Understanding the Challenges of OpenSCAD Users for 3D Printing

J. Felipe Gonzalez
Carleton University
Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL
Lille, France
johannavila@cmail.carleton.ca

Thomas Pietrzak
Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL
Lille, France
thomas.pietrzak@univ-lille.fr

Audrey Girouard
Carleton University
Ottawa, ON, Canada
audrey.girouard@carleton.ca

Géry Casiez*
Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL
Lille, France
gery.casiez@univ-lille.fr

ABSTRACT

Direct manipulation has been established as the main interaction paradigm for Computer-Aided Design (CAD) for decades. It provides fast, incremental, and reversible actions that allow for an iterative process on a visual representation of the result. Despite its numerous advantages, some users prefer a programming-based approach where they describe the 3D model they design with a specific programming language, such as OpenSCAD. It allows users to create complex structured geometries and facilitates abstraction. Unfortunately, most current knowledge about CAD practices only focuses on direct manipulation programs. In this study, we interviewed 20 programming-based CAD users to understand their motivations and challenges. Our findings reveal that this programming-oriented population presents difficulties in the design process in tasks such as 3D spatial understanding, validation and code debugging, creation of organic shapes, and code-view navigation.

CCS CONCEPTS

• **Human-centered computing → Empirical studies in HCI**.

KEYWORDS

Programming-based CAD, OpenSCAD, 3D printing, Maker culture

ACM Reference Format:
J. Felipe Gonzalez, Thomas Pietrzak, Audrey Girouard, and Géry Casiez. 2024. Understanding the Challenges of OpenSCAD Users for 3D Printing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3613904.3642566

*Also with Institut Universitaire de France.

CHI '24, May 11–16, 2024, Honolulu, HI, USA

ACM ISBN 979-8-4007-0330-0/24/05…$15.00
https://doi.org/10.1145/3613904.3642566

1 INTRODUCTION

Computer-Aided Design (CAD) applications are used to aid design processes across various fields, including the rapidly growing 3D printing community. 3D printing technology allows individuals to design and fabricate objects easily and quickly in what is known as the digital personal fabrication practice [6, 57]. After identifying an idea or a need, makers create a digital model from scratch or retrieve it from model-storing websites [83], edit it, and print it in hours. Generally, users follow an iterative process going back and forth between the creativity, design, and printing stages until they achieve a satisfactory result [32].

Makers design using CAD applications that come in several flavors, integrate different technologies, and support different features. Most of the available applications, such as TinkerCAD [7], FreeCAD [80], or Fusion360 [34] use the **direct manipulation** interaction paradigm [74] where users can edit the models by making changes directly to their visual representation through simple metaphors such as drag-and-drop, menus, and buttons. Direct manipulation provides immediate feedback, incremental and reversible operations [73, 74], which enable a rapid learning curve [72].

A less popular category of CAD software use a **programming-based** approach that allows users to create 3D models by coding in a text editor with a specific programming language. Users must create scripts describing the models, and the system compiles and renders the result in a 3D viewer. Programming brings valuable advantages to 3D design. For instance, repetitive actions can be easily generalized, such as placing multiple elements in a specified pattern, through the use of programmatic structures like conditionals and loops. Moreover, programming allows the creation of very complex structured geometries, such as fractals or trees, through programming techniques like recursion. It facilitates the utilization of mathematical formulas, version control, and abstraction [86]. Our primary focus lies in CAD applications that embrace the programming-based approach to its fullest extent. Although some direct manipulation applications allow users to make modifications through code, as seen in FreeCAD with Python scripts [23], this paper excludes such cases from the category of programming-based CAD because the code primarily executes specific actions rather than serving as a comprehensive model description. In programming-based CAD, all model changes are reflected in the code, which always represents the complete model description. When modifications occur, the

system re-executes all scripts to generate a new 3D model. OpenSCAD [64], CadQuery [16], IceSL [49], and JSCad [63] are examples of programming-based CAD software.

Programming-based CAD applications are important players in CAD design and, specifically in the personal fabrication field. They present a solution to design complex structured geometries and offer a different paradigm from direct manipulation, allowing an alternative way of 3D designing [19, 38]. However, the potential of programming-based CAD may be underestimated and its challenges may not have been adequately studied. Previous research on user behavior in 3D printing design has focused on investigating improvement opportunities for CAD almost exclusively in direct manipulation applications [43, 45, 53, 84], with findings specific to this context. For instance, Mahapatra *et al.* [53] studied the barriers to taking measurements of objects, transferring them to CAD applications, and manipulating this information digitally. Some of the difficulties classified as "*Digital*" are specific to direct manipulation applications such as "*3D camera causes manipulation errors*". Furthermore, the literature related to programming-based CAD applications is limited. Previous research has examined the limitations of understanding and navigating code in programming-based CAD [27] and its potential for students to learn programming through 3D design [38]. However, these studies address specific task difficulties within a limited scope of the 3D printing design experience. A better understanding of the user experience with these applications, their advantages, and the problems users face when 3D printing is still missing. We have set out to investigate the following research questions: What are the motivations and challenges of using programming-based CAD? What are the current limitations of these applications? In the context of 3D printing, how do the challenges previously identified in direct manipulation CAD applications relate to the ones present in programming-based CAD applications? There is an interesting opportunity for HCI to investigate the current challenges programming-based CAD users face to facilitate the design process and possibly lower the entry barrier for newcomers.

This paper investigates how users of programming-based CAD software experience the design and fabrication process. We conducted semi-structured interviews with 20 users of the most popular programming-based CAD software in 3D printing for personal fabrication, OpenSCAD [52, 59]. We asked participants about their design experiences with programming-based CAD, direct manipulation applications, and comparisons between them in the design and printing process. Additionally, we draw on previous work to explore their motivations and contrast barriers found in 3D printing with direct manipulation applications. Specifically, we included questions related to problems measuring physical objects to create digital designs [53] and limitations working with pre-existing models from websites [2]. We also included a short hands-on exercise to observe design workflows and difficulties [27, 86]. Based on the findings of the interviews, we provide a comprehensive analysis of the preferences of programming-based CAD users, current challenges in the design and printing process, and desired features expressed by the participants to improve these applications.

Our contribution is a qualitative analysis that aims to provide a comprehensive understanding of the programming-based CAD population. Specifically, we examine their motivations, design challenges, and challenges in the application field of 3D printing. Our findings suggest that programming-based CAD users, often with a long programming experience and a programming-oriented mindset, face significant difficulties measuring and designing organic and curve shapes, mentally connecting the code with the view, performing spatial transformation due to the required mathematical skills, and addressing uncertainty when 3D printing.

2 BACKGROUND

We describe some technological aspects of CAD software to frame our findings. We begin by defining what a programming-based application is. Then, we differentiate between the workflows of CAD applications, specifically parametric and direct modeling. Additionally, we explain the two primary data representations that CAD applications use. Finally, we introduce the CAD application we used for this study, OpenSCAD.

2.1 Programming-based CAD

Programming-based CAD refers to the applications that allow users to describe models entirely through coded instructions while the system renders the result in a view. In particular, the code represents the full description of the model, and any edit of the model is described in the code. Text-based applications such as JSCad [63], BRL-CAD [17], and OpenSCAD [64] fall into this definition. CAD applications using visual programming such as BlockSCAD [35] or Grasshopper [20] can also be considered as programming-based CAD.

Some direct manipulation applications allow users to modify the model with code. McGuffin and Fuhrman [55] present a taxonomy of applications using different interaction paradigms, which can be applied to CAD. *Content-Oriented Programming* paradigm enables the output to be affected by instructions and direct manipulation. Blender [22] is an example of such an application, with which users create and edit meshes in the view and run scripts for specific actions with a code editor. Another category that uses both direct manipulation and coded instructions is *Programming By Example*, where direct manipulation interactions generate instructions that the user can execute later to edit the model. For instance, FreeCAD [23] provides a console where actions in the view generate corresponding Python code statements that the user can run later. These approaches present a fundamental workflow difference compared to programming-based applications. The code is used to perform specific actions, but it does not comprehensively describe the 3D model visually represented. In programming-based, a modification requires the user to go through the code, edit it coherently, and re-execute all the scripts to re-generate the model; in the other paradigms, the code executes specific actions to modify the current state of the model.

McGuffin and Fuhrman also present the *Bidirectional Programming* applications, where users can modify the output with both direct manipulation and instructions. In these applications, every change in the output through direct manipulation results in an update in the code to keep coherence, such as presented in the scalable vector graphics (SVG) environments Sketch-N-Sketch [28] or Twoville [37]. Bidirectional programming CAD applications

[27, 39, 40] follow the definition of the programming-based CAD paradigm by always keeping the code as a full description of the model, but they also extend the interaction capability by allowing direct manipulation interactions in the view. Antimony [40] is an example of a programming-based CAD that allows users to use visual programming to create and edit the model in the view while the system updates the instructions coherently. Despite the potential benefits of these applications, they seldom seem to be used, given the difficulty of finding models online.

2.2 Parametric and direct modeling

CAD applications can be divided into two groups based on how the system stores and executes changes in the model [87]: parametric modeling and direct modeling. Parametric modeling, also called feature-based or history-based modeling [4, 42], allows the user to describe the model in a re-executable set of steps defined by parameters. Parameters are adjustable to create new versions of the model by re-executing the steps. Programming-based CAD is naturally parametric due to the way coding works, having arguments as input that determine the output. On the other hand, direct manipulation applications that apply a parametric approach commonly allow users to define constraints and perform operations to the model, also called features, stored in a *history tree*. Modifiable parameters often control these features. Thus, besides verifying the history of the design process, the user can perform edits on the history tree and re-execute it. Consequently, users can obtain solid model variants by editing parameters embedded in the model [87]. FreeCAD [80] is an example of parametric modeling. In contrast, direct modeling allows users to edit the model without worrying about the history of these edits. The system only captures the current state of the model. Consequently, users gain flexibility, such as not worrying about features and the impact changes may have on their inter-dependencies [14], by renouncing to revisit their steps [14]. Tinkercad [3] is an example of a direct modeling application.

2.3 Data representation

Another distinguishing aspect of CAD applications is how the geometric data is represented and stored. We distinguish between programs using constructive solid geometry (CSG) and boundary representation (B-rep). In CSG, a solid is represented as a set of primitive solid objects (*e.g.* spheres and cubes), transformations (*e.g.* scale or mirror), and boolean operations (*e.g.* union or intersection) as depicted in Figure 1b. Both the surface and the interior of an object are implicitly defined. In other words, there is no description of specific geometric properties, such as points, edges, or positions, but abstract descriptions of primitives and operations. CSG objects are always *watertight* and manifold [79] if the primitives are [30]. Therefore, CSG provides closed and well-formed geometries that are printable, making CSG attractive for 3D printing [25]. On the other hand, a boundary representation (B-rep) describes only the oriented surface of a solid as a data structure composed of vertices, edges, and faces. B-rep can be efficiently rendered on a graphic display system, allowing easy differentiation between the vertices, edges, and faces (Figure 1a). This offers more flexibility than CSG and allows for valuable operations in 3D printing, such as chamfering, blending, or drafting [25, 30]. OpenSCAD [64] uses a CSG representation while FreeCAD [80] uses a B-rep.

2.4 OpenSCAD

OpenSCAD is an open-source parametric CSG programming-based CAD application. Users can describe 3D models in its *functional declarative* programming language using the text editor, and the system compiles and renders the scripts in a viewer. Although OpenSCAD has applications in various domains [77], it provides mainly 3D printing-oriented features. For instance, OpenSCAD preferences menu offers features such as connecting with OctoPrint [33], a web interface for controlling consumer 3D printers or export models into the standard STL format for 3D printing [62]. OpenSCAD aims to give full control over the design by being purely programming-based. It is the most popular of the programming-based CAD applications, which are mainly parametric CSG applications such as IceSL [49], JSCad [63], BRL-CAD [17], ImplicitCAD [51], or RapCAD [8].

