UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2020

Title: 3D Printing as a Tool for Increasing Student Motivation in Introductory Programming Courses

Author: Cameron Adams

Project Abbreviation: code3d

Supervisor(s): Gary Stewart, Aslam Salfa (second reader)

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	10
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	15
System Development and Implementation	*0*	*20*	10
Results, Findings and Conclusions	*10*	*20*	15
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

3D Printing as a Tool for Increasing Student Motivation in Introductory Programming Courses

Cameron Adams
Department of Computer Science
University of Cape Town
Cape Town, South Africa
admcam003@myuct.ac.za

Abstract

Enrollment of introductory programming courses are increasing as programming skills are becoming more prevalent and sought after in various fields. Studies have shown that students face difficulties in comprehending and applying programming concepts when writing programs to solve problems. New ways of increasing motivation and engagement in introductory programming courses are being tested. This paper presents a web application that students can code 3D artefacts and then simulate the 3D printing of these models. Educators at the University of Cape Town were surveyed to answer whether or not this proposed tool is considered a viable way to motivated new programmers and whether or not this system is usable. The results of this study found that the educators find the combination of programming and 3D printing simulation has potential to be introduced the classroom and curriculum.

CSS Concepts
Computer Science education - Visual tools; Visualization - 3D printing simulation

Keywords
Graphics libraries, visualization, 3D printing, web development, novice programming

1 Introduction

The world is becoming immensely more digital and the computer has massively contributed to the growth of civilization. A global acknowledgement of the importance of Computer Science education is growing [9]. Previously unrelated professions, such as journalism and law, are starting to need technical programming skills [1].

Computational thinking (CT) is becoming more widespread. Many people, who have no particular interest in Computer Science, but who engage in social media, typically discuss the results of the underlying recommendation systems. Computer Science enrollment at universities is growing fast and the challenge still remains on how to capitalise on the increased interest and to expand the enrollment among women and minorities [12].

Learning to program is not easy. An article by Gomes *et al.* [6] neatly summarised the difficulties and problems into a) teaching dynamic concepts that use static materials, b) learning programming is practical and some students tend to just read textbooks, c) students need adequate mathematical and logical thinking, d) complex syntax and a demand for high level abstract thinking can be challenging, e) there is an image of what a typical programmer is are some people are put off.

1.1 Proposed Solution and Motivation

Graphics libraries such as p5.js, Processing and Logo Turtle were created to motivate new programmers, which made drawing and creating program visualizations easy and efficient. Extending this principle, an initial tool was proposed, that allowed students to create and print 3D artefacts effortlessly. By bridging programming and 3D printing, students have the possibility to explore emerging technologies like 3D printing, which is a very promising and exciting technology for both industry and education.

However, due to the COVID-19 pandemic, social distancing laws were enforced and research was conducted remotely. This means surveying students and using a 3D printer was substituted with surveying educators and using a 3D printing simulation.

It was argued that this is still has potential to be a viable way to get students excited about programming, as it gives them the thought that their skills and knowledge *can* produce tangible products, as well as introduce them to the innovative world of 3D printing.

This tool was designed for students who are not complete beginner programmers, and have some understanding of programming fundamentals.

1.2 Work Allocation

There were two systems to this tool: Model3D and Print3D. Model3D deals with users being able to code a 3D artefact and visualise it, and was worked on by Maria Nanabhai. Print3D deals with users being able to interact with a simulation of the user-coded-model, and was worked on by Cameron Adams. Both of these parts were critical to the running and testing of the tool. In the text to follow, 'the system' will be referring to Print3D most of the time.

1.3 Research questions

1. Do educators consider a 3D printer simulator a useful tool to increase student motivation in learning introductory programming?
2. Is it possible to create a usable 3D printing simulator?

2 Background

2.1 Computer Science Education

Literature in the teaching of introductory programming can be categorized into four sub-fields: curricula, pedagogy, language choice and tools for teaching [7].

Curricula: Countries generally have their own highly influential groups to discuss and plan new curricula. In the US, they have major professional computing societies (the ACM and IEEE Computer Society) and in Europe they have the Bologna Accord. Every introductory computing curricula has a programming metaphor and paradigm. Metaphors like "computation as calculation" or "computation as interaction". Paradigm examples are object oriented or functional programming.

Pedagogy: While the curriculum defines the material to be taught, pedagogy deals with the manner in which the material is taught, learnt and managed in order to get desired learning outcomes.

Language choices: Many languages have been used for teaching introductory programming. Factors such as faculty preference, industry relevance, complexity and syntax influence the choice.

Tools for teaching: Tool research papers were divided into three sub-fields. Visualization tools, automated assessment tools and programming environments.

Automated assessment tools: These are needed for the high enrollment. They help teachers check the correctness of a program's execution and provide a way for students to learn from errors and improve work before final submission. A limit to the number of submissions can mitigate the case where students might exploit a trial-and-error approach to programming.

Programming environments: Integrated Development Environment (IDE) increases productivity by organising program components, highlighting syntax, and visualizing debuggers.

Visualization tools are particularly relevant to this paper, these tools and visual learning will be discussed in the following subsection.

2.2 Visualization in CSE

Visual learning has lagged behind relative to verbal learning [3]. Visual representations attract and maintain motivation. Visualization provides an additional way of representing information and encourages the learning that students may not get from text alone. Visual literacy is a crucial component of technological literacy and its development is one of the main goals of STEM education [17].

Program visualization (PV) has been defined by many authors but the general idea is: the use of various techniques to enhance the human understanding of computer programs. Examples are debuggers and IDEs. These examples visualize *implemented* code. With this definition, physical computing, which is the use of microcontrollers to control electronics, such as LEDs and switches, could be considered as program visualizations [16].

Visual programming (VP) is the use of "visual" techniques to specify a program. Examples are Scratch and Alice which are environments that do not necessarily use code but instead use drag-and-drop or other visual abstractions to interface with 'read' code to then implement a sequence of computational steps.

Algorithm visualization (AV) is the visualization of a high-level description of a piece of software. AVs are a subset of PVs, as a program may include multiple different algorithms.

