Digital Fabrication as a Tool for Increasing Student Engagement in Introductory Programming Courses

Maria Nanabhai
Department of Computer Science
University of Cape Town
Cape Town, South Africa
nnbmar003@cs.uct.ac.za

ABSTRACT

Computer programming is an increasingly in-demand skill in modern society, but it is considered difficult to learn and students often become disinterested in pursuing it. Engagement in the subject is a counter to this, affecting performance, understanding, and retention. Many interventions to improve student engagement have been investigated. In this review, we examine one such intervention: digital fabrication. This approach has been widely implemented, using tools such as embedded computing devices, laser cutters, and 3D printers to create physical artefacts. We review a number of these implementations and find that they have achieved significant success in boosting student motivation, autonomy, and engagement, despite facing unique challenges compared to conventional approaches. We recommend further research into long-term implementations in formal courses, as well as investigation of 3D printing as an avenue because this has not yet been studied in much detail.

CCS CONCEPTS

• **Social and professional topics** → **CS1**; • **Physical computing** → *Digital fabrication*; • **Human-centered computing** → User interface programming.

1 INTRODUCTION

Information and communications technologies (ICT) have become ubiquitous in modern society, playing increasingly important roles in the economy as well as across all disciplines. However, the growth of individuals with computing skills has not kept up to pace with the growth of technology, creating a large skills gap, especially in developing countries such as South Africa. Educational institutions and governments thus consider it a high priority to promote studying ICT, and more especially computer programming, both by making it more appealing to learners to improve enrolment rates, and by improving success rates [5].

This is a challenge given that programming is considered a difficult skill for novices to learn, which intimidates many students [35]. The computer science (CS) major has some of the highest drop-out rates across all disciplines – it is estimated that fewer than 40% of students enrolling for a computer science major persist to the completion of their degree [39]. The approaches presently used in courses do not effectively counter this – a study conducted at the University of Witwatersrand found that many students become less positive about working with computers after completing an introductory CS course, a finding in line with other research [24].

Various interventions have been investigated to address the problem of sustaining student interest in CS. This review presents the literature on one such approach that has enjoyed substantial success – creative coding as implemented through digital fabrication.

We begin by introducing the main difficulties in computer science education, in particular, the challenge of student engagement. We also outline the theoretical framework underlying this review. Then, in section 3, we present the creative coding paradigm and the maker movement, and analyse some implementations of this approach. Finally, we examine the viability of the specific instance of creative coding that is relevant to our project: 3D printing.

2 COMPUTER SCIENCE EDUCATION

2.1 Importance of Student Engagement

Many factors impact the likelihood of student retention in a CS course. These include the perceived gains from the course, a supportive environment, personal values and emotions [25], interest, expectations of the course, previous programming experience, and performance [33]. Psychological factors thus comprise a large proportion of motivators for retention. Biggers, Brauer & Yilmaz [16] compared graduating CS students with those who chose to leave the major, and identify five core challenge areas affecting student perception of computer science:

(1) Loss of interest due to limited understanding of the “big picture” of CS
(2) Lack of relevance of coursework
(3) Perceived asociality in the subject
(4) Lack of familiarity with CS and lack of confidence
(5) Perception of coursework as boring and pointless

These challenges speak to a lack of intrinsic motivation towards computer science. Greater autonomy in motivation is associated with higher levels of engagement, quality of learning, and performance [47]. It is especially important in tasks that emphasise performance quality over quantity – i.e. tasks of higher complexity that require deeper engagement with the material, as well as abstract tasks, such as computer science [19]. Contextualisation, personalisation and offering choice are some of the most effective strategies at encouraging intrinsic motivation for abstract tasks [21]. It is then no surprise that the recommendations of Biggers et al [16] for most of these factors involve contextualising coursework within real life and presenting computer science as being about more than just coding.

Factor 4 aligns with performance in the course. However, several studies show that motivation and performance are deeply intertwined – higher motivation leads to higher performance, and vice versa [33][15]. Thus, one of the most important aspects of CS education to address is increasing student engagement and motivation, especially by placing programming in a larger context.

2.2 Theoretical Background: Constructivism and Constructionism

Constructivism is a theory of learning that says that people learn by experiencing things in the world, reflecting upon them, and then integrating them into their existing ideas in order to construct their understanding of the world [13]. It suggests that learners can best derive meaning from material by directly experiencing it in the world. The most important idea for the present review is that learning is active rather than passive.

Honebain [13] says that constructivist learning environments should aim to embed learning in realistic contexts, encourage students to have a voice and ownership of the learning process, and to encourage multiple modes of representation. These goals are similar to the strategies identified in 2.1 Constructivist learning environments also emphasise the importance of process over product, pursuit of students' own interests and learning as an interactive process.