OpenSCAD is not an interactive modeler and does not focus on the artistic aspects of 3D modeling but on the CAD aspects [64]. In addition to CSG modeling techniques, it allows the extrusion of 2D outlines. It also provides a *preview* mode that generates approximations for rapid visualization and a *render* mode that generates exact geometries using longer rendering times. In preview mode, OpenSCAD allows the user to right-click on the models in the view to display a menu of the CSG elements that create the clicked part. The user can click on a menu item while the system places the text cursor in the line of code that creates the CSG element. Moreover, OpenSCAD language includes *modifiers* for debugging. Modifiers are specific characters that can be placed at the beginning of code statements to ignore, highlight, or isolate elements in the view[1]. Finally, OpenSCAD handles command-line arguments and is not limited to the graphic user interface.

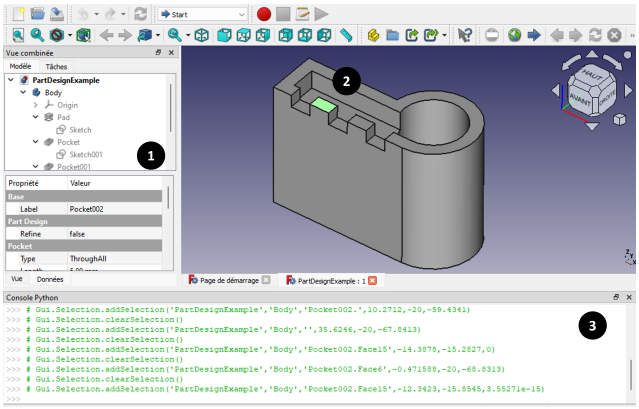
3 RELATED WORK

Creating 3D models is an essential part of the 3D printing process. Makers can create a model from scratch using specialized CAD software. Furthermore, they can get a pre-existing model from model-storing websites to print it as is or edit it in a CAD application to get a customized version. We describe previous work investigating end-user behaviors and challenges in 3D design and 3D printing in such scenarios. We start by looking at lessons learned from work on direct manipulation programs, and we continue with related work on programming-based CAD. Finally, we describe some existing problems with model-storing websites.

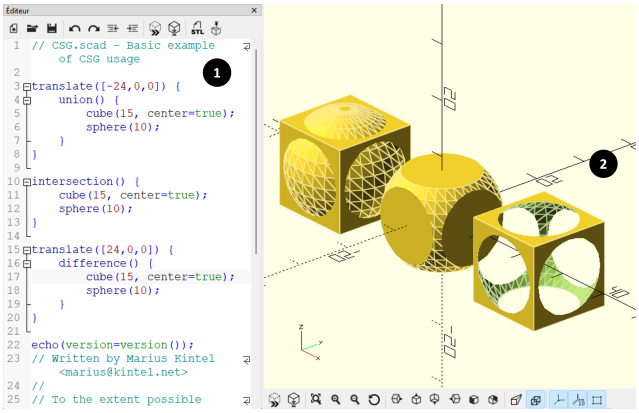
3.1 Modeling with direct manipulation programs

The direct manipulation paradigm facilitates user interaction by reducing the cognitive resources required to understand and use user interfaces [1, 75]. The actions must be fast, incremental, and reversible, while the objects of interest must be visible and directly

[1] https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/Modifier_Characters. Accessed: 11/12/2023



(a) FreeCAD is a parametric direct manipulation software using B-rep. (1) Users can check the history tree to revisit their steps. (2) They can also interact in the view, individualizing vertices, edges, or faces. (3) FreeCad provides a Python console in which users can execute specific actions through code.



(b) OpenSCAD is parametric programming-based CAD software using CSG. (1) Users can describe the models through primitives (*e.g.* sphere), transformations (*e.g.* translate), and boolean operations (*e.g.* union) in a text editor. (2) The system compiles and renders the result in a 3D viewer.

Figure 1: Example of two parametric CAD applications.

manipulable [73, 74]. Hence, this paradigm guarantees several usability benefits, such as recovering from errors and learning how to use the interface [72]. However, direct manipulation presents well-known challenges, such as difficulties in performing repetitive actions, manipulating small objects (especially in high-density spaces of objects), or manipulating intangible properties (abstract properties without visual form) [24, 47].

In 3D printing CAD, Hudson *et al.* [32] studied novices interacting with TinkerCAD and reported that manipulating elements in a 3D space through a 2D screen can confuse and lead to errors that are known problems in other digital applications [21]. Similar difficulties related to spatial thinking skills have also been reported in children using TinkerCAD [9]. Specifically, problems related to understanding the 3D perspective, understanding rotation in a 3D space, using the correct primitive shapes, and grouping primitive shapes. Programming-based CAD applications can present similar problems by having a 3D viewer rendering the output.

An everyday use of 3D printing is to repair [32] or to augment objects [6]. In such cases, measurements of physical objects, their transfer, and their meaningful use in the design are necessary. Mahapatra *et al.* [53] study how self-identified novice users capture physical measurements, transfer them to digital design, and verify their accuracy. The study carried out on TinkerCAD reported several obstacles when novices create a digital replica of an object, classified into three groups according to the moment they occur: 1) *Physical* when capturing the data, 2) *Digital* when using the data in the digital design, and 3) *Transition* when transferring (from physical to digital) or evaluating (from digital to physical) the data. Although physical obstacles are independent of the modeling CAD, digital and transition may differ when using a programming-based application. Probably some challenges may persist (*e.g. "3D camera causes confusion"*), others may occur differently (*e.g. "Relative placement problems"*), and others may not make sense in programming-based CAD (*e.g. "Miscalculating by hand"*). We take inspiration from this work to explore what problems programming-based CAD users face when measuring, transferring, and using data in the design process.

3.2 Modeling with programming-based CAD

Significant efforts have been made to understand the difficulties in learning programming languages [44, 67]. Expectedly, programming-based CAD users may present similar challenges, such as difficulties in structuring and breaking down the problem into smaller problems, difficulty finding the features the programs offer, or documentation problems.

Yeh and Kim [86] report problems with programming-based CAD and direct manipulation software offering scripting features for 3D design. These problems include difficulties reading code, re-using code, aligning objects, selecting parts, refactoring code, and 3D printing. Unfortunately, these undetailed findings were obtained from undocumented feedback from novice students with OpenSCAD and online forums such as StackOverflow. More recently, Gonzalez *et al.* [27] investigated challenges in OpenSCAD related to navigating and editing models. The findings include difficulties in linking the code to the view and performing spatial transformations. We aim to cover a broader scope of the 3D design, 3D printing, and re-using pre-existing models experience with OpenSCAD.

3.3 Sharing and re-using models.

Sharing is a keystone in the 3D printing community [46]. Multiple model-storing websites allow authors to upload models to share with other users, such as Thingiverse [83], MyMiniFactory [58] or Printables [5]. Some of them allow authors to upload coded parametric models that expose widgets so other users can modify parameter values for the system to create a customized version of the models [71, 82]. Thingiverse, with its Customizer application [82] is an example of these solutions that store OpenSCAD models. Oehlberg *et al.* [61] reported how, after a year of the release of

Customizer, about 40% of the Thingiverse models were created from parametric models. These findings depict how some users re-use models to remix them by changing parameters. However, it is unknown if the authors of these parametric models adopt the re-using practice in their models. We aim to understand the role of model-storing websites in programming-based CAD design.

In summary, some previous research has explored end-user experiences in direct manipulation programs, with findings not always applicable to programming-based CAD. Furthermore, there has been limited exploration of programming-based CAD problems and the sharing and re-using practice. Our work addresses these voids and contributes to a better understanding of this population in depth.

4 METHOD

We conducted twenty semi-structured interviews to understand the motivations and challenges of OpenSCAD users empirically. The interview was divided into three parts. First, we asked participants for demographic information. Also, we asked them to self-rate their skill level on a scale from one to five, one meaning novice and five expert, on direct manipulation CAD applications, programming-based CAD applications, and general programming languages outside CAD. Similarly, we asked participants to self-rate their skill level in OpenSCAD on the same scale. The responses are reported in Table 1.

In the second part, we asked participants open questions about their experience in 3D printing and 3D modeling. Specifically, we were interested in understanding the motivations of makers in using OpenSCAD for 3D design, the challenges and limitations they face using OpenSCAD for 3D printing, their perception of direct manipulation programs compared to OpenSCAD, and ideas to improve OpenSCAD that might apply to other programming-based CAD applications. Furthermore, we draw on previous work on understanding the complexity and challenges of 3D modeling in direct manipulation programs to contrast these findings with the experience of OpenSCAD users. Concretely, we have included questions related to difficulties measuring physical objects and transferring data to digital designs [53], and sharing and re-using models in model-storing websites [2, 86].

Finally, we wanted to understand the limitations of more specific actions when designing in OpenSCAD. We decided it would be easier to study participants' behavior in a real scenario while we observe them instead of only asking them to describe how they use the software, which could lead to easily missing specific actions or strategies they use. Thus, we asked the participants to perform a short hands-on exercise in the third part to observe their behavior while performing tasks. Based on the findings of previous work, we report problems in programming-based CAD related to selecting specific parts to apply operations, including challenges in reading, navigating, refactoring, and understanding code [27, 86]. If possible, we asked the participants to bring one of their own OpenSCAD models to the interview. P2 did not provide a model, so we used the example `candleStand.scad` provided by OpenSCAD.

The participants explained the motivation behind the model and went through their code, discussing difficulties and how they modeled their object. We asked them to perform search tasks replicating the need to select a part to modify it or apply an operation. We pointed at specific parts in the 3D view and asked the participant to locate the lines of code that created them. We asked participants to think aloud while we carefully observed the process, recurrent behaviors, and strategies. We paid special attention to the software features they used, the typical patterns they followed to perform the tasks, and the errors they made. Last, we discussed ideas they could have to improve their experiences in OpenSCAD.

The interviews lasted approximately 60 minutes on average. The questionnaire used is included in the Appendix. We took notes of their answers and the observed behaviors during the hands-on exercise. The experiment protocol was examined and approved by the ethics board in our laboratory.

4.1 Recruitment and Participants

We relied on the common use of 3D modeling in research and the active sharing nature of programming and maker communities on social media. We recruited participants from research laboratories and OpenSCAD channels on Reddit (`r/openscad`) and Facebook (`OpenSCADAcademy`) to conduct the semi-structured interview using video conferencing or in person. The only requirement was having enough experience with OpenSCAD to read and write code, but we also expressed that having 3D printing experience would be an asset.

We report participant demographics and experience in Table 1. All the participants self-identified as male and varied in age: one was between 20 and 29, four were between 30 and 39, seven were between 40 and 49, four were between 50 and 59, and four were between 60 and 69 (average: 48.0, standard deviation: 11.7). All participants, except P8, had three or more years of 3D printing experience (average: 7.9y, standard deviation: 3.8). Except for P13 and P18, all participants self-rated with four or more in at least two programming languages. Moreover, all participants, except P20 mentioned having experience with direct manipulation CAD programs. Only five participants self-rated their direct manipulation CAD application skills with four or more. Regarding experience with other programming-based CAD applications or applications that allow scripting, only six participants expressed having any, and only P2 and P12 self-rated their skill level above 3 in one of those applications. Finally, participants self-rated their skill level with OpenSCAD as follows: One participant with 1, one participant with 2, six participants with 3, eight participants with 4, and four participants with 5.

