xCoAx 2023 11th Conference on
Computation, Communication,
Aesthetics & X
2023.xCoAx.org
Weimar, Germany

Seeing Programming Seeing: Exploring the Pedagogical Values of Functional Errors in Creative Coding

Jennifer Sykes
j.sykes@arts.ac.uk
Creative Coding Institute,
University of the Arts, London, England

Mick Grierson
m.grierson@arts.ac.uk
Creative Coding Institute,
University of the Arts, London, England

Rebecca Fiebrink
r.fiebrink@arts.ac.uk
Creative Coding Institute,
University of the Arts, London, England

DOI 10.34626/xcoax.2023.11th.29

Creative coding has been gaining momentum within Art Schools over the past two decades. However, as a discipline within Art Schools it is still relatively new as a creative pathway. As the use of code as a material for creative expression becomes increasingly prevalent, the methods for teaching creative coding within Art Schools have also begun to diversify. This paper presents a range of alternate teaching examples that emphasise reflective problem solving through error when teaching creative coding within Art Schools. The paper will review how these examples are implemented, the impact they have on student comprehension, and the future educational tools they support.

Keywords: Creative Coding, Functional Errors, Problem Solving, Art, Design.

1. Introduction

The approaches to teaching creative coding within Art Schools has often combined various methods from Computer Science, ranging from didactic lectures, peer coding and problem-solving activities. In more recent times, steps have been taken to explore "how aesthetic production or critical thinking can be cultivated and developed through learning to code" (Soon & Knotts 2019).

A common teaching method for creative coding is delivery of fundamental programming concepts via constructing creative examples or templates. In doing so, a student follows instructional steps, constructing their own copy of said creative output. An emphasis is placed on understanding basic concepts via screen-based outputs that contextualise a certain method for the student. However, a challenge of this approach is students are presented with a friction point when deviating from and iterating upon these examples independently for their own creative outcomes.

This paper discusses the use of mistakes and error as mechanisms that facilitate existing pedagogical methods when teaching the fundamentals of creative coding. In teaching students how to identify and edit Functional Errors in creative applications of code, this method lays the foundation for a broader range of practical, educational research tools that critically analyse teaching methodology within creative coding and expanding to Physical Computing.

1.1. Functional Errors

Functional Errors refers to instances of code that, although compiling without software compiler errors, result in different outputs than the users intended. Functional Errors can be found in both screen-based and physical computing environments. However, the purpose of this paper will focus on the context of screen-based graphical outputs.

This area of research is particularly relevant in an Art School environment, where creative coding practices exist within curricula of other established disciplines such as Fine Art or Visual Design. In mixing multi-disciplinary approaches to working, students often face "conflicted situations of practice" (Schön 1994) where a desired objective in code does not match the conceptual objective and vice versa.

As such it is noticeable that a Functional Error is also the result of a conceptual misunderstanding between a familiar material and new creative material (often code). "When teaching novices programming, misconceptions can occur" (Hermans et al. 2018) and when students are unfamiliar with the terminology in programming, it can often be difficult to identify the source of an error when using descriptive language more common place in other disciplines.

In framing Functional Error examples thematically within Art and Design vernacular it is hoped this can bridge a familiar language and new, unfamiliar language for students. In identifying and understanding Functional Errors the objective is the language used to explain an issue is developed in tandem with problem solving skills. This research aims to facilitate a reflective framework to recognise both technical and conceptual misunderstandings in students' practice-based Art School education.

2. Background

2.1. Existing Methods

The creative coding curriculum within Art Schools is undergoing a shift away from traditional lecturer-centered instructional methods, towards a more contextually and thematically integrated approach to programming. There has been an acknowledgement of those "who prefer to work improvisationally, instead of following formulas; and aim to create things that are expressive rather than utilitarian" (Buechley 2012).

An important component to conceptualising creative coding activities has been incorporating "Active Learning" (Bonwell & Eison 1991) strategies. These activities not only engage students, but also enhance their comprehension skills by encouraging them to actively participate in "doing things and thinking not just listening" (Jung et al. 2021).

Whilst Active Learning strategies promote more doing things and thinking, rather than simply imposing content on a learner through direct instruction, the retention of crucial information may still be limited to the specific context in which it is learned. In adopting a more Constructionist position emphasis can be placed on how "the *laws of learning* must be about how intellectual structures grow out of one another and about how, in the process, they acquire both logical and emotional form" (Papert 1980).

When transitioning from Constructivists to Constructionist methodology the application of visual programming environments has facilitated a move towards a high-level understanding of programming structure via modular building blocks. In his analysis of the Turtle, Papert compares the building blocks of "Learning Math by talking to Turtles is like learning dancing by dancing with people while learning math by doing pencil and paper sums is like learning dancing by rote memory of pencil and paper diagrams of dancing *steps*" (Papert 1976).