Software visualization (SV) can be defined to include all of these as a means to reduce ambiguity and covers all of the software design process from planning to implementation [15].

These SV tools aid lectures to teach and students to learn though there are different levels of engaging with a SV, that have been shown are more effective than others.

A 2002 study by Hundhausen *et al.* [13] argued that no matter how well crafted a visualization is, it is of little value unless students are in an active learning activity. The study's purpose was to develop a taxonomy to provide a framework for conducting empirical experiments that attempt to evaluate the effectiveness of AV in teaching. The six categories in the taxonomy are: 1) No viewing, 2) Viewing, 3) Responding, 4) Changing, 5) Constructing, 6) Presenting

Using this taxonomy, Grisson *et al.* [8] in 2004 investigated the influence of visualizing *sorting* algorithms in lectures on students' learning from 3 different universities across USA. The study compared 3 different ways students could engage the sorting algorithm visualization. Engagement levels were (1) not showing, (2) showing and (3) allowing interaction. Results showed that students that were allowed higher levels of engagement in the visualization scored higher marks.

2.3 3D printing in CSE

Digital fabrication is the transformation of digital designs into physical products by manufacturing machines (3D printers, laser cutters, CNC routers etc.) which are controlled by computers [3] and can enable the manufacturing of complex shapes and geometries.

Educators are becoming more aware of 3D printing for its potential in STEAM education. Even though the potential of digital fabrication in STEAM education is clear, its potential to computing education remains unclear [2]. Digital fabrication techniques such as 3D printing offer an opportunity for students to write programs that produce tactile objects, providing an accessible way of exploring program output [11].

A 2016 study [14] looked at teaching programming fundamentals through 3D printed objects at I.S.I. Fermi, a high school set in Lucca, Italy. 3D printed cylinders with different heights were asked to be sorted, by height, using popular sorting algorithms such as bubble sort. The study found that teaching data structures and sorting algorithms through physical 3D printed objects was successful.

A 2014 study by Kane *et al.* [11] explores integrating data analysis and 3D printing into programming instruction for blind students at a four-day workshop in Maryland, USA. Blind and visually impaired students wrote Ruby programs to analyze data from Twitter regarding a fictional ecological crisis.It can be argued that if 3D printing motivated blind students to learn programming, the same can be said for sighted students. After the study students were motivated to learn about 3D printing technologies, and to use their programming skills to create 3D-printed artifacts.

In 2018, Chytas *et al.* [2] used a very similar approach to motivate new programmers but had a strong emphasis on programming style and techniques. They also used a programming language that users can program a 3D artefact and allowed students to 3D print their models. Since the paper was on *teaching* and *learning* programming, the effect of allowing students to 3D print their designs was not the main objective and was not surveyed. Only observations were made. Though the observations were positive in the sense that students were motivated and engaged by the opportunity to manufacture their designs, making more effort than in previous workshops consisting *only* of programming 3D models.

3 System Development and Implementation

3.1 Requirements

The system is a web application that has two main features: The first is a text editor, with a built in graphics library where uses can code and render a 3D model. This 'sub system' is an independent tool of it's own and it is called Model3D. The second feature is a 3D printer simulator that users can interact with and view the virtual printing of their own artefact. This is also a 'sub system' that can be independent on its own and it is called Print3D.

So to summarise the core properties of the system:

- A text editor to code a 3D model.
- A render window to view the output of the code.
- A window to simulate the print of a model.
- Controls to interactive with simulation.

3.2 Design

Model3D

User implemented code, specifying a 3D model, is inputted and Model3D outputs the rendering of the model. The model is coded in Javascript via a web text editor Ace. The code uses a 3D graphics library, OpenJSCAD, to create and operate on solid 3D CAD (Computer Aided Design) objects.

This library was chosen since its syntax is simple and efficient to make 3D artefacts and, hence, is suited for beginner programmers. After the model developed, it is rendered in the front end via OpenJSCAD.

Note that this tool was not designed to teach introductory programming, but is a motivating tool to get users excited to create and code.

Print3D

A 3D model is inputted and Print3D outputs a simulated print of this model. The 3D model is sent in the form of a STL (Standard Tessellation Language) file via the the front-end/client to the back-end/server. A STL file is native to a CAD software. Though, a 3D printer needs an algorithm to move it's tip and extrude filament around a 3D space. Gcode is the language that specifies these movements. An STL file can be 'sliced'/converted into a Gcode file via slicing software. This is what the server does, it uses a well known slicing software, Slic3r, and slices the user-coded-model (STL file) with a default printing configuration. The Gcode is then sent to the client and is visualized with Three.js, a JavaScript library used to create animated 3D computer graphics in

a web browser. Since a Gcode file is essentially a list of 3D points that a 3D printer follows, a program was developed that controls the drawing of lines between these points. This simulates the print of a 3D model.

Note that due to time constraints, users cannot specify 3D printing parameters and their models are sliced with default setting. This feature is hypothesised to have potential to increase users motivation to learn programming and is a great future addition. Though users argued that the system was sufficient to answer the research questions. Users also argued this also may take students away from the objective of programming and rather to teaching 3D printing.

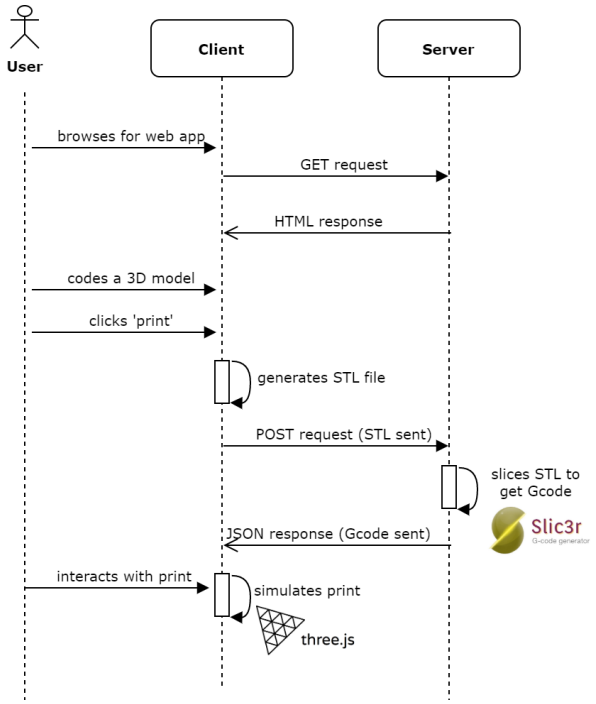

Figure 1: Sequence diagram of Print3D

Figure 1, shown above, is a sequence diagram that highlights the steps Print3D does to simulate a user-code-model.