Seymour Papert [41] takes constructivist theory a step further to formulate a learning approach known as 'constructionism'. Papert says that the mental construction of knowledge happens most effectively when the learner is consciously involved in constructing an actual entity – learning by creating. This approach is manifested in CS education through (a) the *Creative Computing* approach, and (b) the incorporation of ideas from the Maker Movement, including a physical computing approach.

3 CREATIVE COMPUTING AND MAKING

3.1 Creative Computation

The most direct realisation of constructionism is the programming language Logo [6], developed by Papert himself in 1967. The environment consists of an area for entering text commands, and an area for a graphical turtle that moves around according to the entered commands. The path of the turtle traces a line onto the screen, and students can use this to draw a variety of pictures. This provides both a visual representation of the commands written, and an avenue for students' creativity. Educational programs following the Logo philosophy and using the language across over 250 classrooms were found to have dramatic improvements in student engagement, confidence, problem-solving abilities, and motivation, among others [49].

Since then, a great number of visual and creative computing tools have been developed for the teaching of introductory programming. Contextualising computation as a creative medium itself, many technologies extend the basic Logo turtle concept with added functionality, allowing students to create a broader range of aesthetically pleasing designs through simple code. Processing [23] is a programming language designed to facilitate this. We discuss the Processing project and a curriculum designed around these principles in section 3.3.2.

Other approaches, such as visual programming tools that allow a student to create a narrative, are also popular and successful. Scratch [4] is a visual, drag-and-drop programming environment derived from Logo. It allows students to create stories, games and animations, and has become one of the most popular tools for CS education [40]. It has also been found to have a high degree of success in supporting understanding CS concepts such as loops and conditionals, computational thinking, and engagement, across several contexts [37][58][44]. Alice [7] is a similar environment that extends this concept to three-dimensional objects as a means of teaching object-oriented programming.

Media computation is another form of creative computing that has had considerable success in improving student engagement and performance. It involves manipulating images, audio and other forms of digital media programmatically. Digital media is pervasive in modern life, and students feel that media computation helps them gain a deeper understanding of these common technologies, placing computing in a context that is directly relevant to their lives. Further, media computation allows for a high degree of complexity in creations. A review on interventional approaches found it to be the most effective approach at improving pass rates, better than course or assessment changes, games, or the introduction of a preliminary course, among others [51].

Our main concentration is the digital fabrication part of creative computation. This can involve the creation of digital artefacts such as those described above, as well as of physical objects. The latter has been suggested to have a stronger effect on students' self-esteem. One student observes after attending a making session, "It wasn't 'school stuff,' it was the 'real thing'" [52]. We next examine this through the lens of the Maker Movement.

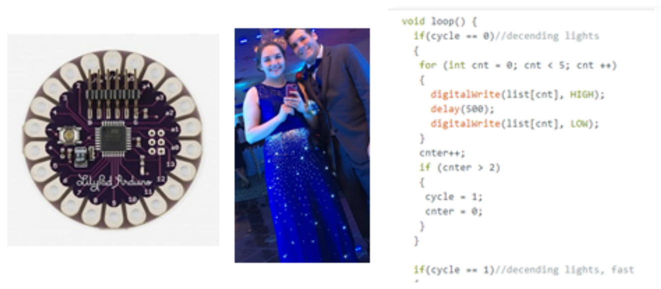
3.2 Maker Movement

Making, in this context, has been defined as "a class of activities focused on designing, building, modifying, and/or repurposing material objects, for playful or useful ends, oriented toward making a 'product' of some sort that can be used, interacted with, or demonstrated." [36] Central to the philosophy of the Maker Movement is the empowerment of the individual over the technology in their lives rather than being a passive user. The movement has grown with the increased availability of technologies that allow for new ways to interact with physical materials via computing, such as 3D printing, embedded computing tools such as Arduinos, and augmented or virtual reality devices.

A review by Papavlasopoulou et al. [40] investigating the applications of Maker Movement ideas to education found that there has been widespread interest in this approach, primarily in the computer science, mathematics and engineering disciplines. Most studies found that making sessions improve participants' competence and engagement, as well as expanding their perceptions of the subject. Moreover, making boosts the self-efficacy of participants, which brings indirect improvements in confidence, interest and engagement.