The building blocks of Visual programming environments have been found to enhance the cognitive association between programming terminology and their associated operations. However, they often do not answer the issues that “pertain to their inability to help users build a skillset that can be transferred to other programming environments and paradigms and their inability to be extended through new components” (Hansen 2019). As such visual blocks can fall short in addressing the conceptual structure of code, leading to the need for additional supplementary material for effective learning. The purpose of incorporating Functional Errors into the educational process is not to supersede such pedagogical tools, but rather to augment them, creating further steps and pathways that support students in their learning journey encouraging self-reflection and evaluation.

2.2. Problem Solving

The teaching of problem-solving skills is an integral part of computer science in promoting a deeper understanding of fundamental concepts. Studies have revealed that a lack of such approaches resulted in students limiting their understanding and “rather than *getting the big picture* of computer science, they narrow their focus to *getting this program to run*” (Allan & Kolesar 1996). This holds true not only in the field of computer science but equally when creatively working with code in adjacent disciplines. Given that students in the field of creative coding often approach the subject as novice programmers yet possess university-level knowledge in their accompanying disciplines, there is significance in including repetition allowing students to “actively engage with the content, work through it with others, relate to it through an analysis” (Fee & Holland-Minkley 2010). By incorporating teaching examples that intentionally include such problem-solving strategies, it encourages creative agency and support future self-exploration.

2.3. Reflection

As Schön identifies, often students “can deliver without being able to say what (they) are doing” (Schön 1987, 23). In providing practical programming examples for Reflection In Action it enables a dialog between student and process that helps them identify where misconceptions, error and miscommunication have occurred. The Functional Error examples are disseminated via a website that includes embedded, interactive programming sandboxes. Graphical preview windows offer visual feedback of the code that students edit offering real-time prompts to reflect upon each practical step they have undertaken. In doing so, the examples draw influence from Schön’s theory.

> A designer sees, moves, and sees again. The designer sees what is “there” in some representation of a site, draws in relation to it, and sees what he or she has drawn, thereby informing further designing. This process of seeing-drawing-seeing is one kind of example of what I mean by designing as a reflective conversation with the materials of a situation. (Schön 1991, 133)

3. Non-Linear Process

Art School education centres on the Studio based, practice driven culture with a focus on learning through process-driven methods encouraging an “experimental attitude, precisely of making trials, of learning from experience by prompting problems and failures” (Schnapp & Shanks 2009). It is important to frame the use of Functional Errors within this context. In shaping lessons around Functional Errors the mode of delivery supports cyclical reflection fostering future abilities to explore and experiment independently.

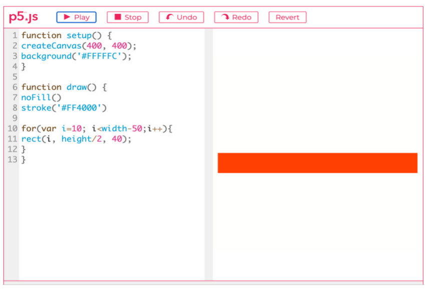
“Project-based learning is a key pedagogical approach in the arts” (Brain & Levin 2021, 6), encouraging students to explore far greater expressive adaptability. However, the integration of context within creative coding templates, the delivery of teaching exercises often remains centered around the original intended outcome. The ability to re-frame linear paths and identify why navigating left, right, forwards, and backwards causes a change in outcome can become difficult for students when developing their own creative coding work. In discussing Casual Creators Compton and Mateas observe “many creative tools exist to support task-focused creativity, but in recent years we have seen a flourishing of autotelic creativity tools, which privilege the enjoyable experience of explorative creativity over task-completion” (Compton & Mateas 2015).

The acquisition of creative flexibility in the artistic process often requires years of practice. It can be challenging to foster these skills during the initial stages of programming, when the foundations are still being established. Turkle compares the experiences of those learning to program to “those of the bricoleur scientist or mathematician. Bricoleurs construct theories by arranging and rearranging, by negotiating and renegotiating with a set of well-known materials” (Turkle 1990). In pairing creative disciplines with the computer sciences, Art Schools are encouraging the development of distinct knowledge and understanding across specialisms.

4. Study Method

The series of Functional Error examples are intended to provide diagnostic and reflective dialogue of observing, programming and observing again (i.e., seeing — programming — seeing). The purpose of this approach is to uncover patterns within the examples enabling students to broaden their conceptual understanding of earlier, limited templates. In equipping students with interfaces for reflecting upon mistakes, this study explores the relationship between whether, in using of Functional Error examples, does this enhance students' troubleshooting confidence, and ultimately lead to greater flexibility in their creative practices?

The method this study undertook focused on prescribing three graphical examples that explore the concept of Functional Errors within the programming language P5JS. P5JS is a JavaScript library designed to support creative coding "with a focus on making coding accessible and inclusive for artists, designers, educators and beginners" (McCarthy 2014).



```
function setup() {
createCanvas(400, 400);
background('#FFFFFC');
}

function draw() {
noFill()
stroke('#FF4000')

for(var i=10; i<width-50;i++){
rect(i, height/2, 40);
}
}
```

Figure 1: P5JS sandbox editor.