3.3 Architecture

Figure 2 is an architectural diagram, shown below, highlighting the core frameworks used to develop Print3D. It shows a client-server model, which is typical of a web application. The front-end/user interface basic structure was developed using HTML, CSS and Javascript. Model3D fully operates on the client side while Print3D operates in both the front-end and back-end. The back-end was developed in Node, a JavaScript runtime environment that executes JavaScript code outside a web browser. These are very popular libraries used to create web applications.

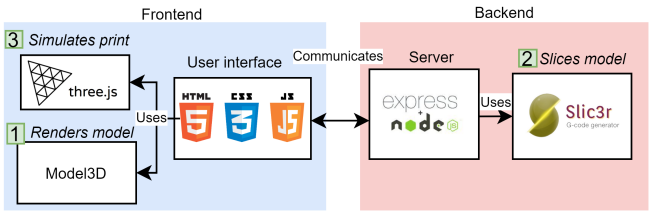

Figure 2: System architectural diagram, highlighting Print3D's core frameworks and main functionality

3.4 Implementation

The project APIs, Model3D and Print3D, encapsulate open source graphics libraries and 3D printing software mentioned in the Design section 3.2. They also add extra functionality to make it as simple as possible for users to create and visualise 3D artefacts and prints.

It was planned to develop the two systems independently and integrate them later. The prototype independent systems with the core requirement were demonstrated to the project supervisors and feedback and critiques were returned for improvements. There after both developers got together to plan and integrate the two systems into one combined system.

The two independent systems were web applications of their own and thus has a client/server model. Integrating them together was to simply combine the each systems' client and server implementations. The back-end of Model3D was straight forward, only waiting for HTTP GET requests and then serving the front-end, a single HTML file. The back-end of Print3D is waits for POST requests where STL files are posted, sliced into Gcode and sent back to the client. The integrated back-end had these two functionalities.

For the front-end integration, Print3D depended on Model3D's functionality to generate a STL file from a user-coded-model. When a simulation of print was invoked, Model3D generated an STL file to give to Print3D which posted the file to the server for slicing. Figure 1 highlights this process. This implementation was used for user testing.

Two user testing sessions were held and based on the feedback received from the first round, relatively minor improvements and additions were made. Main critiques were the two visualizations to be on separate pages, to improve flow and usability. A 'card' based UI was implemented such that the two visualization can be viewed independently in an simple and intuitive way.

See Appendix C and D for a overview of the user interface for Model3D and Print3D

3.5 Software Engineering Methodology

An agile methodology was decided on the implementation process. This promoted regular incremental builds of software as well as frequent communication. By using Git, a software version control tool, both systems, Print3D and Model3D, were kept in check.

4 Research Methodology

An investigation was set out to gain an understanding on how the combination of 3D printing simulation and programming using graphics libraries can contribute to computing education and more specifically motivating the learning of programming.

Due to the constraints imposed by the pandemic, it was chosen to use expert evaluation for assessing the system. On account of the small available sample size, a mixed methods approach was employed, combining both quantitative and qualitative methods. This included surveys regarding system usability and motivation, as well as interviews with participants to discuss in greater depth their experience using the system. The usage of multiple measurement approaches means that results are more likely to be due to underlying phenomena than the particular method used, reducing bias and error and improving validity [10].

4.1 Materials/measures

Usability

The PSSUQ (Post-Study System Usability Questionnaire) is a 17-item standardized questionnaire. It is widely used to measure a user's perceived satisfaction of a website, software, system or product at the end of a study. The questionnaire allowed users to rate the 17 statements using a 7-point Likert scale (1=strongly disagree up to 7=strongly agree).

Appendix A shows 17 items of the usability questionnaire. The questionnaire provides four subscores:

- Overall: the average scores of questions 1 to 17
- System Usefulness: the average scores of questions 1 to 8
- Information Quality: the average scores of questions 9 to 15
- Interface Quality: the average scores of questions 16 to 17

The sub-scales provide a more detailed breakdown of different factors affecting the system.

Motivation

Instructor attitudes towards the system as a tool for teaching will be assessed using the M-SERI (Modified Survey About Educational Robotics for Instruction), an 18-item questionnaire with 5-point Likert scale (1=strongly disagree, 2=disagree, 3=not sure, 4=agree, 5=strongly agree). The original survey was modified slightly by interchanging "educational robots" with "3D printing" and altered some questions to concentrate on utility towards motivation.

Appendix B shows 18 items of the motivation questionnaire. The questionnaire provides four subscores:

- Overall: the average scores of questions 1 to 18
- Utility of the system (UTIL): the average scores of questions 1 to 5
- Ability to use the system (ABIL): the average scores of questions 6 to 11
- Intent to use the system in the future (INT): the average scores of questions 12 to 18

As with the usability questionnaire, the sub-scales provide a more detailed breakdown of different factors affecting the system.

The validity of this survey is ensured by two factors; reverse coding and midpoint response.

Reserve coding means some statements, if agreed to, results in a negative response to the statement. For example, "I don't know how to use this software" is reverse coding *i.e* if users rated 4 (agree) or higher, it's a negative response towards our system. Therefore, the reverse coded scores need to be 'reversed' to get the results needed to answer the research questions. The 'reversal' is a mapping as follows: 1 and 5 swop, 2 and 4 swop.

Reverse coding protects against users who are either unable to understand the questions (e.g. possibly due to a language barrier or learning disability), or due to low interest may choose not to read the questions, and simply respond to all items with "strongly agree" [4].

A *midpoint* Likert scale option provides respondents a choice when they truly do not have an opinion, rather than forcing them to create artificial opinions.

