with artists and designers that are experienced with programming (with some of those also teaching at colleges / universities), the online articles cannot be clearly attributed to being written by authors with specific backgrounds related to coding or higher education (HE).



Categorisation of **Interviewees** and **Authors**

All survey participants are involved in higher education (from five different universities / colleges in the UK). Most of them are experienced with coding, the ones that are not are artists or designers. The majority of participants are educators who are artists or designers themselves – or at least teach an artistic or design related subject.



Categorisation of **Survey Participants**

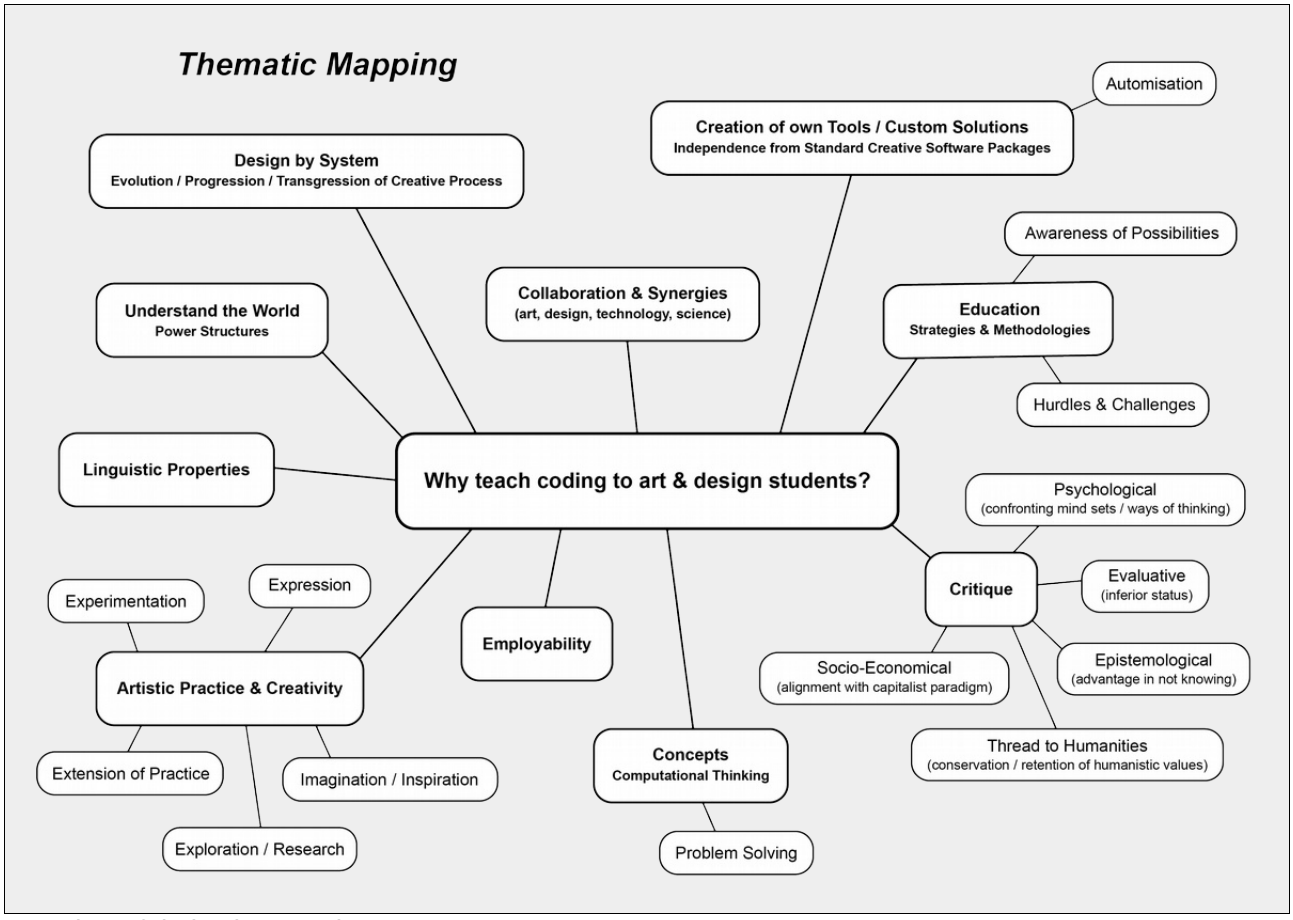
4.2 Primary Data Collected

For the literature review ten online articles and interviews, as well as six course descriptions from the respective educational institutions' websites were sourced (see Appendix: 6.2 Literature Review Sources).

The invitation to participate in the study received ten replies with extensive contributions in content (see Appendix: 6.5 Anonymised Email Survey Responses).

4.3 Derived Results

Through the examination of all primary data the following themes / categories evolved:



Overview of derived categories

The extracted arguments were, in concise note-style form, listed in respective categories (see Appendix: 6.6 Categorised Arguments).

4.4 Conclusions / Discussions

Sources from the public sector in the UK offer solely a statutory guidance for the national curriculum in England by the UK Department for Education, outlining what needs to be taught to pupils during their GCSE (General Certificate of Secondary Education) key stages in any computing courses they might choose [3]. No intersection of contents with the respective guidelines for art and design programmes of study can be identified [4].

At least those guidelines tells us, in how far students entering higher education might already have acquired computational skills, that they might want (if not expect) to see integrated into their further studies – even though, it can be safely assumed that only few pupils with programming skills will enter an art or design course at university (well, some, like me, do – and maybe more in the future; although the current focus on supporting MINT subjects would suggest otherwise). Within architecture courses (and, to a slightly lesser extend, other design-related courses), we might find a higher rate of students that are programming affine, as the already present high degree and increasing use of computational techniques in their subject suggests.