In this study we utilise the P5JS web editor which offers an integrated sandbox environment for programming with minimal set-up. The web-editor offers participants a user-friendly interface that facilitates real-time code editing while synchronously previewing the graphical output produced as a result of any modifications made to the text. The editor is embedded into a general teaching webpage providing information that is easy to disseminate, thereby removing the barrier of installation and configuration [Fig. 1]. The primary objective of this study is understanding and examining each Functional Error that reiterates programming methods students have already encountered in previous classes.

Participants consisted of 14 adults (7 Female, 3 Male, 2 Non-Binary and 2 undeclared) who were enrolled in a year-long Diploma in creative computing. Participants had two previous years' experience in an adjacent Art and Design discipline. These participants had prior experience in a related discipline in Art and Design and had completed a module in creative coding in the three months leading up to the study. It is noteworthy that prior to the start of the Diploma program six months ago, the participants had limited to no programming experience. Therefore, this group provides an opportunity to evaluate their retained knowledge in the field of creative computing.

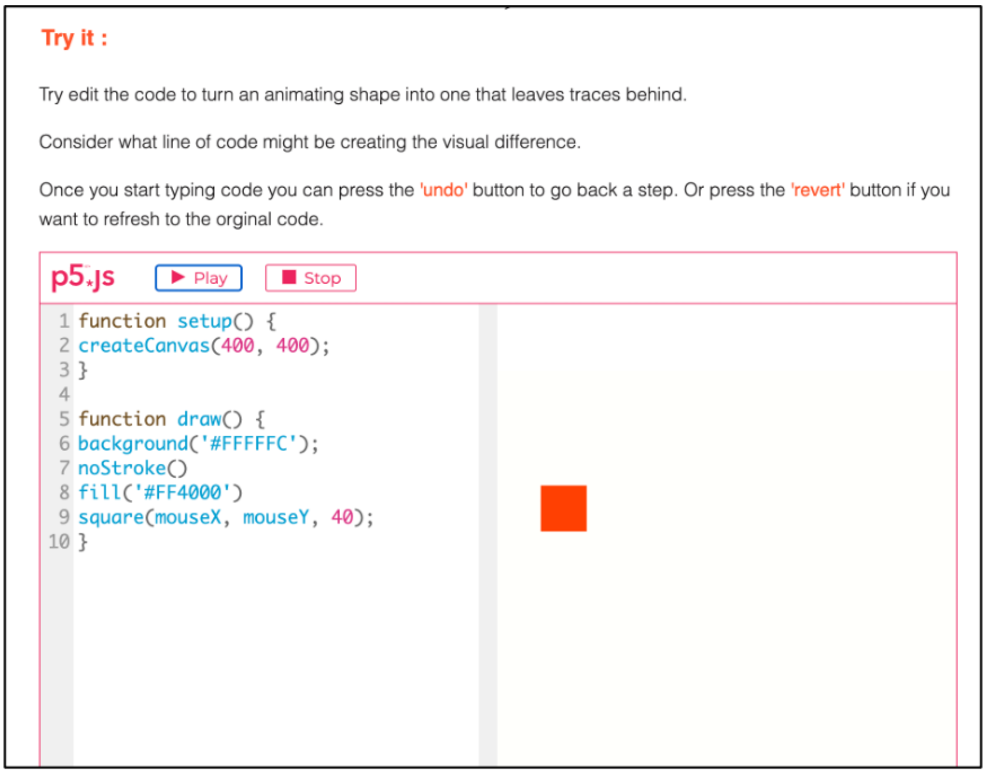
4.1. Examples

The three examples were selected through a systematic observation of the recurring Functional Errors made by students while teaching creative coding across various higher education institutions.

Each participant was presented with two animated visual outputs per Functional Error; the intended visual outcome and the actual visual outcome containing the Functional Error. The participants were presented with a web-editor containing the Functional Error and were then challenged to resolve each instance so that it reflected the intended visual outcome.

The three Functional Error examples consisted of:

A. Drawing: A common scenario students face when approaching programming from a Visual Design background is a familiarity with the affordances and signifiers of animation. An understanding of timelines and editing workflows can often result in assumptions of how a graphical output is generated. In this example, participants are presented with code that takes mouse position as an input control for the movement of a graphical square. The square refreshes every positional change and thus visually mimics animation. Participants have the task of modifying this code [Fig. 2] so that the animated graphical square leaves a trail of its movements on the screen, mimicking the act of drawing on paper [Fig. 3]. Importantly, neither version is syntactically wrong. However, the conceptual meaning and intended outcome of the graphical representation may differ based on the students' intentions. In resolving this Functional Error, students achieve an understanding of the `setup()` and `draw()` functions within the P5JS language, which are core components in understanding the structure and running order of code.


Try it :

Try edit the code to turn an animating shape into one that leaves traces behind.

Consider what line of code might be creating the visual difference.