4.2 Participants

Four staff members from the Computer Science department at the University of Cape Town, participated in our study, recruited using the purposive sampling method through the projects supervisor. This means that participants are selected on the basis of the experience and knowledge they possess, which is an efficient method for collecting expert data [5]. These 4 educators and teaching assistants had experience teaching introductory programming courses.

4.3 Procedure

The purpose of the procedure is to organize the experiment. Two rounds of user testing were conducted, an initial round and a final round. User testing interviews were conducted through a 30 minute video call.

Initial Round
Users were introduced to the project's aim and research questions, and to the system. An example task, using the system, was done to further explain the 3D modeling library and printing controls. All of this was done in the first 10 minutes. With this knowledge, the participant were shown two pre-made artefacts and they attempted to develop algorithms and use built-in functions, and then print these examples. The algorithms included for loops, arrays, variables and functions.

Enough information was left out in our introduction such that the users have to think for themselves and use the system independently.

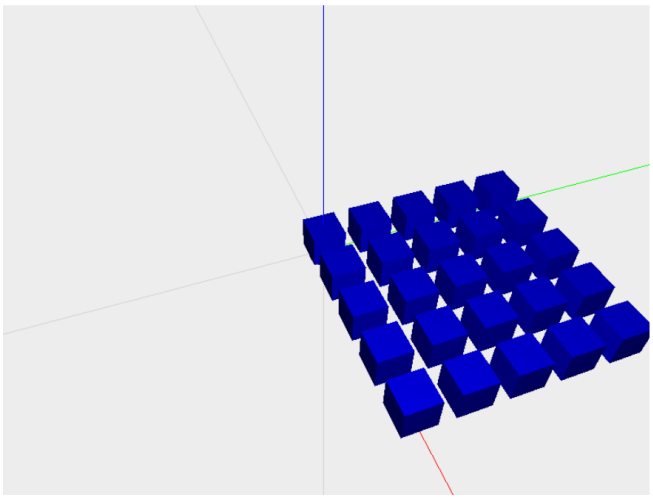

Figure 3: User task 1, a grid of cubes, for the initial user test

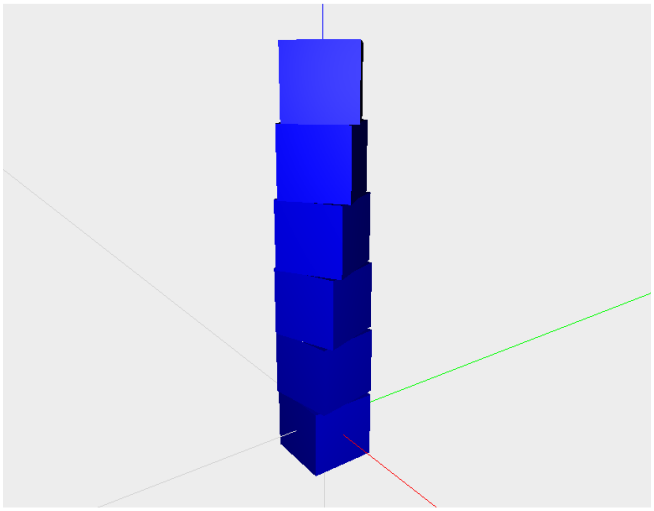

Figure 4: User task 2, a stack of rotating cubes, for the initial user test

Figure 3 and figure 4 were the two artefacts they needed to model and print for the initial user test. User task 1 requires a nested for loop, the cube object and the translate function to position the cubes. User task 2 also required a for loop and translate function, as well as a rotateZ function.

At the end of the session, users were asked complete two surveys regarding motivation and usability of the system, and to provide any critiques for the system to be improved. Suggestions for improvement were also collected throughout the call. With this feedback, adjustments were made to the system.

Final Round
Users were introduced to an improved system and, as before, ask to complete a user task.

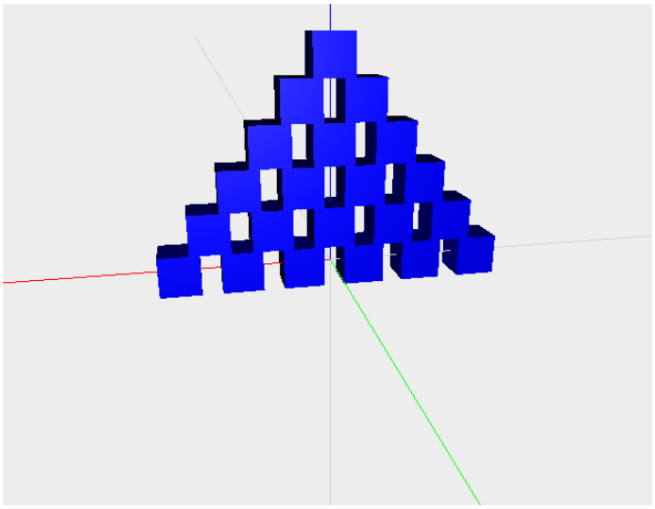

Figure 5: User task 3, a tower of cubes, for the final user test.

Figure 5 was the user task for the final round. This time scaffolded code was provided as this is a way to show users that you can create custom shapes and not just primitive standard shapes provided by OpenJSCAD.

At the end of the session, an interview was conducted, asking about the users background in Computer Science, 3D modeling and 3D printing. Users, again, completed the two surveys. With this feedback, the potential of creative coding and 3D printing in computing education, is revealed, and more precisely, in introductory programming.

5 Results

Section 4 discussed how the testing was done, this section presents what the testing revealed.

5.1 Initial user testing

The aim of the initial round was to introduce users to the project and the system. Users got to try the system

a code a few tasks and they provided comments and critiques to improve the usability of the system.