In **publications relating to higher education in creative fields the UK** programming is only mentioned in specialised study programmes like 'Interactive Design' or 'Game Design' (see Appendix: 6.2.2 Course Descriptions). Only very limited interdisciplinary approaches that include programming are observable, with the notable exception of the 'Institute of Coding' with involvement of several HE institutions, though it is not specifically focusing on the creative sector [5]. So far, teaching how to program seems to have not been included in regular curriculums of art and design studies - only with higher specialisation in subjects that point towards engineering disciplines like Industrial Design and Architecture, e.g. in support of contemporary design and manufacturing techniques (CAD/CAM related programming, e.g. 3-dimensional weaving) and business strategies. Key-lines here are: parameterised design, product customisation and manufacturing on demand. (see Appendix: 6.1 Exemplary References of Works, Case Studies, Practitioners)

The information extracted from **articles, interviews and survey responses** provided the majority of collected arguments. As to a high degree expected, from the perspective of practitioners that are already using programming in their work, the suggestion of teaching students how to code is mostly seen positively. These views might be partially influenced by a motivational bias (e.g. a common inclination of not wanting to question one's own decisions and doings), though, consistently spanning all sources, the pointing out of benefits is supported by reasonable argumentation. But also, since being close to the subject, some of those involved developed a critical eye for aspects that can only surface after being exposed to a field for a sufficient time and that are hidden to an outsider view.

Noticeable is a justified critical stance in accordance to non-approving views of recently accelerating developments in our socio-economic system. We currently see governments that, nudged by influential think-tanks, are proclaiming *The 4th Industrial Revolution* (4IR) [6] – what is presented as the inevitable consequence of progression (i.e. advancement for the greater good and / or forced by global competition) is in short a big-economy friendly, technology driven initiative (including themes like *Automisation*, *Internet of Things*, *Artificial Intelligence*, .. all the way toward *Transhumanism*) with wide-spanning consequences for societies worldwide.