3.3 Case Studies

3.3.1 Lilypad Arduino. The Lilypad Arduino [1], a fabric-based embedded computing kit for developing wearables, is one of the most commonly-studied technologies for making in education [40]. It is also a form of making that emphasises creativity more than



```
void loop() {
  if(cycle == 0)//decending lights
  {
   for (int cnt = 0; cnt < 5; cnt ++)
   {
     digitalWrite(list[cnt], HIGH);
     delay(500);
     digitalWrite(list[cnt], LOW);
   }
   cnter++;
   if (cnter > 2)
   {
    cycle = 1;
    cnter = 0;
   }
  }

  if(cycle == 1)//decending lights, fast
```

Figure 1: (a) The Lilypad Arduino microcontroller [2], (b) an example of a dress designed using the Lilypad, (c) example code [8]

conventional robotics does. This allows us to examine an application of creative computation and making to CS education in depth.

The Lilypad Arduino system consists of a fabric-mounted microcontroller, sensors and actuators, and conductive thread, which can be sewn into clothing to create a programmable and interactive ‘e-textile’. The Lilypad is programmed through Arduino Software (IDE), which is an open-source development environment for microcontrollers.

Most educational applications of the Lilypad have consisted of small co-curricular workshops with middle and high school students. They also frequently focus on the potential of this tool to expand diversity in computer science classes, as fashion and creative pursuits are preferred by girls compared to more ‘technical’ classes. This needs to be kept in mind when considering the generalisability of the results that follow.

Buechley et al. [17][18] pioneered the work in this subject. Their preliminary workshops resulted in students who were highly enthusiastic about e-textiles, but struggled and were frustrated by the programming aspects. They suggest that this could be explained by the programming language and environment used being insufficiently user-friendly. Indeed, later workshops using an improved IDE found considerable improvements. Other work using Modkit [12] – a Scratch-like visual programming environment for Arduinos – also finds that students’ comfort with, enjoyment of, and intention to learn programming improve after Lilypad workshops [43]. Studies using later, more refined versions of the Arduino IDE also report similar effects [30][29]. This highlights the importance of user-friendly programming environments for digital fabrication.

There are also challenges in bridging the gap between software and the physical product. Significant planning and understanding of the hardware component is required, as mistakes are difficult to correct [18]. Physical constraints can also cause difficulties, for example, in one study, students did not realise that the speed at which their program ran would cause their LED light patterns to change too quickly to be noticed by the human eye [34]. The Modkit environment is also supportive in this regard – it provides a hardware view, which displays the circuitry and properties of components such as LEDs [43].

However, excessive support may compromise the learning-by-doing philosophy and relegate the physical creation to be a side-effect of a chiefly-computing task rather than a central component.

Figure 2: Art created by the Processing community [3]

Several researchers recommend emphasising the creative, aesthetic, and practical aspects of Lilypad workshops, as this encourages students to take ownership of their work and be more involved, challenging themselves to do better [28][42][29][31]. Furthermore, transparency in technology learning is key to deep understanding. The difficulties with correcting mistakes may encourage greater forethought and planning, which is a beneficial skill for programming, as is working to understand the technical details of the technology at hand.

A scaffolding approach [48] that gradually exposes or reduces complexity depending on students’ progress may be a good middle ground, allowing for a “low floor” (ease of entry) as well as a “high ceiling” (depth of capacities for what one can create) [42][29]. Alice has a typical example of a scaffolding approach – it allows students to write block-based programs, and then convert them into code, and eventually write code from scratch.

One of the most significant findings with the Lilypad Arduino is that workshops consistently expand students’ perceptions of computer science. Students find working on e-textiles to make computing more accessible and closer to ‘real’ work done by computer scientists than, for example, working in Alice [50]. They also consider it more relevant to their own identities and lives, and see programming as a creative activity [29]. This speaks to several of the factors affecting student engagement and retention, as discussed in section 2.1.

3.3.2 Processing. The Processing Project [23] was created with the aim to facilitate an exploratory and design-oriented approach to teaching programming. The Processing programming language is based in Java and provides a flexible set of code elements to create visual designs in both two and three dimensions. Figure 2 shows a sample of the range of art possible working in Processing. It provides simplified syntax for beginners but also supports full Java integration. This versatility makes it very amenable to educational projects.

Creative Coding with Processing presents a comprehensive creative-computing-based curriculum for first year computer science courses that, unlike most research in the field, has been applied at several institutions in the long term. It thus provides useful perspective on the sustainability and longitudinal effectiveness of creative computation approaches. The curriculum has been used and improved at two universities over six years, as well as high schools and in smaller workshops. Several other institutions adopted it later on [55][57]. The courses teach the conventional foundational CS1 topics, such as control structures, functions, and data structures, but also integrates creative coding concepts such as drawing primitives, transformations, and design. Students build a visual portfolio through assignments across the semester [26].