Usability

The educators level agreement to the usability of the simulator was measured using the 7-point Likert scale (1=strongly disagree up to 7=strongly agree). A descriptive analysis of the reading are divided into three levels, namely low (1.00 – 2.99), medium (3.00 – 4.99), and high (5.00 – 7.00)

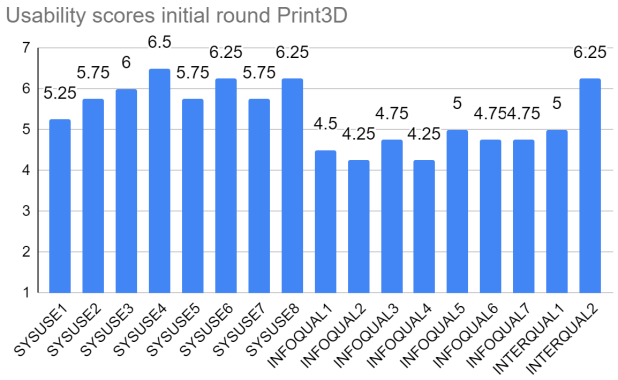

Figure 6: Results from initial usability questionnaire

Figure 6 shows the average initial responses to the usability survey in a bar chart form. Based on the mean score, the respondents give a high response for all items except for most (6 out 7) items regarding information quality, which had a medium response.

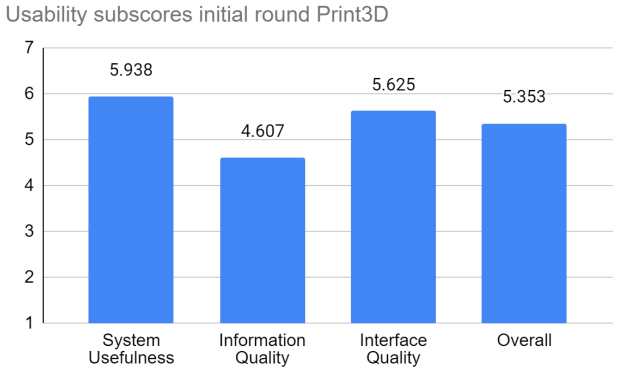

Figure 7: Results from initial usability questionnaire in subscores

Figure 7 shows the three subscores and overall score of the initial responses to the usability survey in a bar chart form.

The highest subscore was system usefulness (5.938) which indicates educators find the system simple and efficient. The lowest subscore was information quality (4.607) which means there was room for improvement in terms of information being shown on screen. A suggestion for a user was to have the controls react to the simulation, for example, have the play button's text toggle between play and stop if the print is idle or printing respectively.

Motivation

The educators level agreement to the simulator being a motivational tool for students to learn programming fundamentals was measured using the 5-point Likert scale (1=strongly disagree, 2=disagree, 3=not sure, 4=agree, 5=strongly disagree). A descriptive analysis of the reading are divided into three levels, namely low (1.00 – 2.33), medium (2.34 – 3.66), and high (3.67 – 5.00).

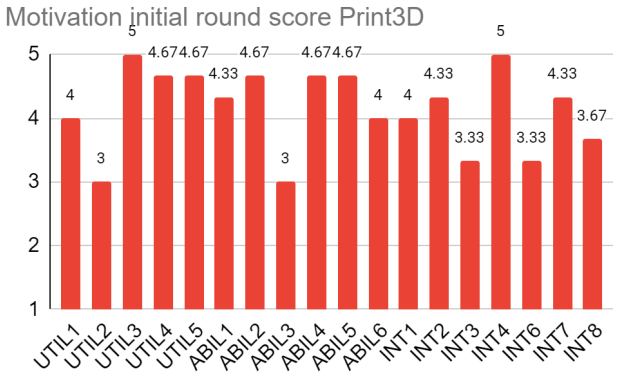

Figure 8: Scores from motivational survey inital round

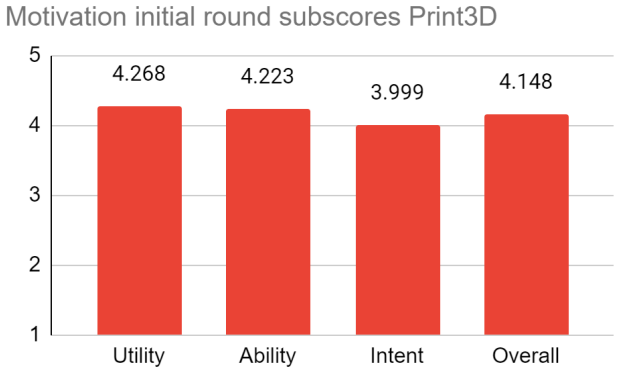

Figure 9: Subscores from motivation survey initial round

Figure 8 shows the average initial responses to the motivation survey in a bar chart form and figure 9 shows the three subscores and overall score of the initial responses to the motivation survey in a bar chart form.

Overall, the response was high (4.148) and so was each subscore, utility of the system, the ability to use the system and intent to use the system. Though each category

had at least one item with a medium response. This randomness for each category could be due to the system being so new to the participants.

The highest items were UTIL3, "I think working with this software in class would be enjoyable, and stimulating" and INT3, "I think this software is not interesting", both a score of 5 (note that INT3 is reverse coding, meaning the level of agreement to the statement is reversed *i.e.*if a participant rated with a 1, the score is 5, and if rated with a 2 the score is 4) This indicates that the system is exciting and interesting

The lowest items were UTIL2, "Children should be introduced to this software in school", ABIL3 ,"I am unsure of my ability to integrate this software into my classes", both with a score of 3. This indicates this tool may be too dawning for younger learners and that the system may need further development in order to bring it to the classroom.

All 4 participants were interested and engaged when trying the system for the first time. This is some positive feedback from a user - "Overall, I really enjoyed using the system and felt motivated to experiment with new ideas for 3D Prints. The immediate feedback loop is something I really appreciated and I feel like this really fits in with the quick iteration philosophy of 3D printing."

The main critique was the viewing of both visuals - the model and the print. Initially, they were both on the same page and a user suggested trying a 'card' based UI to switch between the two visuals easily and effectively.

5.2 Final user testing

Users were introduced to an updated user interface and tried to code another task.