Fitting this narrative, programming is by many already seen as the new 'blue-collar job' (an expression that designate the mass worker) [7]. Computer programming has become a commoditised skill. Outsourcing of programming tasks to lower wage countries is common practice, whilst there are numerous programs in place (some of them either fully or partially state-sponsored) to re-train people with the aim to supply lateral entry employees with sufficient programming skills to larger companies in the information-technological sector.

All too often, attempts by HE institutions to integrate new technology in their creative courses appear to foremost fulfill a policy of making art & design more 'valuable' in terms of economical paradigms (e.g. optimising the exploitability of human resources -to-be). So students are being taught skills that make them fit (for) the modern job, were the skillset is predefined by the industry according to their current needs.

Whether you welcome or reject those developments – all the above imply that programming enters more and more fields where it will be applied. The academics behind *Design & Computation* – a *"new inter-university, interdisciplinary, research-based Master's program between Berlin University of the Arts and Technical University Berlin"* – acknowledge *"an ever-increasing penetration of far-reaching areas of public and private life by new technological applications"* and that *"the classical design disciplines are faced with the challenge of incorporating new technologies and methods into existing practices at an increasing pace while subjecting them to critical review"* [8].

Programming skills differ in purpose depending on the learner's context in the creative fields. While students in the applied arts might want to acquire programming skills for creating or extending digital tools to their professional advantage, artists might want to do the same for use in their artistic expression, but also might want to critically examine the cultural change and implications stemming from the applications of computer programming through their art practice. For the purpose of developing a substantiated position, the possibility to gain insight and first hand experience in the field is beneficial – if only with another meaningful outcome of realising that digital technology is not some kind of magic, something that is too difficult to understand for 'normal' people, but indeed, can be handled / used proactively. Acquiring knowledge has to do with self-empowerment and in our case is able to grant a certain amount of freedom from technological authority (e.g. being constrained by the digital tools available vs. creating your own).

So-called 'disruptive technologies' are implemented to replace existing process structures in the analog world. Whilst currently the phrase "Software is eating the world" [9] relates foremost to businesses and industries, will soon the same apply to tools for creative production? Will we then see a renaissance of (traditional) craftsmanship as counter programme?

Many creatives (especially those in the fine arts) are only slowly warming up with the idea of digital technology entering their fields – with the exceptions of already established forms of digital sound and imaging (video art also needed decades to reach today's status). Artistic traditions involve and are defined by manual skills. This might explain why many artists are self-distancing from technical approaches and solutions, which are seen as of minor value and of inferior status within the art world. Furthermore, a culture of pigeonholing creatives associated with technological skillsets (as was mentioned in *web_source_1*; see Appendix: 6.2.1 Articles and Interviews) is not exactly an incentive for artistically motivated people to get involved with computer programming.

The current level of programming's relevance in artistic expression remains ambiguous: aligning with post-modernistic paradigms, tools and means of expression become more and more irrelevant; almost any form of expression is (or at least seems to be) equally justifiable, every technique is acceptable – computing as much as drawing, painting, printing, photography, moving image, sculpting, weaving, sewing, ...

Whilst some of them are taught, others should be made available to be studied by offering 'first contact points' and further support in obtaining knowledge if desired by the learner. It is clear that digital tools are not necessary for meaningful expression. But here it is also important to point out the universal character of programming skills that are laid-out in this study's results. Although 'creative programming' is rarely taught in courses, there is a noticeable trend in artists picking up the available digital techniques to use in their practice (see Appendix: 6.1 Exemplary References of Works, Case Studies, Practitioners). So far, most of these are either autodidacts or work with the help of people that are proficient in programming.

4.5 Achievement

The study's aim was not to jump to conclusions but to enrich the spectrum of debatable positions and possibilities. Its outcome provides a good overview for this purpose by presenting a collection of arguments from diverse sources in one place, structured in a way that highlights the different areas of aspects to be considered. The findings will hopefully work in support of those that wish to engage in helping students in creative fields to find access to programming skills as an extension to their practice.