Once you start typing code you can press the 'undo' button to go back a step. Or press the 'revert' button if you want to refresh to the orginal code.

```
function setup() {
createCanvas(400, 400);
}

function draw() {
background('#FFFFFC');
noStroke()
fill('#FF4000')
square(mouseX, mouseY, 40);
}
```

Figure 2: Example A: Drawing. The web-editor code interface participants interact with.

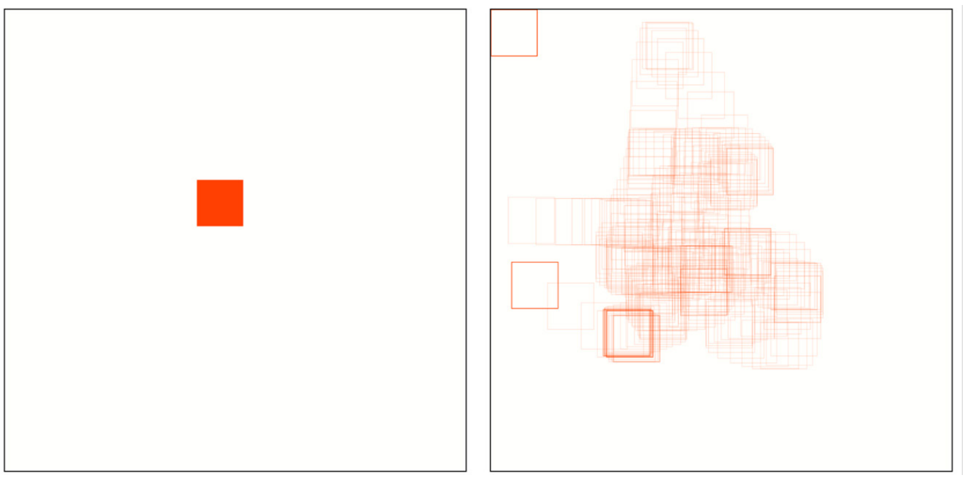


Figure 3: Example A: Drawing.

B. Iteration: The second example presents a visual output resembling a solid rectangular shape. However, the title and intended output indicate the shape should reflect several squares spaced out visually on the canvas [Fig. 4]. Participants are presented with code comprised of for loops, not necessarily required for drawing one singular rectangle as the present visual displays. Participants are required to decipher 1) why a for loop may be required and 2) why several iterative shapes are not displayed on the canvas. In resolving this Functional Error, participants gain an understanding for concepts such as conditions within for loops and mathematical iteration.

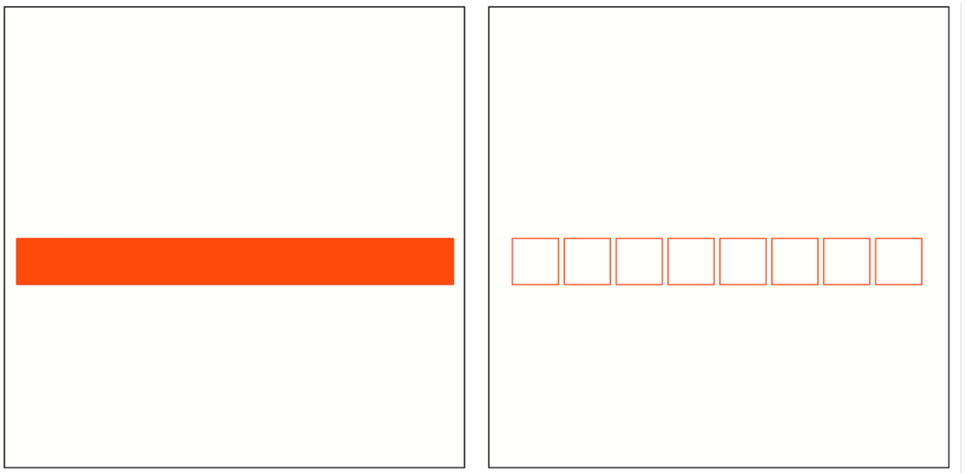


Figure 4: Example B: Iteration.

C. Order: Lastly, the third example presents a visual representation of two graphical squares. The objective of the intended output is to display three squares, each with a distinct coloured outline. However, the actual outcome demonstrates three squares with the same solid colour. Participants are presented with the challenge of identifying how to modify the code to distinguish between the solid colour of each shape and the outline colour [Fig. 5]. In doing so, participants begin to understand the programming order and sequence. This concept is often misunderstood by Art and Design students, who tend to reverse the order, drawing parallels with layer manipulation in creative production tools.