Usability

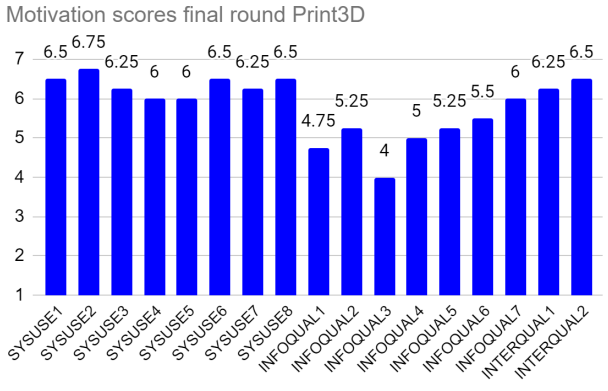

Figure 10: Results from final usability questionnaire

Figure 10 shows the average final responses to the usability survey in a bar chart form. Initially there were 6 out 7 information quality items that had a medium, but after the final round, 4 of them moved from medium to high, leaving only 2 with a medium. This improved is most likely due to the two systems being separated in the UI and information being more segmented.

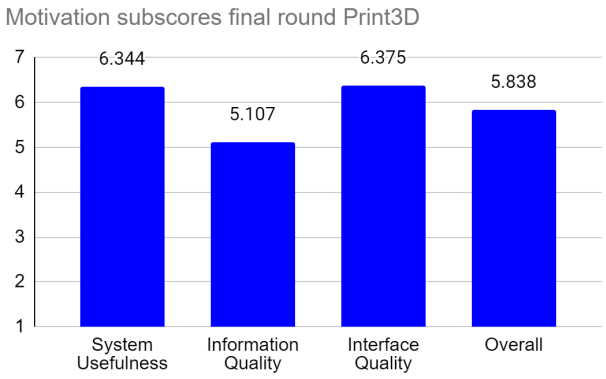

Figure 11: Results from final usability questionnaire in subscores

Figure 11 shows the three subscores and overall score of the final responses to the usability survey in a bar chart form.

Each subscore had a high response and had improved. At the end, the mean average score for the overall usability study is 5.838 (max of 7), which shows that it is possible to create a usable 3D printing simulator.

Motivation

Since this is the second time the participants try the system, it was expected that the results improve, since they were more familiar with the general idea of system, and since minor adjustments to user interface and functionally were made from initial round feedback.

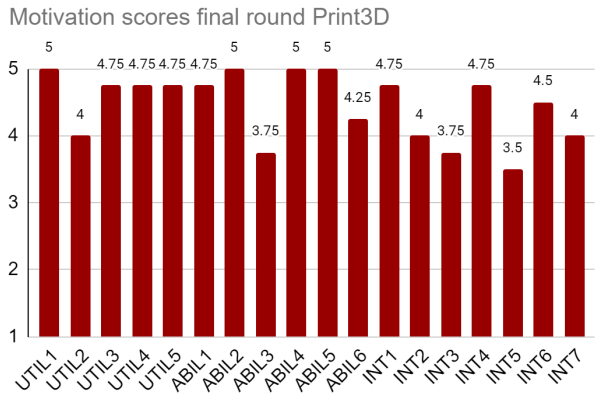

Figure 12: Results from final usability questionnaire in subscores

Figure 12 shows the average final responses to the motivation survey in a bar chart form.

17 out of the 18 items had a high response which indicates this has high potential to motivate introductory programming. ABIL2, "I am confident that I could learn to use this software for teaching," scored a 5 from all participants. This indicates the potential to introduced this tool into introductory programming circular and classes.

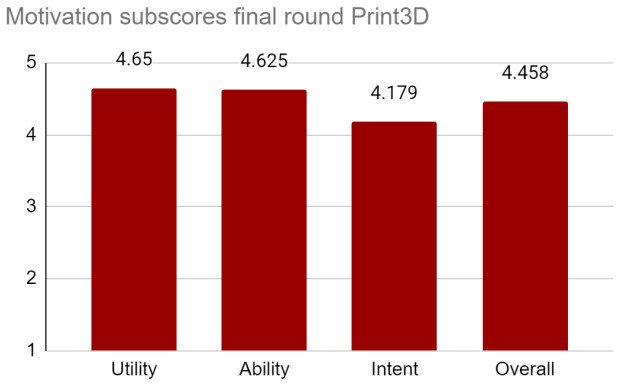

Figure 13: Results from final usability questionnaire in subscores

Figure 13 shows the three subscores and overall score of the final responses to the motivation survey in a bar chart form.

At the end, the mean average score for the overall motivation study is 4.4 (max of 5), which shows that educators do consider a 3D printer simulator a useful tool to increase student motivation in learning introductory programming

A collection of the interviews shows that participants most favourite aspect of the system is the immediate feedback/interactivity of the system. The most least favourite was the lack of documentation/guide to using the system. This indicates the system could be more 'user friendly' with pop up messages to direct and guide a user. If this tool user were to be introduced into an introductory curriculum, learners would be assigned a few days to learning and understanding the language and system.

6 Discussion

Our study has revealed that educators consider the combination of 3D printing simulation and programming 3D models to motivate new programmers. The educators were showed that these technologies they can provide possibilities for students to explore programming and 3D modeling.

Previous work showed that the higher the level of engagement to a visualisation, the higher the learning. The highest levels of engaging in a visualization are constructing and presenting. Users construct 3D models and interact with and present the 3D print simulation.

Generally, the educators seemed to enjoy the idea of creating something and potentially 3D printing it. A pleasantly surprising thing that was noticed, was when a user tried to complete the user tasks, sometimes they made a mistake in the code, and a different yet satisfying model was outputted. This would surprise and inspire them, at what they could produce using the tool. In a standard intro-to-programming course assignment, this logic error would likely be corrected and forgotten, but using this system users can get excited about programming is many more ways.

Based on our observations and experiences with the educators, it can be argued that the system that was well suited for participants to program and simulate 3D models, bringing unique advantages in understanding programming concepts through the visualization of the generated 3D models.

7 Conclusions

The results indicate that educators consideration towards the 3D printing simulation as a motivating tool to learning introductory programming was high from our M-SERI motivation survey, with an overall score of 4.458 (max of 5). This indicates this tool has the potential to be used in the teaching and learning of programming.

The results also indicate the educators level of agreement to the system being usable was high from our PSSUQ usability survey, with an overall score of 5.838 (max of 7). This indicates this tool is usable and is suitable for beginner programmers to use.