Xu et al. [57] report great success in improving student interest and retention through this approach. Educators comment that it created, "by far the most engaged group of students we have worked with" in five years. The courses' pre-registrations often greatly exceed capacity, and they boast close to 100% retention rates [56]. They find similar results adapting the curriculum for high school students, with increased enrolment, especially from women. [55]. However, these are reported only as observed improvements, as the sample sizes are too small to draw quantitative inferences.

In terms of student experience, many students who were not CS majors, and did not intend to do future CS courses, nevertheless indicated interest in creating their own programming projects after the course. Students said that they spent extra time on homework assignments because they found them "cool" [26]. High school students also became more vocal and confident about their work in the course [55].

Codeable Objects. Expanding these encouraging results into the physical realm, Jacobs and Buechley [27] created a Processing-based library for digital fabrication, Codeable Objects. This was inspired by the lack of novice-oriented computational design tools for digital fabrication. The Codeable Objects library plugs into the Processing environment and provides a means to create and export designs for 2D digital fabrication machines, such as laser cutters. The 2-dimensional artefacts can then be composed to form 3-dimensional objects, as was done by cutting out individual surfaces of a lampshade design and then attaching them together.

After workshops to design lampshades and garments respectively, students were found to have enjoyed the process, be pleased by their creations and more comfortable with programming than before. Almost all participants indicated that they to program again in the future, and many said that they are specifically interested in 'making' and the creation of things that are relevant and useful in their lives.

Many students used the opportunity to create designs that were personally expressive of their identity and values, such as the national flag. Meanwhile, students who entered the workshops feeling that programming was only for "really smart" people and that "people who do fashion aren't really smart" increased their confidence in their own abilities and thereby experienced changes in their self-concept. They also gained greater understanding of programming in a larger context. This highlights the important role that individual identity has to play in learning.

Unlike with the Lilypad Arduino that requires stitching and has one central controller, rapid prototyping with a laser cutter is simple as a new object can simply be cut. One student tested out one part of their project repeatedly before integrating it with other parts, analogous to test-driven development. Students appreciated this close interlinking of the programming and fabrication aspects, as well as being able to iterate their designs. However, the workshop only had one laser cutter for a large group of students, which made the process still slow and frustrating for many. The researchers also recommend that the transition workflow between digital creation and fabrication at the machines be as smooth and consistent as possible.

4 3D MODELLING AND PRINTING

Our research project aims to investigate specifically 3D printing (3DP) as a tool for student engagement in programming. 3DP is an additive manufacturing process for creating physical artefacts from digital models through depositing materials layer by layer and fusing them together. It has experienced a significant rise in popularity in recent years, driven mainly by increased availability of printers and modelling software [54].

4.1 Usage in Education

There has not been much research done into applying 3DP to CS education, but it has been widely applied to other areas of STEM, most prominently in engineering and design as a rapid-prototyping tool. Another common application is to use 3D printers to make models for mathematics or science, such as geometric forms and chemical structures, for use as educational aids [22]. These applications find that 3D objects offer greater insight than graphical modelling alone [14]. We propose that similar effects would occur in a CS education context, offering improvements over digital models such as those created in Processing, Logo and similar tools for programmatic design. One case study is presented in section 4.3.

There are several additional challenges to consider in using 3DP for teaching, including difficulty due to unfamiliarity with 3D modelling, materials performing unexpectedly, and the time and costs of printing [22]. These are very similar to the challenges faced in digital fabrication more generally, as discussed in section 3.

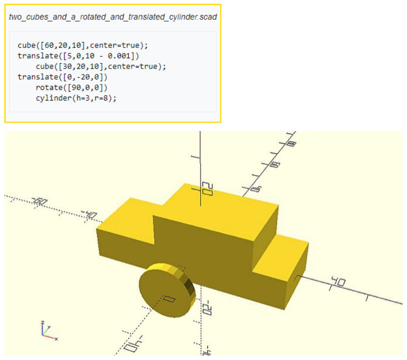
4.2 Process and Tools

The typical 3DP workflow is divided into the following four stages by Nandi et al. [38].

4.2.1 Design. A 3D model is first designed, usually making use of some Computer Aided Design (CAD) software. There is a large variety of software available for this purpose, both programmatic, such as OpenSCAD [32], and graphical. These tools function to render designs onto the screen as well as into formats usable for later stages.

For CS education, our emphasis is on programmatic tools, specifically tools for parametric design – design through algorithmically defining relations between structures. OpenSCAD is a free, text-based parametric solid CAD modelling tool. It renders C-style code describing the properties of the object to be modelled. Similar tools exist in block-based programming form, including BlocksCAD [9], Beetle Blocks [46] and TinkerCAD [10]. The latter two are free software.