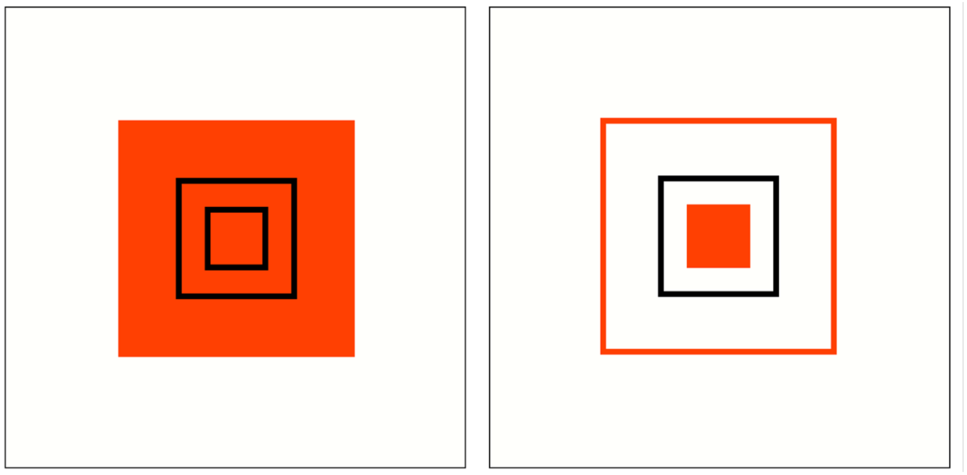


Figure 5: Example C: Order.

When presented with each of the three modifiable examples, participants were asked to reflect and answer to themselves in their own descriptive language (not necessarily programming terminology):

— Can you describe what you are seeing?
— What would you like to see happening?
— Why might it not be doing that?
— Where do you think the issue is?
— Can you identify what to change to make the visual display as desired?
— Why did that resolve it?
— Can you say in technical terminology what and why the change worked?

In allowing participants to verbally answer questions themselves in descriptive language they are comfortable with, it allows them to navigate towards to the correct answer without the barrier of not knowing if they are using correct terminology. After participants have identified they are on the correct path the final question prompts them to narrow their language to technical terminology cementing the foundations of the P5JS language constructs and the functions they may use in editing the example code.

Prior to undertaking this study, data was captured via a questionnaire [Table 1] that gathered information assessing their demographics, self-efficacy in creative coding, troubleshooting, creating conceptual art and creating computational art. Each participant self-assessed their self-efficacy before and after the workshop on 5-point scales (from (1) unconfident to (5) confident). Alongside this analysis, participants were asked to assess the usefulness of using the Functional Error examples in their educational experience both quantitively on 5-point scales (from (1) not at all to (5) a lot) and qualitatively in written responses.

Table 1: Study Questionnaire Questions. (the questionnaire included questions for a broader evaluation of Creative Computing tools and as such these questions and question numbers have been omitted from this table)

Question	Questions asked *before* study
Q1	How confident are you in using P5JS?
Q3	How confident are you in troubleshooting code?
Q5	How confident are you in creating conceptual Art and Design?
Q6	How confident are you in creating Computational Art and Design?
Q7	What do you anticipate the biggest challenge of working with creative coding to be?
Question	**Questions asked *after* study**
Q8	How confident are you in using P5JS?
Q11	How confident are you in creating conceptual Art and Design?
Q12	How confident are you in creating Computational Art and Design?
Q13	Has your biggest anticipated challenge changed from before the workshop and if so, what is it now?
Q14	Do you find the Graphical Functional Error examples useful?
Q15	How confident are you in troubleshooting code?
Q20	Do you have any additional comments regarding Functional Errors and this teaching method?

5. Results

5.1. Quantitative Results

Prior to undertaking this study, the results indicate a relatively low level of self-assessed confidence among participants in level across programming and troubleshooting (Questions 1 and 3). The mean score for participants' confidence in programming P5JS was 2.85 with 38.5% electing a score of 2 (indicating a little unconfident) and 46.2% scoring a 3 (rating themselves fair in confidence). Only a small number of participants, 2 individuals, rated themselves as a little confident or confident in programming P5JS.

These initial quantitative results align with the participants' limited programming experience in only partaking in one introductory module in creative coding prior to this study. These findings provide support for the notion that, despite having practiced these skills, there is a potential disjunct between the absorption of knowledge and the application of it in practice.

Analysis of question six reveals participants demonstrated a high level of confidence in their abilities to create conceptual Art and Design (mean = 4.08) [Table 2] with an interesting observation that no participants rated themselves below fair in this area [Figure 6]. Participants scored marginally lower level of confidence in question seven when asked about creating computation Art and Design (mean = 3.69). However, it is worth mentioning that 82.3% rated themselves as fair or above in this area. These results correlate with the participants' existing knowledge obtained in their respective Art and Design specialisms.

Figure 6 illustrates the distribution of ratings per question, as a percentage, prior to the initiation of the study. Table 2 shows the mean scores for each question.