Simulating the print of the user-coded-3D models seemed to convince educators that it could be a great motivation for students. By bridging programming and 3D printing, educators could see students also having the possibility to explore emerging technologies like 3D printing, which is a very promising and exciting technology for both industry and education.

8 Future Work

Future work to this system include using different programming languages such as Python or Java. Another idea would be to able materialise designs with different machines and mediums such a CNC router and lazer cutting. Future work could also be designing and integrating a curriculum into the system so the user can learn programming from start. A creative competition could be had where students design a print artefacts for show and tell.

9 Acknowledgements

The author of this paper would like to thank Maria Nanabhai, my partner on the implementation of the project and Gary Stewart for his realiable guidence. He would

like to say thank you to Aslam Safla for his input as second reader of this project and to all the participants of the testing phase. Lastly, a special thank you to Andrya Blagusz for all her love and support.

References

[1] Evan Barba and Stevie Chancellor. 2015. Tangible Media Approaches to Introductory Computer Science. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '15)*. Association for Computing Machinery, New York, NY, USA, 207–212. https://doi.org/10.1145/2729094.27612

[2] C. Chytas, I. Diethelm, and A. 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)*. 1978–1987.

[3] Michelle Patrick Cook. 2006. Visual representations in science education: The influence of prior knowledge and cognitive load theory on instructional design principles. *Science education* 90, 6 (2006), 1073–1091.

[4] Todd I Ensign. 2017. Elementary educators' attitudes about the utility of educational robotics and their ability and intent to use it with students. (2017).

[5] Ilker Etikan, Sulaiman Abubakar Musa, and Rukayya Sunusi Alkassim. 2016. Comparison of convenience sampling and purposive sampling. *American journal of theoretical and applied statistics* 5, 1 (2016), 1–4.

[6] Anabela Gomes and António José Mendes. 2007. Learning to program-difficulties and solutions. In *International Conference on Engineering Education–ICEE*, Vol. 2007.

[7] Scott Grissom, Myles F. McNally, and Tom Naps. 2003. Algorithm Visualization in CS Education: Comparing Levels of Student Engagement. In *Proceedings of the 2003 ACM Symposium on Software Visualization (SoftVis '03)*. Association for Computing Machinery, New York, NY, USA, 87–94. https://doi.org/10.1145/774833.774846

[8] Scott Grissom, Myles F. McNally, and Tom Naps. 2003. Algorithm Visualization in CS Education: Comparing Levels of Student Engagement. In *Proceedings of the 2003 ACM Symposium on Software Visualization (SoftVis '03)*. Association for Computing Machinery, New York, NY, USA, 87–94. https://doi.org/10.1145/774833.774846

[9] S. Hodges, S. Sentance, J. Finney, and T. Ball. 2020. Physical Computing: A Key Element of Modern Computer Science Education. *Computer* 53, 4 (2020), 20–30.

[10] R Burke Johnson, Anthony J Onwuegbuzie, and Lisa A Turner. 2007. Toward a definition of mixed methods research. *Journal of mixed methods research* 1, 2 (2007), 112–133.

[11] Shaun K. Kane and Jeffrey P. Bigham. 2014. Tracking @stemxcomet: Teaching Programming to Blind Students via 3D Printing, Crisis Management, and Twitter. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 247–252. https://doi.org/10.1145/2538862.2538975

[12] Adams Nager and Robert D Atkinson. 2016. The case for improving US computer science education. *Available at SSRN 3066335* (2016).

[13] Thomas L. Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, and J. Ángel Velázquez-Iturbide. 2002. Exploring the Role of Visualization and Engagement in Computer Science Education. *SIGCSE Bull.* 35, 2 (June 2002), 131–152. https://doi.org/10.1145/782941.782998

[14] Nicola Papazafiropulos, L. Fanucci, Barbara Leporini, S. Pelagatti, and R. Roncella. 2016. Haptic Models of Arrays Through 3D Printing for Computer Science Education. In *ICCHP*.

[15] Blaine A Price, Ronald M Baecker, and Ian S Small. 1993. A principled taxonomy of software visualization. *Journal of Visual Languages & Computing* 4, 3 (1993), 211–266.

[16] Robert H. Seidman. 2009. Alice First: 3D Interactive Game Programming. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '09)*. Association for Computing Machinery, New York, NY, USA, 345. https://doi.org/10.1145/1562877.1562986

[17] Igor Verner and Amir Merksamer. 2015. Digital Design and 3D Printing in Technology Teacher Education. *Procedia CIRP* 36 (2015), 182 – 186. https://doi.org/10.1016/j.procir.2015.08.041 CIRP 25th Design Conference Innovative Product Creation.

10 Appendix

A Usability Questionnaire

Table 1: Usability Questionnaire

No.	Item Details
SYSUSE1	Overall, I am satisfied with how easy it is to use this system.
SYSUSE2	It was simple to use this system.
SYSUSE3	I could effectively complete the tasks and scenarios using this system.
SYSUSE4	I was able to complete the tasks and scenarios quickly using this system.
SYSUSE5	I was able to efficiently complete the tasks and scenarios using this system.
SYSUSE6	I felt comfortable using this system.
SYSUSE7	It was easy to learn to use this system.
SYSUSE8	I believe I could become productive quickly using this system.
INFOQUAL1	The system gave error messages that clearly told me how to fix problems.
INFOQUAL2	Whenever I made a mistake using the system, I could recover easily and quickly.
INFOQUAL3	The information (such as on-line help, on-screen messages and other documentation) provided with this system was clear.
INFOQUAL4	It was easy to find the information I needed.
INFROQUAL5	The information provided for the system was easy to understand.
INFOQUAL6	The information was effective in helping me complete the tasks and scenarios.
INFOQUAL7	The organization of information on the system screens was clear.
INTERQUAL1	The interface of this system was pleasant.
INTERQUAL2	I liked using the interface of this system.