4.2 Data Analysis

We followed a Reflexive Thematic Analysis (RTA) [11, 15] approach in an iterative coding process. Our study aims to understand the user experience using OpenSCAD, and part of the data collected included behavioral observations from the hands-on experience. Thus, we opted for a data analysis approach suitable for these studies [12, 15] that allows flexible participation of the researcher's interpretations rather than other qualitative analysis approaches such as code reliability or ground theory [10, 78].

We uploaded the interview data into the MaxQDA data analysis software [26]. One of the researchers performed an inductive analysis to develop a set of codes by coding the first ten interviews.

Table 1: Demographics and self-rated skill level in CAD programs and programming languages. Participants self-rated their skill level on the scale: 1 (Novice), 2 (Advanced Beginner), 3 (Competent), 4 (Proficient), 5 (Expert). The level reported in the category *Others* is the highest rank expressed by the participant among the options. *Direct manipulation CAD others: LibreCAD, Sketch Up, AutoCAD, Curve3D, OnShape, Catia, SolidWorks. **Programming Language Others: Prolog, MaxMSP, PureData, Ruby, GoLink, MatLab, Cobol, Pearl, Pascal, Groovy, TypeScript.

				Direct Manipulation CAD						Programming based CAD						Programming languages							
Participant	Age Range	3D printing experience (y)	OpenSCAD	Blender	FreeCAD	Fusion 360	TinkerCAD	Rhinoceros	Others*	IceSL	Python API	JsCAD	CadQuery	BRLCAD	BlocksCAD	C	C++	C#	Java	JavaScript	Python	PHP	Others**
P1	50 - 59	10	3	1	1				1							4			4	4	4	4	4
P2	40 - 49	6	1	3						5	2						5			3	4		3
P3	20 - 29	7	4		2	4			1								4				4		5
P4	30 - 39	5	3		3	3										4	4	4	4		4		4
P5	40 - 49	6	4						4							4		4	4				3
P6	40 - 49	15	4				2											4	2		4	3	
P7	30 - 39	8	4			3						3	3			4			4		4		
P8	40 - 49	1.5	3				2									4	4			5			5
P9	30 - 39	14	3	4	3			4	3								4			5	3		3
P10	60 - 69	8	5				2	1									5				5		5
P11	40 - 49	7	4		1							1								5	5	5	
P12	50 - 59	13	4		1		1		1			4		4	4								4
P13	60 - 69	5	3		1	2							1			1					1		
P14	60 - 69	4	5	1	3		5												2	4		4	3
P15	40 - 49	8	3	1			1		1											4	4		4
P16	50 - 59	6	4						1							4	4			4	4		
P17	30 - 39	15	5	4		4	4									2	2			4	4		
P18	60 - 69	8	2		1	1	1																
P19	40 - 49	3	4			1	1												5		5		3
P20	50 - 59	9	5								1							4			4		5

Then, the coder started grouping codes by recognizing recurring patterns and identifying codes describing a central concept to create subthemes and themes [13]. To achieve a richer interpretation of the coding process [12], a second researcher performed a deductive thematic analysis on a randomly selected interview. The second coder used the codes created by the first coder in this interview and could create new codes when necessary. Then, both coders discussed the disagreements and refined the codes by removing, merging, changing, or adding new codes. After re-organizing codes, subthemes, and themes, the first ten interviews were re-coded with the resulting set of codes. In the second iteration, the first coder continued the inductive analysis with the next five interviews, followed by the deductive coding from the second coder, discussions on the codes, refinement of the codebook, and re-coding of the interviews. A third iteration was performed to complete the coding of the total of interviews.

Although RTA does not seek reliability coding [11], we were interested in tracking the level of agreement between coders. We calculated Cohen's kappa index in every iteration to verify inter-coder reliability [54]. At the end of the coding of all interviews, we achieved a *substantial* [48] agreement: iteration 1 $\kappa = 0.543$, iteration 2 $\kappa = 0.592$, iteration 3 $\kappa = 0.617$.

Most of the codes were created in the first fourteen interviews, achieving a potential code saturation. However, it was not until the seventeenth interview that codes, themes, and subthemes found in the codebook did not have substantial changes, and their meaning was well established, achieving meaning saturation [29].

5 THEMES

We created 266 individual codes to code a total of 783 segments of our notes. We grouped codes into subthemes and then into three main themes. *Programming-based CAD user profile* theme groups 22 codes (59 segments coded) and 9 subthemes as depicted in Table 2. Table 3 depicts the theme *Design* that covers 193 codes (632 segments coded) and 80 subthemes. Finally, *Printing* theme includes 51 codes (92 segments coded) and 20 subthemes (Table 4).

5.1 Programming-based CAD users profile

We start by discussing the design experience of the participants in OpenSCAD. Later, we discuss why participants use a programming-based CAD and their opinions on direct manipulation programs.

5.1.1 Experience with OpenSCAD. Participants ($n = 10$) mentioned the advantages of using OpenSCAD. The first is related to the parametric capability of programming-based CAD. P17, for instance, discussed his work in a laboratory making prototypes "*it was useful to have this programmatic base to create arbitrary variations of similar things*". Participants found it helpful to define complex geometries through mathematical definitions instead of storing high volumes of data when having geometric information, as happens in direct modeling. P2 worked with robotics and talked about his needs "*I want compact shape descriptions that generate highly complex geometries (...) we don't want to store all the geometry with triangles*". Programming-based CAD also helps to generalize models better. For instance, P2 mentioned that resizing a robotic articulated arm involves more than just geometric scaling. The

Table 2: Structure of theme *Programming-based CAD user profile*. Color intensity is proportional to the number of interviews coded with codes of the theme and subthemes.

Theme	Subtheme		
Programming-based CAD users (n = 18)	Experience with OpenSCAD (n = 13)	Advantages (n = 10)	OpenSCAD is popular and open source (n = 4)
			Programming features make design easier (n = 3)
			Easy to describe complex shapes (n = 2)
			Parameters allow a flexible definition of objects (n = 3)
		Disadvantages (n = 7)	
	Programmer mindset (n = 13)		
	Direct manipulation mindset does not work for me (n = 6)		

operation may require adding another articulated section, and describing this behavior in direct manipulation is difficult. Moreover, programming features such as abstraction allow participants to re-use work. Further, participants found it convenient that OpenSCAD is open-source, runs on all major operating systems, and is the most popular programming-based application with community support. P12 mentioned "*It's also the most popular program. If you find a problem, someone else already had it and you can just copy the code or see a different approach*".

However, some participants ($n = 7$) identified liabilities of using OpenSCAD. Despite the support community, they feel that the development of the application is slow. P4 mentioned trying to contribute to the project on GitHub, but multiple pull requests have been on hold for a long time without being integrated into the application. Moreover, they found that the available features are too basic and the rendering time of complex models inconveniently long.

5.1.2 Programmer mindset. All the participants except P18 (Table 1) had a programming background and found the 3D modeling programming-based paradigm convenient. P19 expressed "*I discovered OpenSCAD. I was like, "Oh, hey, this is just models and software" (…) It was beneficial to be able to develop parametric models in code, which was part of my skill set.*". Programmatic interfaces are a good entry door to 3D printing for this population.

Participants ($n = 13$) also mentioned that OpenSCAD also fits their mathematically oriented mindset. Interestingly, P12 and P15 expressed that despite having a programmer mindset, one of their problems was their lack of math skills. P12 said "*I'd like to say I'm an expert in OpenSCAD (…) The one thing that bugs me down is the math. I always get stuck on that*".

5.1.3 Experience with direct manipulation programs. Six participants commented that the direct manipulation paradigm did not work with their mindset. P11 said "*I'm not a visual guy, not really an artist. I can imagine what I want to do and write it down without a preview*".

Interestingly, some participants ($n = 4$) thought that they would inevitably need to learn direct manipulation applications due to the perceived limitations of programming-based ones. However, some of them have succeeded without learning a new application. P5 commented "*I've always said to myself, I'll learn AutoCAD when I need it, and so far I haven't needed it.*"

5.2 Design

We discussed several aspects of the design process and how it relates to the other stages in the fabrication process.

5.2.1 Working with existing objects. Often, makers fabricate objects that will interact with other objects, such as in the case of repairing or augmenting an object. Participants shared their experiences in such cases.

Linear measuring. The preferred measurement tool for linear measurements is the digital caliper. However, two participants stated that the task of measuring increases uncertainty and leads to more iterations. P8 mentioned "*If I was better at taking measurements, I would go through fewer iterations, and my first print would probably be closer to what I want*". Moreover, calipers have size limitations, according to two participants. P14 commented "*calipers don't go big enough to measure a lot of this stuff*".

Measuring organic shapes and curves. In addition, measuring organic or curved shapes is complex ($n = 11$). For instance, P15 commented on their work on repairing parts: "*it's rectangular and then there's a curve. I had to print it like 15 times to get that right.*". To deal with it, seven participants reported using creative solutions. For example, P12, P14, and P20 have used cameras or scanners to get the outline of a shape and use it later in the design by transforming it into an SVG or by measuring the outline and approximating the curves. P14 commented "*I photocopied the object, I put it down on a photocopier, so I could get a picture of the rim of the profile, from which I could make measurements of it. Then I had to run an optimization program in a spreadsheet to figure out how everything fits together, how all the curves match, and what angles join each other…*". P14 also reported using photogrammetry with no good results. "*I tried photogrammetry to make a 3D model of this; it just doesn't work if you have shiny or transparent surfaces. I got a nice model but also sort of a cloud of nondescript points because of all the reflections on the surface*". Other participants have tried to measure some points to interpolate by guessing in a trial-and-error strategy. P12 mentioned using a contour gauge to approximate curves and P14 said that at some point, he would hold the physical object in front of the screen to see if the object to print would match.

Table 3: Structure of theme *Design*. Color intensity is proportional to the number of interviews coded with codes of the theme and subthemes.