In terms of shaping the future of creative course curricula, some of the points might help to overcome already recognized hurdles and tackle foreseeable challenges, that involved educators might not have been aware of.

These findings can be used to convince stakeholders of benefits (if not urgency) and also warn of making counterproductive decisions, taking wrong directions in developing study programmes and avoid mistakes in goal-setting and implementation of course material.

4.6 Limitations and Scope

Apart from the fact that the above discussion can only touch some of the aspects involved, there are several additional points that would benefit the research purpose:

- The available note-style points can be more precisely elaborated upon to deliver a more precise description of opportunities and issues involved.
- Conducting interviews will deliver more detailed aspects, e.g. in regards to identifying problems in teaching programming, highlighting prejudices, ...
- During the time between acquiring the study data and this re-edit, many more general sources (at least in form of articles) about programming entering previously underrepresented fields became available. There are discussions on philosophical aspects concerning AI, e.g. in crediting 'AI generated' art, following the first of such an artwork sold at the Christie's Auction House in October 2018 [10]. There are more (still niche) design agencies specialising on parametric design, and the data science driven economy is booming. Artistic responses (as critique of the current culture) as well as own use of digital tools in the artistic practice became more prominent.

  In consequence, there is an increased amount of case studies available that could be analysed and probably new lines of argumentation derived.

5 References

[1] Dawson, C. (2009) Introduction to research methods: a practical guide for anyone undertaking a research project. Oxford : How To Books [UAL library electronic resource from Dawsonera], p.51

[2] see: https://en.wikipedia.org/wiki/Photovoice [Accessed 8 January 2021]

[3] UK Department for Education. 2013. Statutory guidance — National curriculum in England: computing programmes of study. [Online]. [Accessed 26 November 2017]. Available from: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

[4] UK Department for Education. 2013. Statutory guidance — National curriculum in England: art and design programmes of study. [Online]. [Accessed 26 November 2017]. Available from: https://www.gov.uk/government/publications/national-curriculum-in-england-art-and-design-programmes-of-study/national-curriculum-in-england-art-and-design-programmes-of-study

[5] see: https://instituteofcoding.org/ [Accessed 6 January 2021]

[6] Hancock, M. (UK Minister for Digital) 2017. Transcript of Speech: The Fourth Industrial Revolution. [Online]. [Accessed 6 January 2021]. Available from: https://www.gov.uk/government/speeches/the-4th-industrial-revolution

[7] Thompson, C. (2017) The Next Big Blue-Collar Job Is Coding. Wire magazine. [Online]. [Accessed 6 January 2021] Available from: https://www.wired.com/2017/02/programming-is-the-new-blue-collar-job

[8] see: https://www.design-computation.berlin/#program > background [Accessed 8 January 2021]

[9] Andreessen M. (2011) Why Software Is Eating The World. The Wall Street Journal. [Online]. [Accessed 8 January 2021]. Available from: https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

[10] Max Planck Institute for Human Development, Berlin. 2020. When artificial intelligence creates art. [Online]. [Accessed 8 January 2021]. Available from: https://www.mpg.de/15455301/0929-bild-134137-artificial-intelligence-in-art-a-simple-tool-or-creative-genius1

6 Appendix

6.1 Exemplary References of Works, Case Studies, Practitioners

[All accessed 8 January 2021]

Casa Music Identity
https://sagmeister.com/work/casa-da-musica/

Nutella Unique Design Series
https://www.dezeen.com/2017/06/01/algorithm-seven-million-different-jars-nutella-packaging-design

Eye Unique Cover Series
https://muirmcneil.com/project/eye-magazine/
https://www.dezeen.com/2017/09/11/muirmcneil-8000-unique-covers-eye-magazine-design-graphics

Custom Branding Visuals Tool
https://onformative.com/work/actelion-imagery-wizard/

Generative Design Agency
https://process.studio

Graphic Designer, Animator (uses self-created digital tools)
https://ksawerykomputery.pl