Parametric designs usually consist of *shapes*, such as cubes and spheres; *transformations* such as rotation and translation; *Constructive Solid Geometry* operations for combining shapes in various



two_cubes_and_a_rotated_and_translated_cylinder.scad

```
cube([60,20,10],center=true);
translate([5,0,10 - 0.001])
    cube([30,20,10],center=true);
translate([0,-20,0])
    rotate([90,0,0])
    cylinder(h=3,r=8);
```

Figure 3: Part of an OpenSCAD tutorial for designing a car [32]

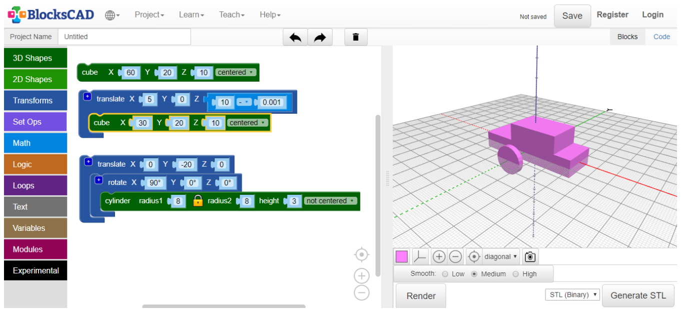
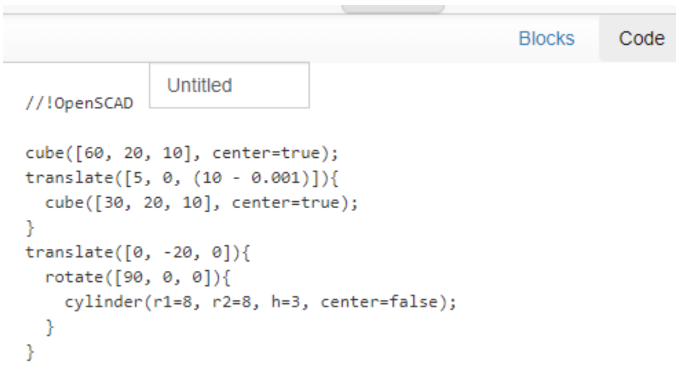


Figure 4: Car model created in BlocksCAD environment in "blocks" mode [9]



```
//!OpenSCAD

cube([60, 20, 10], center=true);
translate([5, 0, (10 - 0.001)]){
  cube([30, 20, 10], center=true);
}
translate([0, -20, 0]){
  rotate([90, 0, 0]){
    cylinder(r1=8, r2=8, h=3, center=false);
  }
}
```

Figure 5: OpenSCAD code generated by the car in BlocksCAD [9]

ways, such as union, difference and intersection; as well as conventional programming aspects such as loops, modules and conditionals [20].

4.2.2 Compiling, Printing and Iteration. The CAD-generated model is then compiled into an intermediate format such as STereoLithography (STL), which is natively supported as an export option in most CAD software. This file is fed into a slicing program that divides the 3D surface into 2D slices representing each layer to be printed. Several tools are available for slicing. Once the object is printed, the final step is to test it, and iterate this process until the object is as desired.

4.3 Case Study

We were only able to identify two studies on the applications of 3D printing to computing: (a) Roscoe et al. [45], who investigated teaching computational thinking via 3DP models created in the game Minecraft [11]; and (b) Chytas et al. [20], who investigate 3DP parametric design for the teaching of introductory programming. We select (b) for this case study because it is more directly applicable, but also because it is a detailed and cohesive study whereas Roscoe et al. fail to report complete results.

Chytas et al [20] make use of the scaffolding approach recommended in section 3.3.1 – they begin by introducing students to parametric modelling through visual, block-based programming in BlocksCAD, progressing to OpenSCAD code generated by BlocksCAD, and then eventually writing code in OpenSCAD themselves, a smooth transition as can be seen in figures 5 and 3.

They found that students were motivated and engaged by the opportunity to manufacture their designs, making more effort than in previous workshops consisting only of digital design. Most students were able to create what they had intended, either exactly or with minor alterations. However, they caution that students should be asked what they intend to design before beginning, as 'good' projects can be achieved 'accidentally'. This is especially significant since many of the participants did not manage to include sufficient programming concepts such as loops, conditions or modules in their projects, which could result in students creating random designs without gaining any deeper understanding of programming concepts.

This study also serves to emphasise the idea of a 'low floor and high ceiling', as those students with little-to-no prior programming experience felt comfortable using the BlocksCAD environment at first and could then, over time, challenge themselves with creating more sophisticated projects. They often found the OpenSCAD environment frustrating at first, especially the syntax. On the other hand, experienced programmers preferred the OpenSCAD environment. Since BlocksCAD has the same expressive powers as OpenSCAD, both groups could be comfortably accommodated and create projects at the level of complexity that was accessible to them.