Theme	Subtheme			
Design (n = 20)	Working with existing objects (n = 13)	Physical Measuring (n = 12)	Linear measurements (n = 11)	
			Curves, round, and organic shapes (n = 11)	I use non orthodox methods (n = 7)
				I prefer not to deal with it (n = 4)
				Challenges (n = 6)
		Making things fit (n = 5)		
	Manipulating the model (n = 16)	Spatial actions (n = 15)	In direct manipulation programs (n = 2)	
			In programming-based CAD (n = 15)	
		Parametric modeling (n = 11)	In programming-based CAD (n = 9)	
			In direct manipulation programs (n = 6)	Advantages (n = 3)
				Disadvantages (n = 3)
	Achieving digital precision (n = 9)	In direct manipulation (n = 2)		
		In programming based CAD (n = 9)	OpenSCAD is convenient (n = 3)	
			Verifying (n = 8)	
	Creating organic and curved shapes (n = 10)	In programming-baed CAD (n = 9)		
		In direct manipulation programs (n = 2)		
	Screen problems (n = 2)			
	CSG (n = 7)	Not possible to individualize points or faces (n = 3)		
		Subtracted parts (n = 6)		
	Versioning and collaborative work (n = 3)			
	Repetitive actions (n = 4)			
	Programming-based specifics (n = 20)	Working with several files/parts (n = 9)		
		Dealing with code (n = 18)	Code practices (n = 14)	
			Challenges (n = 7)	
			Code editor (n = 6)	
		Code-view navigation, selecting parts and understanding (n = 20)	Reading the code to understand (n = 18)	Relate objects with something I know (n = 4)
				Read and analyze the code (n = 13)
				I remember what I coded (n = 12)
			Guessing from the view (n = 1)	
			OpenSCAD search (n = 4)	
			Text Search feature (n = 5)	
			Errors (n = 4)	
			Other challenges (n = 8)	
			Using modifiers (n = 15)	
			Changing colors (n = 3)	
			Removing objects temporarily (n = 10)	
			Trial and error (n = 9)	
		Challenges (n = 7)		
	Other challenges (n = 3)			
	Improvement opportunities (n = 17)	Spatial information extraction (n = 7)		
		Interactive editing (n = 8)		
		Interactive selection and navigation (n = 11)		
	Reusing models or Design from scratch (n = 20)	My pipeline (n = 10)		
		Design from scratch (n = 12)	I enjoy doing it, wanna do it better (n = 5)	
			My needs are too specific for pre-existing models (n = 6)	
			It is easier, faster, or better quality if I do it (n = 4)	
		Reusing models (n = 20)	Pre-existing models do not fulfill my needs (n = 16)	
			Model storing websites and its offer of models (n = 7)	Limitations with parametric models (n = 4)
				Search engines and availability (n = 6)
				Licenses restrictions (n = 2)
			Pre-existing models are time savers (n = 7)	
			Pre-existing models are for inspiration (n = 8)	
			Re-using meshes (n = 14)	It is difficult to edit meshes (n = 4)
				STL files are usually broken non manifold (n = 11)
			Re-using code (n = 10)	Code models are more flexible to re-use (n = 3)
				Libraries (n = 2)
				Available code quality, understanding other people's code (n = 5)
				I adapt pre-existing code to my models (n = 4)
		Sharing models (n = 8)		

Some participants ($n = 4$) prefer to avoid organic shapes and curves. P20 commented that "*I try to avoid them (curves and organic shapes) in my designs. If I'm just talking about rounding corners, I pick up a set of radius measurement tools so I can get the correct size of rounded edges. Beyond that, it's trial and error.*"

Using digital replicas. It is easy to miss context elements when fabricating objects interacting with other things ($n = 5$). Makers cannot imagine every aspect of the physical objects in the 3D view to see if the design will satisfy their needs. P9 explained " *this piece is a cover for an emergency button that you need to screw in manually. I did not think about it in my first iteration, so I could not access the screw hole and had to repeat it, adding a small hole*". Some participants create a digital representation of the physical object to have a reference to work with, making them more confident about the design decisions. For instance, P11 used an STL model in a project for his phone: "*I've used a model of my phone for that. I just use an STL of the phone and design around it*".

5.2.2 Spatial transformations. The keystone modeling action is manipulating objects' position, orientation, and size. Participants discussed the difficulties they usually face when performing this task in OpenSCAD and compared it with direct manipulation applications.

Some participants ($n = 2$) acknowledge how easy this task is in direct manipulation applications. P19 commented: "*With Fusion360 or TinkerCAD (it is handy) to make a cylinder of the right dimension in the right place, basically drag and drop and get it where it belongs*". On the contrary, the same task in programming-based CAD is reported to be very difficult ($n = 15$). Trying to place an element in the right place can be challenging, as commented by P7 "*Most times, my difficulty is not drawing but where it should be drawing. Like I aim to draw a sphere here and OpenSCAD put it over there, and I am like "why?"*".

It seems to be challenging to relate the transformation parameters of a code statement with the spatial coordinates in the view without visual cues. For instance, if a translation is applied with 10 units in the second parameter on a sphere, it is not easy to predict the direction the sphere will move in the view. The camera view can be in a position and orientation that may make it hard to understand where the axes are located and there is not enough visual help for the user. OpenSCAD view has a widget representing the canonical x, y, and z axes directions. However, users often need to locate coordinate systems different from canonical ones. The position and orientation of objects are typically the result of multiple nested spatial transformations. Each transformation has a scope where the relative coordinates system does not match with the canonical one, so the widgets are useless. P15 said "*if you are creating some volume (…) it is difficult to predict, with all the operation, where they land and what precise coordinates would be*".

Participants also found dealing with translations and rotations of the same object challenging. These spatial transformations are not commutative. In other words, applying a `translate` after a `rotate` would not give the same result if the commands are executed in the opposite order. P19 commented "*you need to think about how you want to translate and rotate it before you can even get it to where you want. (Otherwise) You might find that you get your translation operations out of order, and all of a sudden, you're in the wrong place*". To deal with this, participants use a trial-and-error technique. P6 mentioned "*If you would ask me right now, to rotate this in a certain direction, I would not, without testing, be able to tell you what combination of the three parameters I need to get it in the direction I want*". P9 commented on having a specific order for transformations: "*I rotate first and translate later. It is easier because when I translate before rotation, sometimes the rotation center is not the same as the object's center.*" The participants proposed another strategy implementing position and orientation checkpoints. They correctly generated the transformation required for placing objects where they belong. The elements were then designed in the canonical coordinate system, and after completion, the participants applied the previously calculated transformation. P17 commented "*I remember doing like a checkpoint where I start. I make sure that everything starts from this origin point. Then, when you add in multiple modules, make sure that they're centered around so you don't have to keep track of all the different systems.*". Similarly, P16 commented that he created modules solely to place elements in a location and orientation of interest. Some participants mentioned avoiding having several layers of transformations because it becomes unmanageable.

In addition, it was deemed challenging to calculate the appropriate parameter values for the spatial transformations. As objects' position and orientation are built upon multiple transformations and involve the sizes of other objects, the coordinates system changes constantly, and participants must mathematically derive parameters' values of spatial transformations. Depending on the previous spatial transformations, the relative coordinate system varies. P14 commented "*…I have to painfully mathematically calculate where that plane is in space, its slope, and where its normal vector is, and then get the surface on there…*".

5.2.3 Parametric modeling. The creation and use of parametric models were reported as valuable for the participants ($n = 11$). For instance, P14 stated that direct modeling is not enough for serious developments "*TinkerCAD is easy and fun, but not useful for parametric modeling or anything serious.*". Further, they value the possibility of revisiting their steps. P7 said "*because everything is written down … if I pick up something one year later, I'll remember exactly what I did.*". Moreover, they made clear differences between creating a parametric model in direct manipulation software and OpenSCAD.

Six participants shared their thoughts about creating parametric models through constraints in direct manipulation applications such as FreeCAD and Fusion360. The ability to select objects of interest directly from the view is perceived as practical, as mentioned by P14 "*you can grab a vertex and snap it to some other location and not have to worry about numbers and measurements so much, but you can make things fit just by moving things around and snapping things to grid points or to other vertices, I find that's very nice.*". However, the constraints management in such programs was perceived as difficult. Participants found tracking constraints challenging because they are represented as a list in a panel different from the view. In consequence, it is easy to get into a model with several constraints that are hard to edit or that end up being overconstrained. P9 commented " *I try to avoid using constraints normally because when I've tried, I've ended up stuck, I have too many constraints and keep receiving the "Overconstrained" error message, and it is not easy*

to fix it for me. I do not know how to know which constraint is the problem".

Some participants (n = 9) commented on the way of creating parametric models by defining object properties through parameters and variables in OpenSCAD. All agreed on the importance of generalizing model behaviors through the use of variables and avoiding the definition of object properties with hard-coded numbers. P6 said "*... everything is based on making the outer frame as a combination of parameters, so the parameters can change and then the model still works*". However, participants mentioned that defining everything in terms of variables can be exhausting, so sometimes, they use variables only when they anticipate that a certain property will need to change. Moreover, working out the mathematical expressions for creating parametric models is perceived as very challenging. P6 expressed "*I sometimes have a harder time doing the math (to define object's properties), using all the combinations of variables where they should be*".

5.2.4 Achieving digital precision. One primary need of some participants (n = 9) when printing is to achieve precision. They perceive that OpenSCAD allows them to achieve accuracy in easier ways by explicitly expressing the sizes and dimensions of every placed part. P1 commented "*from Blender I could not make things precise (...) I use CAD programs for electronic devices, from there I get measurements. In OpenSCAD, it is very easy to be precise with this. I only need to make a box of this size or that size ...*". P12 acknowledges that some direct manipulation programs allow one to express precise measures but it is not as clear and flexible.

Paradoxically, while users feel a sense of precision by being able to describe position and sizes explicitly, eight participants complained about the lack of means to check dimensions in the OpenSCAD view. P19 mentioned "*I had a really tough time taking measurements and showing measurements visually as part of the model.*" As mentioned, spatial transformation involves several nested operations and variables, so verification is important. P14 said "*I wrote all these formulas, but did the resulting piece have the correct size and location?*". To deal with this, participants use `echo` operations to print the expression results in a console (P4 "*I could put some echos to verify those formulas, but It would be much better measure on the screen.*") or visually inspect parts on the view by emulating a ruler (P11 "*... I put a cube of the right size next to it and just visually inspect if they are the same height*")

5.2.5 Designing curves and organic shapes. Nine participants said that, in general, they feel that OpenSCAD is not a friendly application for creating nonstructured curves and organic shapes that are difficult to define using mathematical expressions. For instance, creating smooth corners was reported as *difficult* and *painfull*. Participants commented: P4 "*(in OpenSCAD) making rounded edges is a pain*"; P13 "*It is just easier for the printer to 3D print something that's rounded (...) But designing it, that is really super hard to do in OpenSCAD*". Even using prebuilt OpenSCAD functions for this purpose is hard, such as `Minkowski` function. P16 and P1 mentioned "*(the most difficult in OpenSCAD is) figuring out how to do rounded edges and fillets and chamfers without using Minkowski because it is too time-consuming to render*" and "*... making smooth corners is possible, I use the Minkowski tool but the dimensions changes, it is inconvenient.*". However, three participants said that when the shape or curve can be defined mathematically, OpenSCAD is convenient for designing it. P10 discussed a project where he defined a Bézier curve that changed parametrically. Although the mathematical expression was hard to create, it would work better to have a parametric design because the curve can be expressed in code.

On the other hand, this task seems significantly easier in B-rep direct manipulation programs. P6 mentioned "*in Fusion360 if I want to have a squared box and I want to have a nice rounded corner, I just can click on both faces and I have a nice rounded edge*". This difference in difficulty between programming-based and direct manipulation applications in this seemingly simple task creates frustration for the participants. P4 expressed "*What makes it hard is that you can not point to a specific edge or corner and make it round. I don't like that*"

5.2.6 Dealing with CSG. Some participants (n = 7) mentioned limitations inherent in the CSG representation. First, four participants said it is hard to verify the result when using the `difference` and `intersect` operations that remove volume. The removed parts are not visible, and verifying the correctness of the operations seems problematic. P4 commented "*When you do a subtraction, it is hard to figure out if you are doing it right, if the subtracted piece is in the right spot, and if it has the right shape*". One mechanism to deal with this difficulty is using OpenSCAD *modifiers* (see section 2.4). However, modifiers require a trial-and-error approach, which is still time-consuming. P16 expressed "*(I use modifiers) with the invisible things, the things I'm subtracting. I'd said it's an easy way to figure it out, but it always takes some time*".

Second, three participants expressed frustration that they were unable to individualize points or faces from the volumes as it is possible with B-rep. For instance, P4 mentioned it was hard not to be able to point a corner and round it from the view. A similar problem occurs when, for instance, the model requires two cubes to touch in one face. By placing one box at the end of the other, there is no guarantee that the objects are overlapping. CSG does not represent faces or vertices information but abstract definitions. Hence, users can only define two boxes with coincident faces by correctly defining their positions and sizes. Thus, it seems to be common practice to add small offsets to ensure overlapping as mentioned P13 "*You can't have coincident faces; they have to go past each other. Every time you put one thing on top of another, you have to add those overlaps*".