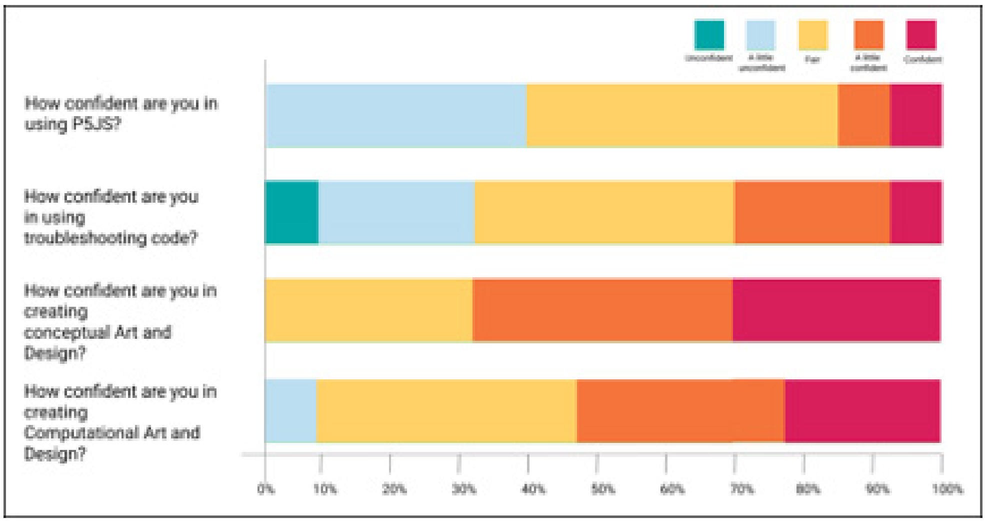


Figure 6: Stacked Bar Chart Results of Questions asked before the study.

Table 2: Mean Statistics of quantitative question results.

	Q1	Q3	Q5	Q6	Q8	Q11	Q12	Q14	Q15
Mean	2.85	3.00	4.08	3.69	3.15	4.00	3.92	4.31	3.31
N	13	13	13	13	13	13	13	13	13
Std. Deviation	.899	1.080	.862	.947	.801	.816	.954	1.182	1.032

Following the study, participants assessed their confidence in troubleshooting code (question 15), with a slight increment from their pre-study evaluation. The majority of participants rated themselves as fair (mean = 3.0) followed by 23.1% as a little unconfident and 7.7% unconfident.

Figure 7 illustrates the distribution of ratings per question, as a percentage, after the initiation of the study. The results post study indicated a general increase in participants' confidence in both question 8 regarding P5JS programming (mean = 3.15) and question 15 regarding troubleshooting code (mean = 3.31) with a positive reduction of the proportion of participants who self-assessed their confidence as a little unconfident to unconfident. Falling from 38.5% to 15.4% when asked to self-assess confidence in programming P5JS. Similarly, the proportion of participants in the lower confidence categories for troubleshooting code fell from 30.8% to 15.4% with the majority (84.6%) now rating themselves fair to confident.

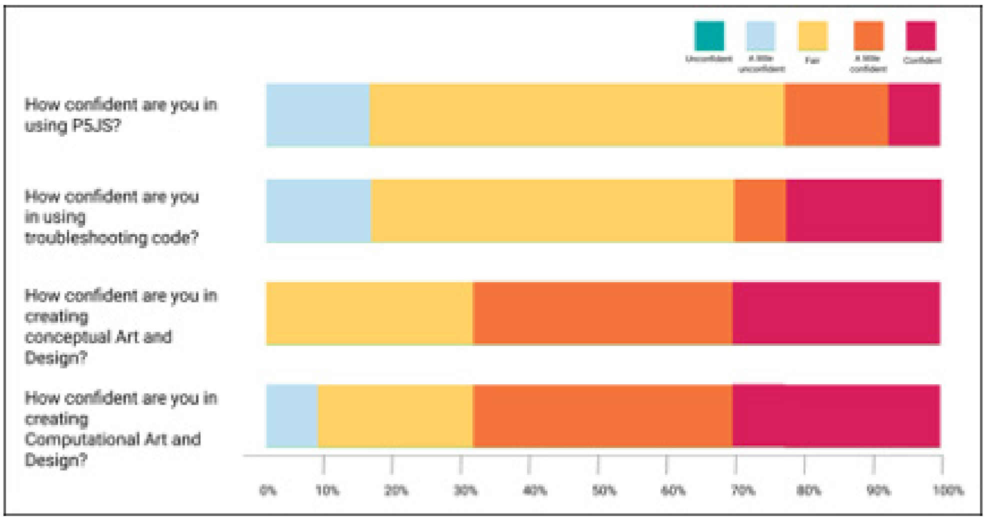


Figure 7: Stacked Bar Chart Results of Questions asked after the study.

Understandably, the results revealed the confidence rating in creating conceptual Art was unchanged after the study as this aspect was not addressed in the content. However, there was a small improvement in participants' confidence when creating computation Art and Design, as indicated by the increase in the mean score from 3.69 to 3.92. An interesting impact was observed among participants who previously rated themselves as fair, now rating themselves as a little confident to confident. This observation aligns with the results of question fourteen as illustrated in Fig. 8; how useful did you find using the graphical Functional Error examples? Participants rated the examples overwhelmingly useful (mean = 4.31) with a substantial majority (84.6%) rating the examples as a little to very useful.