5 DISCUSSION

Student engagement is a significant matter of concern in computer science education, affecting performance, understanding, and retention. Many solutions have been proposed to improve this. Here, we reviewed digital fabrication as one possible tool.

A constructionist approach to learning bodes well with known theories of motivation and engagement. Creating and interacting with objects increases experiences of autonomy, involvement and places learning into context. This was found to apply consistently to CS education, with tools ranging from Logo turtle and Scratch in the early days of computing, to Lilypad Arduinos and 3D printing today. Students are more motivated to put effort into their work, and more interested in pursuing CS in the future. They also feel more comfortable and confident with programming. Almost all studies find positive results in this regard.

However, there are some limitations to be considered. Most of these studies employ a qualitative methodology, frequently consisting of field notes, videos and observations during making sessions. While Papavlasopoulou et al. [40] describe that such methodology is best suited to evaluating students' attitudes towards the subject and experience of the session, as Weiner et al. [53] also find in their review, there is a lack of research on the efficacy of making sessions towards actual learning outcomes. We expect that improved attitudes contribute positively towards learning through improved self-efficacy and intrinsic motivation as discussed above, but this has not been empirically established for this approach in general.

It is also important to note that most research into making has been done on adolescents and youths in informal settings such as afternoon workshops. We presented Creative Coding with Processing as one long-term implementation that was similarly effective and employed in formal courses at higher institutions. Researchers advocate for wider and longer-term applications of tools found to be successful at small scales, but this has rarely been done.

Regardless, many lessons can be learnt from the research that has been done. One area that consistently present challenges to learning from making is difficulties with beginners facing complex syntax, which is a challenge for teaching programming in general. A scaffolding approach has been recommended, where students are taken from intuitive interactions such as a drag-and-drop environment, and gradually exposed to greater complexity. This ties in with the recurrent theme of having a 'low floor and high ceiling' – to ensure accessibility for beginning students but allow for high creative power and expressivity as a motivator.

Challenges more unique to making include a detachment between digital representations and physical artefacts. Fabrication machines often pose significant constraints on design and may manufacture products that students would not anticipate based on the code alone. Smoothing out this gap to make fabrication efficient and predictable as far as possible was emphasised in all three case studies. This also allows for rapid-prototyping and error-correction, something that is very important for novices, who are likely to make a lot of mistakes at first.

Tools that emphasise student individuality and creativity are also important. Students were especially motivated by being able to create objects of personal relevance and value, for example in the Codeable Objects case. Encouraging students to take ownership of their work is also a significant aspect of the making philosophy. Further, we have noted that students' identities play a major role in their motivation and involvement, as well as what they create and how they are affected by interacting with programming. As was seen in the introduction, the effects of this interaction are significant and can easily have adverse effects towards a student's attitude. It is thus imperative to consider students' identities and the transformation thereof as an integral part of the making process.

6 CONCLUSIONS

In this review, we looked at digital fabrication (DF) as a possible tool to improve student engagement in introductory programming courses. We presented theoretical motivation for considering DF as an approach by looking at constructionist theories of learning and their relationship to motivation and engagement. We then looked at a number of educational implementations of creative coding and making approaches, of which DF is one form. We considered in detail the successes and lessons to be learnt from specific implementations: the Lilypad Arduino, Processing Project, and a trial of 3D printing.

We find that digital fabrication is a promising approach, supported by a great number of successes. It has been found to consistently improve student motivation, autonomy, and engagement. However, it also comes with unique challenges to successful implementation, such as difficulties in the gap between digital design and physical creations, as well as students struggling with learning complex syntax. These need careful consideration to be addressed in future efforts. There is also a need for research into wider, longer-term applications, such as full courses, of these tools.

Limited research has been done on the applications of 3D printing, specifically, to introductory programming. Regardless, we believe that the results from other forms of digital fabrication can be extended to 3DP, suggesting that it would be a valuable subject of study, especially if we can account for the challenges identified.