B Motivation Questionnaire

Table 2: Motivation Questionnaire

No.	Item
UTIL1	This software sounds like problematic instructional technology to me.
UTIL2	Children should be introduced to this software in school.
UTIL3	Some day, I will use this software in my classroom.
UTIL4	This software can create more learning opportunities for students.
UTIL5	This software is a valuable educational tool.
ABIL1	I don't know how to use this software (i.e. this software is unfamiliar to me)
ABIL2	I am confident that I could learn to use this software for teaching.
ABIL3	I am unsure of my ability to integrate this software into my classes.
ABIL4	I believe this software will help motivate students.
ABIL5	I would enjoy using this software in my classes.
ABIL6	I think using this software in class would not work for me.
INT1	Some day, I will use this software in my classroom.
INT2	If given the opportunity, I would like to learn to use this software for instructional activities.
INT3	The challenge of using this software for instruction does not appeal to me.
INT4	I think this software is not interesting.
INT5	I will do as little work with this software as possible.
INT6	I think that integrating this software with teaching would take too much time.
INT7	I am willing to spend time setting up this software for instruction.

C Model3D User Interface

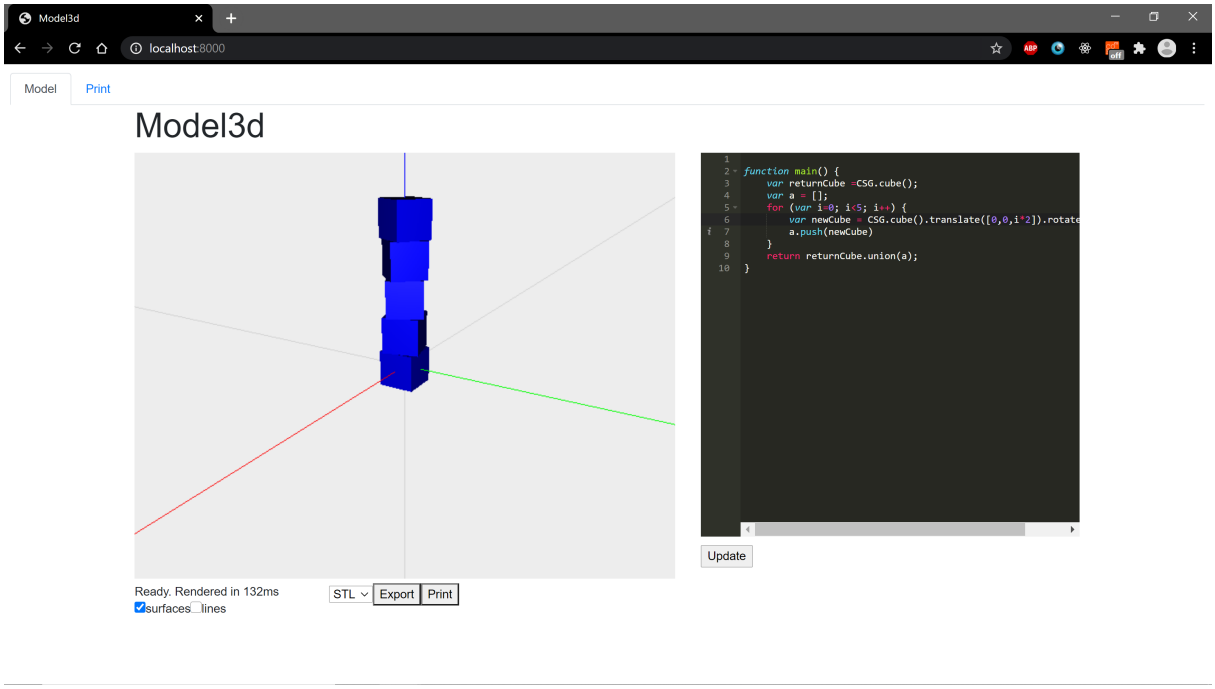

Figure 14: Overview of the Model3D user interface, where users can code a 3D model and view the rendering of the model.

D Print3D User Interface

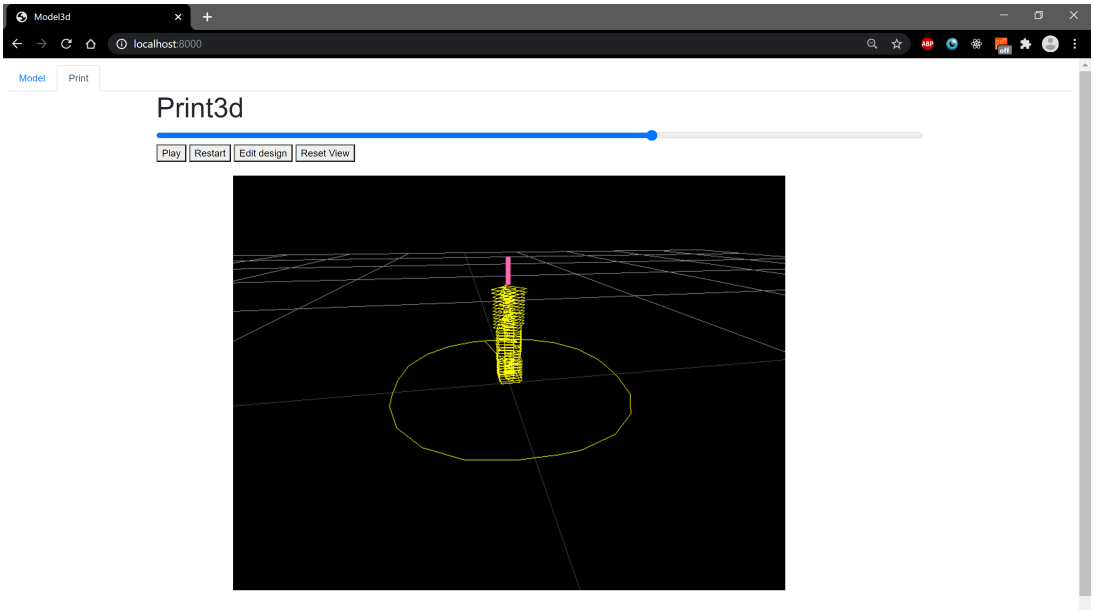

Figure 15: Overview of the Print3D user interface, where users can interactive and view the simulation of their models.