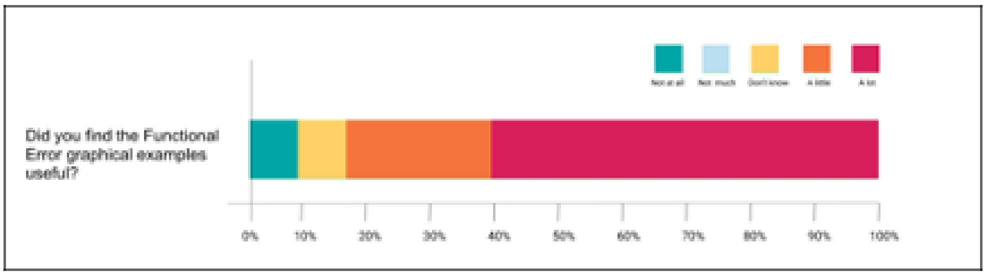


Figure 8: Stacked Bar Chart Results of Questions asked after the study.

5.2. Thematic Results

The quantitative data was accompanied by a set of thematic questions [Table 3] aimed at supplementing the statistical results.

Table 3: Thematic qualitative questionnaire answers.

Participant ID	What do you anticipate the biggest challenge of working with creative coding to be?	Has your biggest anticipated challenge changed from before the workshop and if so what is it now?	Do you have any additional comments regarding Functional Errors and this teaching method?
P1	Getting the visuals to behave how I envision it to.	No	n/a
P2	p5.js	p5.js, feels less natural to me than Arduino	n/a
P3	Technical problems	Technical problems and There is very little growth in computer art about art.	Add some training on the aesthetic aspects of computing. Teaching methods should be more diverse and not just teaching software alone
P4	Translating ideas to code	It is still the code but understanding how to simplify it has increased	n/a
P5	Making something work the way you intended, trail and error and finding solutions	Nope	This teaching method was very useful in understanding the code and how it works.
P6	processing	p5js	n/a
P7	The coding frame of mind is a lot more rigid than the freeform art mind. It takes some adapting to learn the different principles of coding.	No. I think that my biggest perceived challenge is something that takes some time to change and overcome.	Functional errors re-frames coding in a different way, to be a different kind of puzzle, providing a different perspective.
P8	Feeling the difference between searching for the answers and code and thinking of the code myself and figuring out my errors	Slightly, my way of thinking about it did	Maybe have two tutors around as the time is short and lots of us need help
P9	Applying my concept-based art practice to displaying creative comp work	I think my horizons on what I could expect when conceptualising creative computing displays have been broadened.	n/a
P10	Learning to code	Not really as it was only 1 session, but I feel having a whole term of these would making me A LOT more confident with doing code	I think it was really great compared to the lessons I have had, I felt like I learned a lot more in one lesson than I did in multiple lessons previously.
P11	My code not running and not knowing how to fix it	I would still be worried about not being able to fix the code but I would at least attempt to do so myself first	I thought working through the p5js problems was really useful
P12	Technology	Technology	n/a
P13	Realising where certain creative code would fit in a project	No	n/a

One of the questions posed to participants prior to the study was what did they anticipate the biggest challenge of working with creative code to be? A selection of responses from participants reflects the discursive arguments in this paper:

— “Translating ideas to code.” (P4)
— “Feeling the difference between searching for the answers and code and thinking of the code myself and figuring out my errors.” (P8)
— “Making something work the way you intended, trial and error and finding solutions.” (P5)

When participants were asked if their biggest anticipated challenged had changed post workshop, the majority of thematic responses remained within the context of troubleshooting P5JS code. However, a small number answered differently, expanding upon this context, and providing additional insights:

— “It is still the code but understanding how to simplify it has increased.” (P4)
— “I feel having a whole term of these would making me A LOT more confident with doing code.” (P10)

Lastly, participants were asked to respond if they had any additional comments regarding the use of Functional Errors as teaching methods. A selection of answers included:

— “I felt like I learned a lot more in one lesson than I did in multiple lessons previously.” (P10)
— “Functional errors reframe coding in a different way, to be a different kind of puzzle, providing a different perspective.” (P7)
— “This teaching method was very useful in understanding the code and how it works.” (P5)

6. Discussion

Often code that results in Functional Errors is opaque in its nature, making it difficult for students to identify the underlying cause. In providing students with classroom opportunities to explore patterns of programming structure that produce different conceptual intended and unintended visuals it offers a basis for developing confidence in their own understanding and communication skills.

The participants’ thematic answers support a hypothesis that participants require supportive pathways to effectively utilise code in an autonomous direction. The approach to teaching the fundamentals of programming structure through reflective acceptance of mistakes and error encourages process and pattern observation within existing task specific examples. In doing so, this iterative learning process replicates a material investigative approach familiar to students in Art School and facilitates their navigation of current aesthetic and contextual project templates.

Early thematic analysis pointed towards the disconnection between applying creative coding methods independently when no longer in the context specific to a technical example. The positive impact on participants' confidence from those who rated themselves low and those who already felt a little confident, demonstrates this reflective teaching method was beneficial in facilitating understanding for all parties. The thematic answers regarding the use of the Functional Error teaching examples support the speculation that the use of this diagnostic approach to teaching has a positive impact on the relationship between disseminating knowledge and its practical application.