5.2.7 Versioning and collaborative work. Few participants (n = 3) highly valued the flexibility of code to use repositories such as GitHub for versioning and collaborative work. Although it is possible with direct manipulation programs such as FreeCAD, it is not that convenient, as expressed by P3 " *I especially like how well OpenSCAD checks in the GitHub repository ... this is a very collaborative project, and you can check in Fusion360 in GitHub but it just does not work. The repository becomes huge very fast. But OpenSCAD is text-based, making it very well suited to the collaborative environments we are already using for the source code.*"

5.2.8 Specifics of programming-based CAD. We discussed with the participants the advantages and limitations of programming-based CAD applications in general and specifically in OpenSCAD.

Working with several files/parts. It is normal to break down the model into parts when having complex models. In addition to facilitating the design process, it is an alternative when the part has to be printed in several parts. This scenario presented some difficulties for the participants ($n = 9$). OpenSCAD does not have any option to define parts in the design, so the participants can export parts individually. One solution mentioned by the participants was to create the design and later apply a `difference` and `intersection` with a cube. First, the participant would apply the `difference` and remove half of the model to export it to an STL file for printing. Later, the participant would use the same cube and apply an `intersection`, removing the second half to export and print. Once both parts are printed, they will be assembled. The result would not make parts easy to connect. Therefore, other participants used some libraries to split the model into parts to better assemble them.

In other cases, applying `difference` and `intersection` is not enough when the model is complex. Thus, participants create a file for each part. This allows them to isolate each part and focus their efforts without being distracted by the other parts. Nevertheless, the process of validating all the parts together is very difficult. For instance, P5 shared with us a model of a GoPro camera gimbal with many parts that form an articulated arm. He imported all the parts into the same project when he wanted to verify how it would look together. However, when modeling individually, all pieces are placed in the center. Thus, when importing all parts together, it is not easy to place them in the correct position and orientation only to verify the result.

Dealing with code. Eighteen participants discussed issues and experiences managing code in OpenSCAD. In general, participants try to keep good coding practices to make their models easy to understand. They mentioned the importance of commenting on code, avoiding very long modules, organizing the code, and using expressive and telling names for variables and modules. However, they acknowledge that using expressive names for all variables is impossible. Moreover, using the expressiveness of the language is not always useful. For instance, P15 explained his model, which was extensively documented in his mother tongue, Slovakian.

Some participants ($n = 7$) discussed the challenges they face when dealing with code. According to them, problems include difficulties in easily finding parts in the code, keeping track of variables on complex models, and refactoring code. Moreover, we observed another challenge in the hands-on exercise. When participants were looking for a code statement of a particular part, they would analyze and perform tests on the code that was not being evaluated. For example, elements created inside a conditional structure that was not evaluated. OpenSCAD does not warn users about this situation, and they realize this after spending some time analyzing the code.

Some participants ($n = 6$) stressed that OpenSCAD code editor is basic and lacks more advanced code editor features, as commented by P6 "*OpenSCAD really lacks richness in helpers how to write code, there is no autocompletion and that kind of stuff*".

Code-view navigation. In programming-based CAD, the description of the model (*i.e.* code editor) is separated from the model visualization (*i.e.* viewer). Thus, users must constantly switch between the code where they edit the model and the view where they validate the result of the modifications. As a result, navigating the model and making edits can be difficult [86]. Some participants opted to use Visual Studio Code (VS Code) [56] instead of using OpenSCAD code editor. They stated that VS Code allows for the installation of OpenSCAD plugins that streamline the coding process. The participants used VS Code to modify their code files and accessed a separate window within OpenSCAD to review the output. We observed the behavior of the participants in the hands-on exercise and discussed with them the challenges related to these tasks.

We identified a three-step search pattern when looking for code statements that create specific parts in the view. First, they would try to identify the block of code where the target part could be defined. Then, they would study the code to confirm the selected code statement logically. Finally, they would seek a visual confirmation in the view.

Participants had five strategies for trying to locate the code statement based on the view: rely on their memory, link the part to a variable and search the variable, guess how the part should be created and look for the pattern, follow the comments, and using OpenSCAD search feature (see section 2.4). As participants worked with their own models, some participants ($n = 12$) tried to remember how the model was built and relate it to their normal way of structuring the code. For instance, when P7 tried to find the code statement of a part, they said "*I know how the hex array (main frame of the model) is organized in the first play, so I would go here (scrolls the code until finding the hex_array module), ok here it is*". The second strategy was to link the target part with the variables and use a text search feature, as P17 commented "*I would assume that it's related to this variable*" before searching for the occurrences of the variable. This strategy did not work well when the model repeatedly used the variable the participant picked to locate the target. The large number of occurrences made it difficult to study all of them to decide which code statement was correct. Moreover, OpenSCAD code editor features are basic and do not provide visual cues to help developers understand the code (*e.g.* highlight calls in the scroll bar of a selected variable or jump to the definition of a selected module by clicking on the module call). Participants who use VS Code could easily follow the places in the code where a variable was used. In some occasions, participants using VS Code made mistakes by editing the code of a file different from the file OpenSCAD was rendering. They edited the code and did not see any change in the view until they realized the problem after some time. The third strategy was to try to think about how the selected part should have been created in the code and look for that pattern in the text editor. For example, the target element for P6 and P2 was a hole, so they started looking for a `difference` code statement. In the fourth strategy, when the model was well documented, participants used comments to understand the structure of the code and find the correct statement. Finally, participants could locate the code statement with the OpenSCAD search feature. Only three participants knew about these features and mentioned not using them normally. When seeking the code statement, participants always read the code to understand it and confirm that it was the target statement.

After the participants thought they had located the target code statement, they normally sought visual confirmation. The strategies used to confirm were removing the code statements and verifying

missing objects, changing the parameter values of the code statement and verifying changes in the object's properties, using a `color` operations to highlight the object, and using OpenSCAD modifiers as a debugging task. We could identify some challenges when performing these strategies. Removing code statements can break the syntax of the code. For example, when a `translate` statement is written without opening and closing brackets, the transformation is applied only to the next code statement. The system will report an error if the code statement is temporarily removed and the next statement is not an object (*e.g.* variable definition). Also, using `color` operation does not work if the object is already inside a `color` scope. The system will override the statements, prioritizing the statement placed higher in the tree of operations. Participants frequently use modifiers for visual inspection but they often forget the correct syntax and characters to use. Moreover, modifiers do not work when they are used on 2D elements or code statements that are not being used (*e.g.* not evaluated conditional). In any case, all participants used a trial-and-error technique and required reading and understanding of the code.

Some participants ($n = 8$) mentioned some challenges they identified after the hands-on exercise. They acknowledge that reading and understanding the code is difficult. It becomes more challenging in complex models with long scripts, highly decoupled with several module calls, and with several parameters. They also mentioned that finding code based on the view is a repetitive, hard, and mentally demanding task.

Other programming-based CAD challenges. Eight participants mentioned some difficulties they identified when designing with OpenSCAD. We observed that participants are often surprised by the changes performed on the models because it is impossible to anticipate the edit's result confidently. Programming-based applications do not provide a transition between two states of the code. Consequently, understanding the impact of the changes made to the code is not always easy or obvious, as commented by P2 "*One of the difficulties is that you don't have immediate feedback when you change something as a result of the screen; there are always delays.*". Other challenges were related to difficulties in managing text elements in the view and confusion with OpenSCAD units.

5.2.9 How to improve OpenSCAD. Participants ($n = 14$) shared ideas about features that they considered would help their modeling experience. One key idea was the capability to easily identify parts of the code based on the view and vice versa. Participants thought that there could be features to help with this task. In particular, they would like to have a visual cue that helps them locate the code based on interactions with the view. P13 commented "*If I could point at something in OpenSCAD and tell me where this is (in the code editor), that would be a huge help. Especially if you have complex geometry, it's just super hard to figure out just where you are.*".

Participants also commented on how to facilitate the task of applying spatial transformations. First, they mentioned the need to be able to measure distances in the view. They also commented that it would be very convenient to be able to extract spatial coordinates of objects directly from the view. P14 commented that "*There are cases where I want to know all those coordinates, but OpenSCAD doesn't give that to you. I would like it to tell me what the bounding boxes of my model are. It doesn't have to show me on the screen, but at least when I render it in the output area, it would tell me the bounding box and the center of mass*". Moreover, seven participants indicated that they would like the system to assist them in retrieving the spatial coordinates in terms of the existing variables for parametric models by interacting with the view. P3 said "*(I would like) if I could click on this point (in the view) and OpenSCAD would automatically create an expression using the variables to describe that point (…)not hardcoded numbers because it is not useful anymore that way*". Further, in addition to working out the expressions, they would like the system to allow the creation of constraints directly from the view. P5 commented "*So being able to simply point at an object to say, I want to attach this face of this item to that point there and for it to be able to calculate that distance would be such a timesaver*".

Finally, participants mentioned the need to facilitate some recurrent tasks for 3D printing. Specifically, they mentioned that they would like to have more elaborate libraries to perform usual actions, such as creating chamfers.

5.2.10 Designing from scratch or re-using models. We discussed re-using models from websites such as Thingiverse, Printables, or Github with the participants. Half of the participants ($n = 10$) expressed that it is normal to check pre-existing models from websites before starting to design one from scratch. However, the reasons for that differ according to several aspects. For instance, P8 said that he usually starts by searching for models in Google when it is something complicated. This is probably because P8 has a short experience with OpenSCAD and 3D printing. Nine participants mentioned that if they needed to print something very popular, they would definitely go to check other people's solutions first. P20 said "*if it's something that's really popular that I absolutely know that there would be models for (…) I'll just go get one rather than design yet another one*".

Nevertheless, opinions on preferences between re-using models (and what using them for) or designing from scratch are divided.

Designing from scratch. Six participants prefer to design from scratch because they have specific needs and look for very customized models. P4 commented "*I can design to fit my own setup …for instance, my screwdriver holder, all the holes are a different size to fit my particular set of screwdrivers so that they fit snug…*". Other participants ($n = 5$) expressed their satisfaction in challenging themselves to build models of good quality on their own, as mentioned by P11 "*I just like the process of designing and printing things and be ready to say, I've done this by my own.*". Furthermore, four participants said that starting from scratch is easier, faster, and has better quality results.

Re-using pre-existing models. We discussed with participants their thoughts about using pre-existing models. We asked them if they use them, their motivations, limitations, and the model format they prefer to look for.

Most of the participants ($n = 16$) reported that pre-existing models do not meet their needs. Even when several models exist, they usually do not exactly fit participants' needs. And even if the model is parametric, available parameters rarely cover the modification participants want. Moreover, some models are too complex, with several parameters adding significant complexity to the printing process, so they are discouraged from using them. In

addition, participants perceive several difficulties with the available model-storing websites. To start, six participants commented that searching on these websites is challenging. The naming system is deficient, so it is difficult to find useful models. Moreover, dealing with licenses can be problematic. Some participants use 3D printing for commercial applications, so using public models is not ideal for them. Participants also commented on the *Customizer* application from Thingiverse. Although they find it useful, they also mentioned that it is very limited because authors can only upload models with one file, without the possibility of making references to other files or libraries.

However, the participants also acknowledge the potential of such websites. Six participants mentioned that pre-existing models can be time savers. They said that for printing popular objects, and for people with little experience in design, it is a good alternative. Some participants ($n = 8$) used pre-existing models as inspiration. These websites present alternatives for participant's projects in progress where they collect ideas. Some of them mentioned being surprised to see things they never thought possible to do with 3D printing.