REFERENCES

[1] [n.d.]. Arduino Lilypad. https://www.arduino.cc/en/Guide/ArduinoLilyPad
[2] [n.d.]. Lilypad Arduino main board. https://store.arduino.cc/lilypad-arduino-main-board
[3] [n.d.]. OpenProcessing. https://www.openprocessing.org/
[4] [n.d.]. Scratch - imagine, program, share. https://scratch.mit.edu/
[5] 2007. Tertiary Level ICT Skills Development. http://www.cs.ru.ac.za/ICTSkills/Declaration%20-ICT-Skills%202007.pdf
[6] 2015. Logo history. https://el.media.mit.edu/logo-foundation/what_is_logo/history.html
[7] 2017. Alice: tell stories. build games. learn to program. https://www.alice.org/
[8] 2017. Starry night prom. https://create.arduino.cc/projecthub/Maddy/starry-night-prom-2eb206?ref=part&ref_id=15653&offset=0
[9] 2019. BlocksCAD editors. https://www.blockscad3d.com/editor/
[10] 2019. Tinkercad | from mind to design in minutes. https://www.tinkercad.com/
[11] 2020. Minecraft Official Site. https://www.minecraft.net/en-us/
[12] 2020. Modkit. http://www.modkit.com/
[13] Steve Olusegun Bada and Steve Olusegun. 2015. Constructivism learning theory: A paradigm for teaching and learning. *Journal of Research & Method in Education* 5, 6 (2015), 66–70.
[14] James Baleshta, Peter Teertstra, and Benny Luo. 2015. Closing the loop: Integrating 3D printing with engineering design graphics for large class sizes. *Proceedings of the Canadian Engineering Education Association (CEEA)* (2015).
[15] Susan Bergin and Ronan Reilly. 2005. The influence of motivation and comfort-level on learning to program. (2005).
[16] Maureen Biggers, Anne Brauer, and Tuba Yilmaz. 2008. Student perceptions of computer science: a retention study comparing graduating seniors with cs leavers. *ACM SIGCSE Bulletin* 40, 1 (2008), 402–406.
[17] Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. 2008. The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 423–432.
[18] Leah Buechley, Mike Eisenberg, and Nwanua Elumeze. 2007. Towards a curriculum for electronic textiles in the high school classroom. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*. 28–32.
[19] Christopher P Cerasoli, Jessica M Nicklin, and Michael T Ford. 2014. Intrinsic motivation and extrinsic incentives jointly predict performance: A 40-year meta-analysis. *Psychological bulletin* 140, 4 (2014), 980.
[20] Christos Chytas, Ira Diethelm, and Alexandros Tsilingiris. 2018. Learning programming through design: An analysis of parametric design projects in digital fabrication labs and an online makerspace. In *2018 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 1978–1987.
[21] Diana I Cordova and Mark R Lepper. 1996. Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *Journal of educational psychology* 88, 4 (1996), 715.

[22] Simon Ford and Tim Minshall. 2019. Invited review article: Where and how 3D printing is used in teaching and education. *Additive Manufacturing* 25 (2019), 131–150.

[23] Ben Fry and Casey Reas. 2020. Processing.org. https://processing.org/

[24] Vashti C Galpin and Ian D Sanders. 2007. Perceptions of computer science at a South African university. *Computers & Education* 49, 4 (2007), 1330–1356.

[25] Michail N Giannakos, Ilias O Pappas, Letizia Jaccheri, and Demetrios G Sampson. 2017. Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness. *Education and Information Technologies* 22, 5 (2017), 2365–2382.

[26] Ira Greenberg, Deepak Kumar, and Dianna Xu. 2012. Creative coding and visual portfolios for CS1. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. 247–252.

[27] Jennifer Jacobs and Leah Buechley. 2013. Codeable objects: computational design and digital fabrication for novice programmers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1589–1598.

[28] Yasmin Kafai, Deborah Fields, and Kristin Searle. 2014. Electronic textiles as disruptive designs: Supporting and challenging maker activities in schools. *Harvard Educational Review* 84, 4 (2014), 532–556.

[29] Yasmin B Kafai, Eunkyoung Lee, Kristin Searle, Deborah Fields, Eliot Kaplan, and Debora Lui. 2014. A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles. *ACM Transactions on Computing Education (TOCE)* 14, 1 (2014), 1–20.

[30] Yasmin B Kafai, Kristin Searle, Eliot Kaplan, Deborah Fields, Eunkyoung Lee, and Debora Lui. 2013. Cupcake cushions, scooby doo shirts, and soft boomboxes: e-textiles in high school to promote computational concepts, practices, and perceptions. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 311–316.

[31] Eva-Sophie Katterfeldt, Nadine Dittert, and Heidi Schelhowe. 2015. Designing digital fabrication learning environments for Bildung: Implications from ten years of physical computing workshops. *International Journal of Child-Computer Interaction* 5 (2015), 3–10.

[32] Marius Kintel. [n.d.]. OpenSCAD. http://www.openscad.org/

[33] Külli Kori, Margus Pedaste, Eno Tõnisson, Tauno Palts, Heilo Altin, Ramon Rantsus, Raivo Sell, Kristina Murtazin, and Tiia Rüütmann. 2015. First-year dropout in ICT studies. In *2015 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 437–445.

[34] Winnie WY Lau, Grace Ngai, Stephen CF Chan, and Joey CY Cheung. 2009. Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students. In *Proceedings of the 40th ACM technical symposium on Computer science education*. 504–508.