The current collection of examples encompasses both thematic titles (e.g.: Drawing) and descriptive titles (e.g.: Iteration). The drawing example establishes a clear relationship to presumptions derived from workflows of analogous disciplines, whilst identifying the differences in computing. Future implementations aim to adapt similar thematic approaches that align with the relational contexts commonly found in the fields of Art and Design.

The focus of this study was primarily on the implementation of creative coding in screen-based visual environments. However, the presence of Functional Errors spans a range of programming applications. The field of physical computing represents a significant potential source for miscommunication in the visual programming environment, electric circuitry, and physical capabilities of components. Considering this, future research is aimed at developing the practical representation of Functional Errors in the field of Physical Computing.

7. Conclusion

This preliminary study was relatively small in its scope. Further research is needed to broaden the participant population and increase the sample size, thereby bettering our understanding of the benefits of this pedagogical approach. The limited sample size of the current study hinders the ability to establish a statistically significant correlation between pre- and post-study quantitative results. Whereas an expanded study would allow for the implementation of A/B testing, where participants would be divided into groups, some utilising Functional Errors and some not. This would facilitate statistical analysis, such as Mann Whitney U tests, and provide a more robust evaluation of the Functional Error examples. That been said, early indications suggest the implications of using reflective and troubleshooting techniques in the initial stages of teaching programming holds positive potential.

References

Allan, Vicky. H., M. V. Kolesar. 1996. *Teaching Computer Science: A Problem Solving Approach that Works*. National Science Foundation, Arlington, VA.

Bonwell, Charles. C., James. A. Eison.1991. *Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports. ERIC Clearinghouse on Higher Education.* The George Washington University, One Dupont Circle, Suite 630, Washington, DC 20036-1183.

Buechley, Leah. 2012. *Expressive Electronics; Sketching, Sewing and Sharing* (lecture wats:ON? Festival, Carnegie Mellon University, Pittsburgh, PA).

Compton, Kate, Michael Mateas. 2015. *Casual Creators: Expressive Intelligence Studio*. University of California, Santa Cruz.

Fee, Samuel B., Amanda M. Holland-Minkley. 2010. *Teaching Computer Science through Problems, not Solutions*. Computer Science Education, 20, 129-144 (2010).

Hansen, Stig Møller. 2017. *Deconstruction/ Reconstruction: A Pedagogic Method for Teaching Programming to Graphic Designers*. Department of Digital Design and Information Studies, Aarhus University, Denmark.

Hansen, Stig Møller. 2019. *Danish University Colleges public class Graphic_Design implements Code { // Yes, but how? } an investigation towards bespoke Creative Coding programming courses in graphic design education.* Aarhus University.

Hermans, Felienne, Alaaeddin Swidan, Efthimia Aivaloglou, Marileen Smit. 2018. "Thinking Out of the Box: Comparing Metaphors for Variables in Programming Education." In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education (WiPSCE '18)*. Association for Computing Machinery, New York, NY, USA, Article 8, 1–8. https://doi.org/10.1145/3265757.3265765

Jung, Andrew, Zhuojun Duan, Ingrid Russell. 2021. "Active Learning Strategies: A Computing Course for Undergraduates." *16th International Conference on Computer Science & Education (ICCSE)*.

Levin, Golan, Tega Brain. 2021. *Code as a creative Medium A handbook for Computational Art and Design*. The MIT Press.

McCarthy, Lauren. 2014. *P5JS mission statement*. https://p5js.org/

Papert, Seymour A. 1976. *Some Poetic and Social Criteria for Education Design*.

Papert, Seymour A. 1980. *The Gears of My Childhood. Mindstorms; Children, Computers, and Powerful Ideas.* Basic Books.

Schön, Donald A. 1987. *Educating the Reflective Practitioner.* USA: John Wiley and Sons.

Schön, Donald A. 1991. *The Reflective Practitioner: How Professionals Think in Action*. Taylor & Francis Group.

Schnapp, Jeffrey T., Michael Shanks. 2009. *Artereality (Rethinking Craft in a Knowledge Economy: Art School Propositions for the 21st Century).* The MIT Press.

Soon, Winnie, Shelly Knotts. 2019. "Aesthetic Coding: Exploring Computational Culture Beyond Creative Coding." *International Symposium on Computational Media Art (ISCMA) 2019*. School of Creative Media City University of Hong Kong.

Turkle, Sherry. 1990. "Epistemological Pluralism: Styles and Voices within the Computer Culture." *Signs: Journal of Women in Culture and Society*. 16, No. 1, From Hard Drive to Software: Gender, Computers, and Difference. (Autumn, 1990), pp. 128-157.

Young, David. 2021. "Theorising while() Practising: A Review of Aesthetic Programming." *Computational Culture* 8. Royal Holloway University of London. http://computationalculture.net/theorising-while-practising-a-review-of-aesthetic-programming