Although participants did not use pre-existing models to print them directly, some have incorporated or edited pre-existing designs for their projects. Fourteen participants commented on difficulties in editing STL files in direct manipulation and programming-based applications. All of them agreed that finding manifold and printable geometries is rare. In most cases, they needed to fix broken geometries before being able to use them, which was reported to be "*difficult*" and "*painful*". Participants preferred to have a pre-existing model *with code* instead. The offer of models with source code is more limited and quality varies significantly. Often, available models are not coded following good practices of programming, such as adding documentation, so reading and understanding the code is very hard. Some participants mentioned the importance of good-quality code if it is meant to be shared as mentioned P5 "*One thing is designing for yourself, and other designing for sharing*" This is not always the case and depends on the author.

Sharing models. Some participants ($n = 8$) had profiles on websites such as Thingiverse and Printables, where they shared their designs. They manifested that they enjoyed sharing their models, contributing, and seeing other people using and commenting on their models. Three participants said that although they liked sharing, they did not want to spend more time adjusting their models. Unfortunately, the models need to be adjusted to share parametric models in applications like Thingiverse Customizer. For instance, Customizer only accepts one-file models. P5 commented that he would like to share more of his models, but he often re-uses his own libraries that cannot be uploaded to the website. P6, on the other hand, said that he would like to customize the parameters to provide a better experience using sliders instead of input boxes, but would not spend time learning how to do it. Also, some participants mentioned that websites are not controlled, and they had seen users taking models from authors and selling them on other websites.

5.3 Fabrication

Some of the questions in the interview were intended to identify challenges in the fabrication process. We discuss our findings related to motivations and challenges.

5.3.1 Motivations for 3D printing. Participants use 3D printing for prototyping, for work, for repairing other objects, and as a hobby. They discussed what they liked about 3D printing. Five participants said that they were interested in technology and applications so it was almost natural for them to try 3D printing. Other participants mentioned that they have an "*intellectual satisfaction*" when they fabricate complex customizable things successfully. In addition, seven participants mentioned enjoying the fabrication process and especially having the physical result.

5.3.2 Design for printing. Three participants mentioned that designing and designing for printing are two different ideas. As explained by P11 "*Surely you can design everything in CAD, but having prints with many overhangs or supports is impossible. You need to design your models with printing in mind*".They also commented that it is easy to find very complex models that, in practice, are not printable. When rendering a model in OpenSCAD, users can use a rapid preview or a more realistic rendering option. P3 commented that these differences add uncertainty to the design process because the way both rendering options process the information can lead the designer to errors.

5.3.3 Printing challenges. Participants ($n = 15$) discussed different problems encountered with 3D printing. Very often, they expressed that tolerances and clearances are factors that introduce high uncertainty on the success of the printed object. 3D printing does not precisely reproduce the dimensions and sizes in the digital design. P10 commented " *I usually have to print multiple times to get the clearances correct, especially if there are moving parts. It usually takes several times to get the tolerances right*".

Another challenge is the difficulty of taking into account the material property in the design process. Different materials have different behaviors that are difficult to include in the design because OpenSCAD does not support this information. For instance, how to design taking into account weak points in the design. P14 commented that "*most difficult in designing for 3D printing is making sure you're accounting for the anisotropic strength properties of the material ... anything you print is weaker along layer lines, for instance*".

Nine participants considered that knowing the printer they use is a key factor in minimizing uncertainty. They mentioned that hardware in 3D printing is sensitive to failure and requires good skills to set up for success, as commented by P19 " *if it's your own printer, you have a better sense of what to expect and you have agency over the state of your machine*".

Other limitations were discussed. When the model is large, there is less flexibility to iterate due to the amount of material required. Moreover, participants mentioned that sometimes, putting models in a correct orientation for better printing is not trivial. Furthermore, some participants ($n = 4$) mentioned that printing with supports can be tedious, so they prefer to avoid it when possible.

5.3.4 Testing. Testing was important to avoid false printings and waste of material. Participants ($n = 4$) acknowledge that their only validation strategy was test prints. P13 and P16 said they would print layers to verify clearances before printing the entire piece. Although test prints are time-consuming, they were unable to find

Table 4: Structure of theme *Printing*. Color intensity is proportional to the number of interviews coded with codes of the theme and subthemes.

Theme	Subtheme		
Printing (n = 19)	Motivations for 3D printing (n = 16)	It started with a family member or a friend (n = 3)	
		What do I like about it? (n = 10)	I like creating customized things (n = 7)
			I feel intellectually satisfied with achieving things (n = 3)
			I like tech (n = 2)
		What do I use it for? (n = 8)	Hobby (n = 3)
			I like to repair things (n = 3)
			I use it for work (n = 3)
			Prototyping (n = 3)
	Design for printing (n = 6)		
	Iterating the design after printing (n = 2)		
	Printing challenges (n = 15)	Clearances introduce uncertainty, making objects fit (n = 6)	
		Material properties (n = 3)	
		Printer limitations (n = 9)	
		Limitations to iterate (n = 1)	
		Other challenges (n = 8)	
	Testing (n = 4)		

another testing mechanism to anticipate what the result of printing would look like.

6 DISCUSSION

We are interested in understanding the motivations of programming-based CAD users. Moreover, we aim to study their challenges and limitations to discuss opportunities for HCI research.

6.1 Programming-based CAD users in 3D printing

Findings in programming-based CAD user preferences (section 5.1) allowed us to identify two types of motivations when users 3D print. The utilitarian and the enthusiastic. Users with utilitarian motivation use 3D printing to solve problems pragmatically. They are more flexible in design decisions when there are limitations or a lack of motivation. For instance, participants expressed not liking re-using pre-existing models, but due to the time limitations, for example, P19 found pre-existing models a very convenient solution. He expressed "*As a father with a full-time job, it's difficult to sit down and develop a model (…) So oftentimes, I'll look for existing models, and if they work, I go with them because it's often easier*". Similar situations occurred when printing common objects that participants felt they could find on the Internet.

In contrast, users with enthusiastic motivation invest time and energy in achieving neat and highly customized objects. Participants used very creative methods and could iterate several times to achieve a satisfactory result. P4 shared how a small model for a screwdriver hole changed over 5 iterations to achieve a satisfactory result. P14 had experimented with different creative solutions to capture the outline of curved objects he wanted to repair. When having an enthusiastic motivation, users are more likely to design from scratch rather than use pre-existing STL models with questionable quality and printability, or coded models that would require time and effort to understand. Moreover, some participants were moved by the feeling of pride of "*intellectual satisfaction*" when achieving a result. As commented by P16 "*I like OpenSCAD because it is very functional, not procedural. That's an intellectual like … functional languages (like OpenSCAD) are intellectually satisfying for me.*"

Furthermore, we identified that the fabrication process involved stages that users can or cannot enjoy. P7 mentioned that " *(3D printing) It's actually three hobbies in one, which is why a lot of people find it very overwhelming (…) There's the modeling, setting and improving the 3D printer, and the actual making of the 3D prints*". Indeed, we could observe different enjoyment from the participants at the different stages of the process. For instance, P14 described a project that took him weeks to create a highly customizable model for pliers. On the other hand, P16 explained a workflow he developed to create objects with multiple colors on a single-color printer by playing with the printer settings and the design. Moreover, P8 said that, although he enjoyed the design process, what he liked was the result. Consequently, he rushed other parts despite the errors he could make, such as taking measurements with care. "*I think like to go to the good part, I want to get to the fun, having the thing in my hand.*" Previous studies [32] reported that novices were easily frustrated designing objects for 3D printing, probably because their motivations were to get the objects, not to design them.

6.2 Programming-based CAD design challenges

The interviews reveal challenges that programming-based CAD users face (section 5.2), representing a research opportunity for the HCI community.

Programming-based CAD inputs are coded instructions that the machine uses to create a visual representation where the user can verify the result and perform edits in the code if required. Therefore, the user verifies in one space and edits in another in an intensive and iterative exercise. Consequently, users are forced to understand the connections between the code and the rendered model, but current tools barely try to help users in this task.

Navigability. Our hands-on exercise revealed that users use several strategies to identify code statements responsible for creating specific parts based on visual inspection with considerable difficulty. Most of the participants went through the code to understand it, and in every case, they needed to make edits to seek visual confirmation. Programming-based applications could facilitate this task by using the implicit connection between the code and the view as some direct manipulation applications do, connecting the view and the history tree. Applications such as FreeCAD [80] and Fusion360 [34] allow users to click on specific parts on the view while the corresponding element in the history tree is highlighted. OpenSCAD provides a backward search (section 2.4) to place a cursor on a code statement based on a selected part in the view. However, it does not provide visual cues to confirm that the selected part is aimed, and users need to modify the code to seek visual confirmation. IceSL [49] provides better visual cues to highlight code statements responsible for creating parts in the view but does not discriminate between statements involved, so it is impossible to navigate the CSG structure. Gonzalez *et al.* [27] propose a navigation system that helps isolate specific parts with the corresponding code statements with visual cues in OpenSCAD. Such systems facilitate the understanding of the structure of the code and how it is related to the view without the need to study or modify the code. However, this interaction technique has limitations related to ambiguity when selecting parts that are visually the same but different in code. For instance, a cube is modified by a `translate` or `scale` statement. The visual response is the same when selecting any of these two statements, although each performs different tasks. A navigation system could differentiate these cases, adding visual cues to understand the nature of the statement. In the previous example, the preview could show an animation of the cube being created and moved or scaled, with the transformation statement indicating the dynamic effect of the code.

Understanding spatial coordinate systems in the code. Understanding 3D spaces on a 2D screen is a difficult task reported in direct manipulation [32]. Our interview revealed that this problem can be more difficult in programming-based CAD where the editing space is disconnected from the view, and there is no visual help to relate them. In other words, users have to mentally imagine the behavior in the view that a spatial transformation will have when stated in code. Moreover, CSG structures create nested scopes where the coordinate system is relative to the aggregated effect of spatial transformations performed previously. Participants mentioned the need for trial and error strategies to apply a spatial transformation successfully. The task becomes more complex when using rotations, which are generally more difficult to understand. Some participants tried to avoid nested transformations due to difficulty understanding them. To alleviate these challenges, programming-based CAD applications could incorporate features to enhance users' comprehension of spatial properties during coding. For instance, some direct manipulation applications integrate *"manipulators"* [36], such as row-shaped widgets positioned at the center of objects, indicating the relative 'x', 'y', and 'z' axes, as seen in Unity [81]. In programing-based CAD, Gonzalez *et al.* [27] system allows the user to select a specific part and places these manipulators in the center of the relative coordinate system of a selected object to visualize it. This helps to understand the relative system coordinate in the view, but it does not connect it to the meaning in the code. The correspondence between both spaces can be more explicit through visual cues. For instance, axis widgets have different colors to distinguish the translation axis, which the code editor could use in the space of the transformation parameters to make this correspondence explicit. Moreover, by clicking on the widgets, the application could open a text dialog to edit the corresponding parameter directly in the view, as is possible in some direct manipulation applications [34].

Defining geometric properties based on pre-existing information. In CAD design, the geometric properties of an object are closely related to those of other objects. For example, users may want to place a cube on top of another. Direct manipulation applications facilitate user interaction through visual aids such as rulers and volume highlighting during overlap, the use of snap effects that guide positioning during drag-and-drop actions, or the use of constraints [23, 34]. These applications facilitate the re-use of additional information from the model within the model. However, programming-based CAD makes this task more challenging. Applications such as OpenSCAD or JSCAD often limit interactions within the view, preventing users from selecting specific parts or re-using information, for example, the position of an object. Applications could use visual elements as rulers that allow for measurement or even retrieve raw positions, orientations, or sizes from objects directly from the view when clicking on parts of the model's visual representation. Implementing bidirectional programming [55] behaviors could support this task as has been done in SVG [28] and CAD previous work [27, 40].