Organisation / Gallery for Code Generated Art
https://www.artxcode.io

Compart Archive for Early Digital Art 1950-79
http://dada.compart-bremen.de

Visual Artists (using custom digital tools; artworks not necessarily in digital form)

- Manfred Mohr http://www.emohr.com
- Vera Molnar http://www.veramolnar.com
- Ryoji Ikeda https://www.ryojiikeda.com
- Carsten Nicolai http://www.carstennicolai.de
- Tauba Auerbach https://taubaauerbach.com
- Avery Singer https://www.hauserwirth.com/artists/26658-avery-singer
- Cheyney Thompson https://www.galeriebuchholz.de/artists/cheyney-thompson
- Thomas Ruff (artificial darkroom photograms) https://gagosian.com/exhibitions/2014/thomas-ruff-photograms-and-negatives
- Sun Yuan and Peng Yu https://www.youtube.com/watch?v=_oZc6LTMva8
- The Author of this Study https://thomaspoeser.com

6.2 Literature Review Sources

6.2.1 Articles and Interviews

[All accessed 26 October 2017]

Compiled sources (relevant passages) can be accessed in text (.rtf format) as .zip archive from https://www.tp23.co.uk/why_teach_coding_to_art_and_design_students/wtctaads-web_source-articles.zip

web_source_1
http://tomellard.com/wp/2015/05/why-must-art-students-learn-coding

web_source_2
http://www.eyemagazine.com/feature/article/teaching-in-the-spaces-between-code-and-design

web_source_3
https://medium.com/progetto-grafico/coding-as-a-way-of-thinking-interview-with-casey-reas-cbb9ecdbb980

web_source_4
https://www.digitalartsonline.co.uk/features/creative-business/why-british-art-design-education-at-university-is-broken-how-we-fix-it

web_source_5
http://www.eyemagazine.com/feature/article/reputations-karsten-schmidt

web_source_6
https://www.edutopia.org/blog/qa-art-coding-suzie-boss

web_source_7
https://www.edsurge.com/news/2014-12-26-five-ways-to-teach-creativity-through-coding

web_source_8
https://artech.cc/teaching/

web_source_9
https://runemadsen.com/blog/on-meta-design-and-algorithmic-design-systems/

web_source_10
http://www.eyemagazine.com/feature/article/tools-to-make-or-break

6.2.2 Course Descriptions

[All accessed 30 November 2017]

Compiled sources (relevant passages) can be accessed in text (.rtf format) from
https://www.tp23.co.uk/why_teach_coding_to_art_and_design_students/wtctaads-web_source-course_discriptions.rtf

https://www.arts.ac.uk/csm/courses/short-courses/ fine-art/creative-coding-an-introduction-to-programming-with-processing-online/

https://www.futurelearn.com/courses/creative-coding

https://www.kadenze.com/courses/computing-form-and-shape-python-programming-with-the-rhinoscript-library/info

https://www.kadenze.com/courses/creative-programming-for-audiovisual-art/info

https://www.coursera.org/learn/digitalmedia

https://thephotographersgallery.org.uk/whats-on/workshop/coding-for-beginners

6.3 Relevance Testing of Literature Sources by Word-Cloud Analysis

A short presentation of methods and findings can be accessed as PDF from
https://www.tp23.co.uk/why_teach_coding_to_art_and_design_students/wtctaads-frequency_methods.pdf

6.4 Email Survey Invitation Text

The content can be accessed in text (.rtf format) from
https://www.tp23.co.uk/why_teach_coding_to_art_and_design_students/wtctaads-survey-invitation.rtf

6.5 Anonymised Email Survey Responses

The content can be accessed in text (.rtf format) from
https://www.tp23.co.uk/why_teach_coding_to_art_and_design_students/wtctaads-survey-responses.rtf

6.6 Categorised Arguments

A high resolution diagram with all categories (*Thematic Mapping*) can be accessed as PDF from https://www.tp23.co.uk/why_teach_coding_to_art_and_design_students/wtctaads-categories_map.pdf

Content / Categories (each category is linked to jump directly to the respective list of arguments)

... the listing starts on the following page ...