[35] Louis Major, Theocharis Kyriacou, and O Pearl Brereton. 2012. Systematic literature review: teaching novices programming using robots. *IET software* 6, 6 (2012), 502–513.

[36] Lee Martin. 2015. The promise of the maker movement for education. *Journal of Pre-College Engineering Education Research (J-PEER)* 5, 1 (2015), 4.

[37] Jesús Moreno-León and Gregorio Robles. 2016. Code to learn with Scratch? A systematic literature review. In *2016 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 150–156.

[38] Chandrakana Nandi, Anat Caspi, Dan Grossman, and Zachary Tatlock. 2017. Programming language tools and techniques for 3D printing. In *2nd Summit on Advances in Programming Languages (SNAPL 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[39] Matthew W Ohland, Sheri D Sheppard, Gary Lichtenstein, Ozgur Eris, Debbie Chachra, and Richard A Layton. 2008. Persistence, engagement, and migration in engineering programs. *Journal of Engineering Education* 97, 3 (2008), 259–278.

[40] Sofia Papavlasopoulou, Michail N Giannakos, and Letizia Jaccheri. 2017. Empirical studies on the Maker Movement, a promising approach to learning: A literature review. *Entertainment Computing* 18 (2017), 57–78.

[41] Seymour Papert and Idit Harel. 1991. Situating constructionism. *Constructionism* 36, 2 (1991), 1–11.

[42] Kylie Peppler. 2013. STEAM-powered computing education: Using e-textiles to integrate the arts and STEM. *Computer* 9 (2013), 38–43.

[43] Kanjun Qiu, Leah Buechley, Edward Baafi, and Wendy Dubow. 2013. A curriculum for teaching computer science through computational textiles. In *Proceedings of the 12th international conference on interaction design and children*. 20–27.

[44] Mona Rizvi and Thorna Humphries. 2012. A Scratch-based CS0 course for at-risk computer science majors. In *2012 Frontiers in Education Conference Proceedings*. IEEE, 1–5.

[45] Jonathan Francis Roscoe, Stephen Fearn, and Emma Posey. 2014. Teaching computational thinking by playing games and building robots. In *2014 International Conference on Interactive Technologies and Games*. IEEE, 9–12.

[46] Eric Rosenbaum, Duks Koschitz, Bernat Romagosa, and Jens Monig. [n.d.]. Beetle Blocks - Visual code for 3D design. http://beetleblocks.com/

[47] Richard M Ryan and Edward L Deci. 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist* 55, 1 (2000), 68.

[48] Debra Sanders and Dorette Sugg Welk. 2005. Strategies to scaffold student learning: Applying Vygotsky's zone of proximal development. *Nurse educator* 30, 5 (2005), 203–207.

[49] Saint Paul Public Schools. 1985. *Logo: Learning in a Computer Culture.* https://el.media.mit.edu/logo-foundation/resources/archive_docs/st_paul.pdf

[50] Kristin A Searle, Deborah A Fields, Debora A Lui, and Yasmin B Kafai. 2014. Diversifying high school students' views about computing with electronic textiles. In *Proceedings of the tenth annual conference on International computing education research*. 75–82.

[51] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. 2014. A systematic review of approaches for teaching introductory programming and their influence on success. In *Proceedings of the tenth annual conference on International computing education research*. 19–26.

[52] Shirin Vossoughi and Bronwyn Bevan. 2014. Making and tinkering: A review of the literature. *National Research Council Committee on Out of School Time STEM* (2014), 1–55.

[53] Steven Weiner, Micah Lande, and SS Jordan. 2018. What Have We" Learned" from Maker Education Research? A Learning Sciences-base Review of ASEE Literature on the Maker Movement. *Review & directory-American Society for Engineering Education* (2018).

[54] Erin Winick. 2017. 3D Printing's 30 Year History and Why It's Popular Now. https://medium.com/@erinwinick/3d-printings-30-year-history-and-why-it-is-popular-now-5200ab21a7ed

[55] Dianna Xu, Aaron Cadle, Darby Thompson, Ursula Wolz, Ira Greenberg, and Deepak Kumar. 2016. Creative computation in high school. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 273–278.

[56] Dianna Xu, Ira Greenberg, Deepak Kumar, and Ursula Wolz. 2016. Creative Computation for CS1 and K9-12. (2016).

[57] Dianna Xu, Ursula Wolz, Deepak Kumar, and Ira Greenburg. 2018. Updating Introductory Computer Science with Creative Computation. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 167–172.

[58] LeChen Zhang and Jalal Nouri. 2019. A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education* 141 (2019), 103607.