In general, programming-based CAD could benefit from allowing a more enriched interaction in the derived information rendered in the view while coherently connecting it with the code and supporting the edit based on this information. Moving towards bidirectional programming [28] or live programming [85] paradigms could bridge the gap existing between both spaces, code and view, which is one significant difficulty in programming [60].

6.3 3D printing challenges.

The interviews revealed problems in the process related to the disconnection between the CAD model and the target environment as described in section 5.3.

Fit between the design object and other physical objects. Users cannot consider contextual limitations, resulting in longer processes and creativity limitations [76]. P9 explained that the lack of context resulted in more iterations fabricating a case for an emergency button. Bridging the digital design and the physical environment can facilitate the design process. One approach is incorporating digital references of the physical environment into the digital design.

Some participants upload STL replicas of objects into OpenSCAD and FreeCAD to have a reference of the object and design around it. Websites such as Thingiverse [83] or MyMiniFactory [58] offer models, mostly in STL format, that users can use as references when designing, although participants and previous work have reported some problems with the search engines and meshes quality [18, 50]. Another possibility is to bring the model into the environment. DesignAR [66], for instance, allows designers to work in augmented reality environments and place models in physical environments using direct manipulation. Programming-based CAD applications could explore having virtual or augmented reality previews for a realistic preview of objects. Moreover, these environments could also facilitate code interaction and understanding, as previously explored in other fields [31, 41, 68, 69]. Furthermore, expanding interaction in programming-based CAD can leverage bidirectional programming [27, 28, 55].

Include physical measurements. Capturing data from physical objects has been also reported as challenging [43, 53, 65]. Participants reported difficulties in measuring curved and organic shapes using programming-based CAD. Research could explore sensing devices to capture and transfer information to CAD applications. Some participants use photogrammetry and scanners to capture the outline of curves and reproduce them digitally, but these solutions were found to be time-consuming, complex, and imprecise. Additionally, retrieved information as a point cloud is difficult to parameterize. An alternative solution is to use Bezier curves, where a few control points mathematically define a contour. Solutions like ShArc [70] can capture data from these control points with less effort, creating a suitable solution to replicate organic shapes parametrically. The use of augmented reality could also help to capture control points to create Bezier curves.

Other challenges in 3D printing. Users go through different disconnected stages when 3D printing. CAD applications often limit functionalities to the design, ignoring limitations concerning the printing process. For example, participants used to apply spatial transformations to their models to locate them in an optimal position and orientation for 3D printing. However, if any edit was required, they removed these transformations to see the model in a more familiar position and orientation. Similarly, when participants needed to split a model into parts. They first finish the model and then apply a `difference` and `intersection` with conditionals to save two different STL files.

CAD application could facilitate the design by including information related to the printing process. For example, the same model would not print the same on different printers or with different materials. One of the main factors of uncertainty expressed by the participants was the tolerances and clearances. After printing and detecting problems, they would have to go to the CAD model, adjust it, export it, and print it again. CAD software could inform users of possible problems related to the design complexity (*e.g.* need for supports), printer tolerances, printer capabilities (*e.g.* possible angles of printing), or materials properties when printing.

7 LIMITATIONS

The hands-on exercise was a short observation task rather than a controlled user study. Findings related to it may be limited, missing other challenges that users face in the design process. Moreover, having experience in direct manipulation programs was not an exclusion criterion. Consequently, some of the participants had no previous experience in such software, and their answers related to these applications were not based on a reasonable experience and understanding of the direct manipulation paradigm. Finally, our interview only included OpenSCAD users. Although most programming-based CAD applications also use a CSG representation, each tool provides different features, and not all of our findings may be generalizable. Further, a few programming-based tools that use B-rep representation, such as CadQuery, may provide a different user experience and challenges than the ones reported in our findings.

8 CONCLUSION

We interviewed twenty users of the most popular programming-based CAD tool, OpenSCAD, to investigate their motivations and challenges in the design of 3D objects and the 3D printing process. During these interviews, we included hands-on experience to observe behaviors and difficulties when navigating the model. With the information collected, we performed a reflexive thematic analysis in an iterative process, developing main themes related to the user's profile, design experience, and printing experience. Our findings reveal that users are motivated to use programming-based CAD tools thanks to their parametric capability, the possibility of using mathematical expressions, and the precision for 3D printing. Moreover, it reveals several challenges in connecting the code with the view, understanding and performing spatial transformations, measuring and designing organic and curve shapes, validating dimensions in the view, and re-using pre-existing models. Programming-based CAD could facilitate some of these tasks by enabling the information that the system stores and effectively communicating it to the user. Last, our findings also reveal difficulties in the 3D printing process, such as handling uncertainty introduced by printers and material properties, identifying code locations to perform correction based on physical inspection, and validation before printing.

REFERENCES

[1] Robert Aish. 2012. DesignScript: Origins, Explanation, Illustration. In *Computational Design Modelling*, Christoph Gengnagel, Axel Kilian, Norbert Palz, and Fabian Scheurer (Eds.). Springer, Berlin, Heidelberg, 1–8. https://doi.org/10.1007/978-3-642-23435-4_1

[2] Celena Alcock, Nathaniel Hudson, and Parmit K. Chilana. 2016. Barriers to Using, Customizing, and Printing 3D Designs on Thingiverse. In *Proceedings of the 19th International Conference on Supporting Group Work (GROUP '16)*. Association for Computing Machinery, New York, NY, USA, 195–199. https://doi.org/10.1145/2957276.2957301

[3] Amabilis. 2022. Amabilis Software. http://amabilis.com/

[4] D. C. Anderson and T. C. Chang. 1990. Geometric reasoning in feature-based design and process planning. *Computers & Graphics* 14, 2 (Jan. 1990), 225–235. https://doi.org/10.1016/0097-8493(90)90034-U

[5] Prusa Research a.s. 2023. Base de datos de modelos 3D. https://www.printables.com

[6] Daniel Ashbrook, Shitao Stan Guo, and Alan Lambie. 2016. Towards Augmented Fabrication: Combining Fabricated and Existing Objects. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*. Association for Computing Machinery, New York, NY, USA, 1510–1518.