Understand the World - Power Structures

- Program or be Programmed – equalisation of power in power structure?
- Code is the base-level operating semantics for the modern world.
  (digital economy rules by use of algorithms; i.e. understanding of power structure)
- To learn about power and control structures and engage with the material that is used against oneself.
- Enables participation in a discourse around a life organised, understood and controlled by computers.
- Contemporary culture: increasingly digital, mediated by computers. Role of the arts to reflect on current setting.
- Empowering students means helping them to learn programming, as “all [of them] know they live in a world run on screens”.
- Beneficial to understand working of software as space and processes in the world are increasingly modeled after software.
- Design as an unlimited practice dealing with everything that can be interacted with.
  (need to understand the semantic structure of the world)
- Broadens the mind, facilitates understanding of the modern world which art often requires
  (needs wisdom as much as creativity)

Concepts – Computational Thinking

- Learn 'algorithmic thinking'
  (ways to conceptualise and model the world to express creative ideas using programming)
- Understand that programming is a unique interaction with a common system
- Learn to figure out how to make things happen (also teaches perseverance).
- Mental model mapping (of intention) becoming a conscious act
- Learn to work under constraints, how to abstract bigger problems into smaller tasks, how to be self-reliant, experiment and learn new subjects without supervision or direction, and how to collaborate with others
- Facilitate insight in concepts / processes of how a computer works / can be used / made to do something (facilitate understanding of computers / digital technology)
- Beneficial to encounter computational way of working (strategic methodology)
- A way of thinking (humanist activity, not technical skill). A notation system of instructions.
- Helps to understand how complexity is built from simple parts.
  (reduction of complexity by partialisation)
- Explore the fundamental principles common to all programming languages
- Understanding the underlying principles which are transferable to any other programming language

Problem Solving

- To become self-driving, capable learners who know how to identify and frame problems they want to solve (where it makes sense to solve them with a computer)
- Experience with coding / computational thinking leads to new approaches to solving problems, use of spatial reasoning, learn to define problems.

Linguistic Properties

- Similarity to (set of) spoken language
- Limited vocabulary opens up large spectrum of possibilities
- Functionality based on linguistic architecture
  (linguistic concepts learned intrinsically when learning how to program)
- Coding similar to (spoken, foreign) language that opens new perspectives
- Coding is a language, and allows for articulation of data in varied ways.

Collaboration & Synergies (art, design, technology, science)

- Facilitates bridging art and technological domains (generate synergies)
- Facilitates collaboration (to work, communicate effectively) with technical oriented people
- Coding as part of a more interdisciplinary approach in design
- Facilitates cross discipline projects with code as common language.
- Benefits already found running creative courses alongside tech & science subjects; reversal should be as beneficial. (cross influence of art / design and tech / science)

Employability

- If we didn't we would be neglecting to equip students with skills to make the most of any opportunity they may be offered
- To become future-ready cultural contributors.
- Increases employability
- Digital literacy, contemporary basic knowledge / culture
- Promote software literacy within the visual arts & make accessible to diverse communities
- Contemporary tool / work culture

Artistic Practice & Creativity

- Programming is a form of creation
- Provision of creative license, a permission (i.e. the ability & precursor) to change the world.
- Facilitates freedom of thought (widens / diversifies spectrum of thinking)
- Artists (including Photographers) increasingly look into programming to help them produce artworks.
- Finding an elegant solution for a problem equals an artistic endeavour
  (training the artistic sense ?)

Imaginatination / Inspiration

- Become less intimidated by technology (more encouraged to use technology), opens up imagination
- Creates conditions for new ideas and forms in the visual arts (inspirational)
- Increase possibility of imagining and inventing something new

Exploration / Research

- Computer code allows human desire to be explored at the level of unconscious and implicit human experience, at a scale and with an impact that no other medium has ever achieved.
- Integration of coding as form of artistic discourse and speculative research.
- Facilitates conceptional understanding of art / design work
  (programming as choreography of actions and decisions)
- Description of geometric processes
  (learn how to describe – and thereby better understand – creative process)
- Explore new aesthetic representations

Experimentation

- Experimentation through (programmatical) conceptualisation
- Fosters a culture of experimentation
- Code allows for new approaches to existing forms of expression.
- Lead to novel kinds, relationships, and orders of shape and form

Expression

- Programming as a mode of expression, to empower students to exercise their creativity
- A means to enable articulation of ideas through computing and a digital interface
- Tool for creative expression Code as abstract / conceptual art

Extention of Practice

- Opportunity of feeding back into traditional processes
- Extend conventional workflow with generative methods / self-created tools

Design by System - Evolution / Progression / Transgression in Creative Process

- Continuing trend in the design world: the rise of the meta-designer and algorithmic design systems (i.e. instructional, rule-based design)
- Important evolution of the design profession
- New generation of (influential) designers adopt systematic approach and technical process as accelerator for creativity and liberation from manual techniques and traditional division of labor.
- Expression though systems already present in arts / design history, e.g. theoretical conceptual toward an instructional manual workflow.
- Transition from traditional design practice (direct production of final artefact) to creation of a design system (conceptual / instructional format that, through programming, generates context specific output)
- Example: logo system dynamically used for diverse media output (though programmatic generation of variation)
- Dynamic (with temporal transitional stages) and data-based design products require programming within design process
- Introducing random properties into artwork
- Create parameterised and algorithmically conceptualised artwork
- “The territory has largely shifted from paint to code: so that is where we must go.” – Golan Levin Art always embraces the new (technological ingenuity)
- Code and algorithms performing traditional art-making in the digital domain
- Increasingly easier access to / lower level of entry for algorithmic design tools (programming environments)
- A new medium might be discovered only after a substantial delay (example: typewriter art several decades after invention of typewriter)
- Combining the development of the concept and its implementation breaks up established structures (e.g. hierarchical work classification and separation in agencies)

Creation of own Tools / Custom Solutions - Independence from Standard Creative Software Packages

- Enables to achieve things outside the box
- Allows to create own tools (go beyond using off-the-shelf tools).
- Make custom (software / hardware) interfaces
- Enables to develop custom solutions
- Expand the potential of programmable devices (static vs. self-defined functionality)
- Existing software solutions (for creative work) have a strong pre-defined influence on aesthetics of creative products
- Transition from user to creator & to overcome constraints (of software).
- Going beyond digital craft skills (i.e. beyond working with standard creative applications) Dependence on a (digital) tool shapes (or disfigures) ideas through the tool's metaphors
- “We make our own tools, and then they shape us”
- History of artists creating their own tools (thereby strengthening relationship with materials and process) recently weakened by software revolution (i.e. use of standard creative software packages) disenfranchises artists
- Rediscovering the pleasures of making and breaking own tools
- Question assumptions defined by existing tools.
- Ability to make tools vs. unquestioningly adopting the (digital) tools that are provided
- Recognising limitations (and use them in a positive inspiring way) toward evolving beyond them (limitations as precursor)
- Globally same digital tools result in similar aesthetic through predefined working patterns (resemblance, not innovation)
- Role of a designer shifts from that of a consumer of options defined by others to an active participant in evolving software
- Taking control of the tools and process helps to differentiate designers (by unique characteristics) and create new aesthetic possibilities
- Aesthetic of digital creations restricted by software functionality.
  Programming to expand functionality (open up new dimensions of creativity).
- Create (interactive) effects that are difficult or impossible to replicate using commercial software
- Authoring of own creative software to develop original creative expression

Automisation

- Free people from tedious jobs; automate tedious work
- Make repetitive tasks easier
- automise workflows
- automate existing processes

Education - Strategies & Methodologies

- Since relevant to certain fields of practice, it should be part of the range of tools being taught (like other tools for expression, e.g. languages, drawing)
- During intermediate period – before highly skilled students will enter HE – essential to build those skills
- Imagine and create something of their own, that is fun and surprising.
- Focus on initially teaching programming basics – afterwards, students move into different domains (then support them where needed)
- No point teaching specific platforms / environments as they are transitional (i.e. more important to learn basic concepts / methodology and how to adopt them to new environments)
- Similarities of digital art with video / sound art development since the 60s. (HE to stay up to date with the training of media art).
- Students recognise the value of this skill and when an institution doesn't support it they regard that as a negative outcome. (follow students' expectations)
- Programming can be applied to anything, which makes it more flexible than any other tool
- Coding is applicable to achieve results with diverse media
- It is not an aesthetic, a new radical approach to creativity, or a threat to any old ones – it is an incredibly useful tool.
- Discover that the thinking involved in computing can be as creative as making art, writing poetry, or directing movies

Awareness of Possibilities

- Benefits have to be communicated / exemplified
- Awareness of possibilities of creative coding and how this could expand one's set of artistic tools
- Awareness of technological limitations supports well informed decisions
- Learn about existence of software tools that work on different levels of abstraction / allow different kinds of access and flexibility
- Inspiration (contact to computer science and knowledge of the possibilities it offers) is key. Has to be 'fun, inspiring, and creative'.
- Computer as almost limitless creative / innovative tool requiring mental effort and improvisation Awareness of possibilities (to create own tools)
- Code is far more flexible than any (single, traditional) tool
- Creating computing art (progressing from only observing it) gives students a deeper understanding of the essential and impressive capabilities of computation.
- Promotion of creativity: Development of a mindset, the awareness of being able to 'create important things that the world interacts with'.

Hurdles & Challenges

- Possible knowledge gap between digital native students and tutors (relating to digital products and services)
- Tutoring essential to develop students' talents (vs. MOOCs good to learn specific skills)
- Requiring students to follow a particular line of enquiry can become their focus by default

Critique

Socio-Economical

- To fulfill policy of making art & design more 'valuable' in terms of the economical growth paradigm.
- Capitalism presents cultural production in the same vein as knowledge production (art as asset class, not as humanistic study).

Psychological

- Occupation with coding can have negative effect on art creation process (confronting mind sets / ways of thinking)

Thread to Humanities (conservation / retention of humanistic values)

- Should not be compulsive at arts academy; focus to be on theory (e.g. concepts, inspiration, research, choices of rendition)
- Focus on technology might not benefit society (possible expense to humanity)
- Is not necessary for meaningful expression – can be to advantage and to disadvantage. Coding just one skill of many

  → A sincere choice to not engage with programming reflects diversity; but is suspected to often be lead by feeling intimidated by the idea of learning to code.

Evaluative

- Culture of pigeonholing creatives associated with software / the technical
- Association with programming 'degrading' to technical role by unknowing / alienated creatives for fear of status loss
- (Fine Art) Artists should keep away from useful / utilitarian skills, as they will be taken for crafts- / trade-people (proclaimed separation of arts and the useful)
- Artistic / design traditions involve (and are defined by) manual skills.

  Positive: no dependence on (i.e. binding influence by) specific tool.

  Negative: self-distancing from technical solutions (seen as of minor value / of inferior status).

Epistemological

- Skillfulness blurs efficiency in conveying concepts
- Not knowing as advantage: enables thinking free of rules and expectations
- (knowledge can limit imagination) (artist's role: opinion / interpretation by the non-expert ?)

  → Beneficial to the growth of the artist vs. benefit of ignorance to others / society (e.g. children's naivety facilitates rethinking context, nevertheless they deserve education)