https://doi.org/10.1145/2851581.2892509
[7] Autodesk. 2020. Tinkercad | Create 3D digital designs with online CAD. https://www.tinkercad.com/
[8] Giles Bathgate. 2023. RapCAD. https://gilesbathgate.com/category/rapcad/
[9] Srinjita Bhaduri, Quentin L Biddy, Jeffrey Bush, Abhijit Suresh, and Tamara Sumner. 2021. 3DnST: A Framework Towards Understanding Children's Interaction with Tinkercad and Enhancing Spatial Thinking Skills. In *Proceedings of the 20th Annual ACM Interaction Design and Children Conference (IDC '21)*. Association for Computing Machinery, New York, NY, USA, 257–267. https://doi.org/10.1145/3459990.3460717
[10] Richard E. Boyatzis. 1998. *Transforming qualitative information: Thematic analysis and code development*. Sage Publications, Inc, Thousand Oaks, CA, US. Pages: xvi, 184.
[11] Virginia Braun and Victoria Clarke. 2013. *Successful qualitative research: a practical guide for beginners*. SAGE, Los Angeles. OCLC: ocn811733656.
[12] Virginia Braun and Victoria Clarke. 2021. Can I use TA? Should I use TA? Should I not use TA? Comparing reflexive thematic analysis and other pattern-based qualitative analytic approaches. *Counselling and Psychotherapy Research* 21, 1 (2021), 37–47. https://doi.org/10.1002/capr.12360 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/capr.12360.
[13] Virginia Braun, Victoria Clarke, and V Hayfield. 2019. Answers to frequently asked questions about thematic analysis April 2019.pdf. https://cdn.auckland.ac.nz/assets/psych/about/our-research/documents/Answers%20to%20frequently%20asked%20questions%20about%20thematic%20analysis%20April%202019.pdf
[14] Mark Brunelli. 2022. Parametric vs. Direct Modeling: Which Side Are You On? https://www.ptc.com/en/blogs/cad/parametric-vs-direct-modeling-which-side-are-you-on?
[15] David Byrne. 2022. A worked example of Braun and Clarke's approach to reflexive thematic analysis. *Quality & Quantity* 56, 3 (June 2022), 1391–1412. https://doi.org/10.1007/s11135-021-01182-y
[16] CadQuery. 2023. CadQuery. https://github.com/CadQuery/cadquery originaldate: 2018-10-28T17:57:18Z.
[17] DEVCOM Analysis Center. 2023. BRL-CAD: Open Source Solid Modeling. https://brlcad.org/
[18] Erik Champion and Hafizur Rahaman. 2020. Survey of 3D digital heritage repositories and platforms. *Virtual Archaeology Review* 11, 23 (July 2020), 1–15. https://doi.org/10.4995/var.2020.13226 Number: 23.
[19] 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)*. 1978–1987. https://doi.org/10.1109/EDUCON.2018.8363478 ISSN: 2165-9567.
[20] Scott Davidson. 2023. Grasshopper. https://www.grasshopper3d.com/
[21] James D. Foley, Foley Dan Van, Andries Van Dam, Steven K. Feiner, and John F. Hughes. 1996. *Computer Graphics: Principles and Practice*. Addison-Wesley Professional.
[22] Blender Foundation. 2023. blender.org - Home of the Blender project - Free and Open 3D Creation Software. https://www.blender.org/
[23] FreeCAD. 2023. Python scripting tutorial - FreeCAD Documentation. https://wiki.freecad.org/Python_scripting_tutorial/en
[24] David Frohlich. 1997. Direct Manipulation and Other Lessons. *Handbook of Human-Computer Interaction (2nd Ed.)* (Dec. 1997). https://doi.org/10.1016/B978-044481862-1.50087-X
[25] Ian Gibson, David Rosen, and Brent Stucker. 2015. *Additive manufacturing technologies: 3D printing, rapid prototyping and direct digital manufacturing* (second edition ed.). Springer, New York Heidelberg Dodrecht London.
[26] VERBI GmbH. 2023. Software de Análisis de Datos Cualitativos - MAXQDA. https://www.maxqda.com
[27] Johann Felipe Gonzalez, Danny Kieken, Thomas Pietrzak, Audrey Girouard, and Géry Casiez. 2023. Introducing Bidirectional Programming in Constructive Solid Geometry-Based CAD. In *Proceedings of the 2023 ACM Symposium on Spatial User Interaction (SUI '23)*. Association for Computing Machinery, Sydney, Australia, 1–12. https://doi.org/10.1145/3607822.3614521
[28] Brian Hempel, Justin Lubin, and Ravi Chugh. 2019. Sketch-n-Sketch: Output-Directed Programming for SVG. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. Association for Computing Machinery, New York, NY, USA, 281–292. https://doi.org/10.1145/3332165.3347925
[29] Monique M. Hennink, Bonnie N. Kaiser, and Vincent C. Marconi. 2017. Code Saturation Versus Meaning Saturation: How Many Interviews Are Enough? *Qualitative Health Research* 27, 4 (March 2017), 591–608. https://doi.org/10.1177/1049732316665344 Publisher: SAGE Publications Inc.
[30] Christoph M. Hoffmann. 1989. *Geometric and solid modeling: an introduction*. Morgan Kaufmann, San Mateo, Calif.
[31] Akihiro Hori, Masumi Kawakami, and Makoto Ichii. 2019. CodeHouse: VR Code Visualization Tool. In *2019 Working Conference on Software Visualization (VISSOFT)*. 83–87. https://doi.org/10.1109/VISSOFT.2019.00018
[32] Nathaniel Hudson, Celena Alcock, and Parmit K. Chilana. 2016. Understanding Newcomers to 3D Printing: Motivations, Workflows, and Barriers of Casual Makers. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 384–396. https://doi.org/10.1145/2858036.2858266
[33] Gina Häußge. 2023. OctoPrint.org. https://octoprint.org/
[34] Autodesk Inc. 2023. Fusion 360 | 3D CAD, CAM, CAE, & PCB Cloud-Based Software | Autodesk. https://www.autodesk.com/products/fusion-360/overview
[35] BlocksCAD Inc. 2023. BlocksCAD. https://www.blockscad3d.com/
[36] J. Jankowski and M. Hachet. 2015. Advances in Interaction with 3D Environments. *Computer Graphics Forum* 34, 1 (2015), 152–190. https://doi.org/10.1111/cgf.12466
[37] Chris Johnson. 2023. Computational Making with Twoville. *Journal of Computing Sciences in Colleges* 38, 8 (2023), 39–53.
[38] Petra Kastl, Oliver Krisch, and Ralf Romeike. 2017. 3D Printing as Medium for Motivation and Creativity in Computer Science Lessons. In *Informatics in Schools: Focus on Learning Programming (Lecture Notes in Computer Science)*, Valentina Dagienė and Arto Hellas (Eds.). Springer International Publishing, Cham, 27–36. https://doi.org/10.1007/978-3-319-71483-7_3
[39] Matt Keeter. 2022. libfive. https://libfive.com/
[40] Matthew Keeter. 2023. Antimony. https://www.mattkeeter.com/projects/antimony/3/
[41] Pooya Khaloo, Mehran Maghoumi, Eugene Taranta, David Bettner, and Joseph Laviola. 2017. Code Park: A New 3D Code Visualization Tool. In *2017 IEEE Working Conference on Software Visualization (VISSOFT)*. 43–53. https://doi.org/10.1109/VISSOFT.2017.10
[42] Hyun Suk Kim, Heedong Ko, and Kunwoo Lee. 1993. Incremental feature-based modeling. In *Proceedings on the second ACM symposium on Solid modeling and applications (SMA '93)*. Association for Computing Machinery, New York, NY, USA, 469–470. https://doi.org/10.1145/164360.164525
[43] Jeeeun Kim, Anhong Guo, Tom Yeh, Scott E. Hudson, and Jennifer Mankoff. 2017. Understanding Uncertainty in Measurement and Accommodating its Impact in 3D Modeling and Printing. In *Proceedings of the 2017 Conference on Designing Interactive Systems (DIS '17)*. Association for Computing Machinery, New York, NY, USA, 1067–1078. https://doi.org/10.1145/3064663.3064690
[44] Amy J. Ko, Brad A. Myers, and Htet Htet Aung. 2004. Six Learning Barriers in End-User Programming Systems. In *2004 IEEE Symposium on Visual Languages - Human Centric Computing*. Rome, Italy, 199–206. https://doi.org/10.1109/VLHCC.2004.47
[45] Siniša Kolarić, Halil Erhan, Robert Woodbury, and Bernhard E. Riecke. 2010. Comprehending parametric CAD models: an evaluation of two graphical user interfaces. *Nordic Conference on Human-Computer Interaction* (Oct. 2010), 707–710. https://doi.org/10.1145/1868914.1869010 MAG ID: 2002429313 S2ID: ead43517e8402c84be0e4aec844a3316cefd12f2.
[46] Stacey Kuznetsov and Eric Paulos. 2010. Rise of the expert amateur: DIY projects, communities, and cultures. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries (NordiCHI '10)*. Association for Computing Machinery, New York, NY, USA, 295–304. https://doi.org/10.1145/1868914.1868950
[47] Bum chul Kwon, Waqas Javed, Niklas Elmqvist, and Ji Soo Yi. 2011. Direct manipulation through surrogate objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, New York, NY, USA, 627–636. https://doi.org/10.1145/1978942.1979033
[48] J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (1977), 159–174. https://doi.org/10.2307/2529310 Publisher: [Wiley, International Biometric Society].
[49] Sylvain Lefebvre, Salim Perchy, Cédric Zanni, and Pierre Bedell. 2022. IceSL. https://icesl.loria.fr/
[50] Chen Liang, Anhong Guo, and Jeeeun Kim. 2022. CustomizAR: Facilitating Interactive Exploration and Measurement of Adaptive 3D Designs. In *Designing Interactive Systems Conference (DIS '22)*. Association for Computing Machinery, New York, NY, USA, 898–912. https://doi.org/10.1145/3532106.3533561
[51] Julia Longtin. 2023. ImplicitCad.org. https://www.implicitcad.org/
[52] Felipe Machado, Norberto Malpica, and Susana Borromeo. 2019. Parametric CAD modeling for open source scientific hardware: Comparing OpenSCAD and FreeCAD Python scripts. *PLOS ONE* 14, 12 (May 2019), e0225795. https://doi.org/10.1371/journal.pone.0225795 Publisher: Public Library of Science.
[53] Chandan Mahapatra, Jonas Kjeldmand Jensen, Michael McQuaid, and Daniel Ashbrook. 2019. Barriers to End-User Designers of Augmented Fabrication. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, Glasgow Scotland Uk, 1–15. https://doi.org/10.1145/3290605.3300613
[54] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (Nov. 2019), 72:1–72:23. https://doi.org/10.1145/3359174
[55] Michael J. McGuffin and Christopher P. Fuhrman. 2020. Categories and Completeness of Visual Programming and Direct Manipulation. In *Proceedings of the International Conference on Advanced Visual Interfaces (AVI '20)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/

3399715.3399821
[56] Microsoft. 2023. Visual Studio Code - Code Editing. Redefined. https://code.visualstudio.com/
[57] Catarina Mota. 2011. The rise of personal fabrication. In *Proceedings of the 8th ACM conference on Creativity and cognition (C&C '11)*. Association for Computing Machinery, New York, NY, USA, 279–288. https://doi.org/10.1145/2069618.2069665
[58] MyMiniFactory. 2022. MyMiniFactory | Discover STL files for 3D printing ideas and high-quality 3D printer models. https://www.myminifactory.com/
[59] Yuenyong Nilsiam and Joshua M. Pearce. 2017. Free and Open Source 3-D Model Customizer for Websites to Democratize Design with OpenSCAD. *Designs* 1, 1 (Sept. 2017), 5. https://doi.org/10.3390/designs1010005 Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.
[60] Donald Norman. 1986. Cognitive Engineering. In *User Centered System Design: New Perspectives on Human-Computer Interaction*. 31–61. https://doi.org/10.1201/b15703-3 Journal Abbreviation: User Centered System Design: New Perspectives on Human-Computer Interaction.
[61] Lora Oehlberg, Wesley Willett, and Wendy E. Mackay. 2015. Patterns of Physical Design Remixing in Online Maker Communities. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. Association for Computing Machinery, New York, NY, USA, 639–648. https://doi.org/10.1145/2702123.2702175
[62] Library of Congress. 2019. STL (STereoLithography) File Format Family. https://www.loc.gov/preservation/digital/formats/fdd/fdd000504.shtml
[63] OpenJSCAD.org. 2023. JSCAD - JavaScript CAD. https://openjscad.xyz/
[64] OpenSCAD. 2020. OpenSCAD. http://openscad.org
[65] Raf Ramakers, Danny Leen, Jeeeun Kim, Kris Luyten, Steven Houben, and Tom Veuskens. 2023. Measurement Patterns: User-Oriented Strategies for Dealing with Measurements and Dimensions in Making Processes. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. Association for Computing Machinery, New York, NY, USA, 1–17. https://doi.org/10.1145/3544548.3581157
[66] Patrick Reipschläger and Raimund Dachselt. 2019. DesignAR: Immersive 3D-Modeling Combining Augmented Reality with Interactive Displays. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces (ISS '19)*. Association for Computing Machinery, New York, NY, USA, 29–41. https://doi.org/10.1145/3343055.3359718
[67] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education* 13, 2 (June 2003), 137–172. https://doi.org/10.1076/csed.13.2.137.14200 Publisher: Routledge.
[68] Markus Schütz and Michael Wimmer. 2019. Live Coding of a VR Render Engine in VR. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 1150–1151. https://doi.org/10.1109/VR.2019.8797760 ISSN: 2642-5254.
[69] Víctor Stefano Segura Castillo, Leonel Merino, Geoffrey Hecht, and Alexandre Bergel. 2021. VR-Based User Interactions to Exploit Infinite Space in Programming Activities. In *2021 40th International Conference of the Chilean Computer Science Society (SCCC)*. 1–5. https://doi.org/10.1109/SCCC54552.2021.9650396 ISSN: 2691-0632.
[70] Fereshteh Shahmiri and Paul H. Dietz. 2020. ShArc: A Geometric Technique for Multi-Bend / Shape Sensing. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376269
[71] Irv Shapiro. 2023. MakeWithTech Blog. https://www.makewithtech.com/
[72] Samyukta Sherugar and Raluca Budiu. 2016. Direct Manipulation: Definition. https://www.nngroup.com/articles/direct-manipulation/
[73] Ben Shneiderman. 1982. The future of interactive systems and the emergence of direct manipulation. *Behaviour & Information Technology* 1, 3 (July 1982), 237–256. https://doi.org/10.1080/01449298208914450 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01449298208914450.
[74] Ben Shneiderman. 1983. Direct Manipulation: A Step Beyond Programming Languages. *Computer* 16, 8 (1983), 57–69. https://doi.org/10.1109/MC.1983.1654471 Conference Name: Computer.
[75] Ben Shneiderman. 1997. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces (IUI '97)*. Association for Computing Machinery, New York, NY, USA, 33–39. https://doi.org/10.1145/238218.238281
[76] Ben Shneiderman, Gerhard Fischer, Mary Czerwinski, Mitch Resnick, Brad Myers, Linda Candy, Ernest Edmonds, Mike Eisenberg, Elisa Giaccardi, Tom Hewett, Pamela Jennings, Bill Kules, Kumiyo Nakakoji, Jay Nunamaker, Randy Pausch, Ted Selker, Elisabeth Sylvan, and Michael Terry. 2006. Creativity Support Tools: Report From a U.S. National Science Foundation Sponsored Workshop. *International Journal of Human–Computer Interaction* 20, 2 (May 2006), 61–77. https://doi.org/10.1207/s15327590ijhc2002_1 Publisher: Taylor & Francis.
[77] Brendan M. Sleight. 2023. lasercut.scad. https://github.com/bmsleight/lasercut original-date: 2015-08-26T20:13:41Z.
[78] Joanna Smith and Jill Firth. 2011. Qualitative data analysis: the framework approach. *Nurse Researcher* 18, 2 (Jan. 2011), 52–62. https://doi.org/10.7748/nr2011.01.18.2.52.c8284
[79] David Stutz. 2018. A Formal Definition of Watertight Meshes • David Stutz. https://davidstutz.de/a-formal-definition-of-watertight-meshes/
[80] The FreeCAD Team. 2022. FreeCAD: Your own 3D parametric modeler. https://www.freecadweb.org/
[81] Unity Technologies. 2023. Unity. https://unity.com/
[82] Thingiverse.com. 2022. Customizer by MakerBot on Thingiverse - Thingiverse. https://www.thingiverse.com/app:22
[83] Thingiverse.com. 2022. Thingiverse - Digital Designs for Physical Objects. https://www.thingiverse.com/
[84] Hannah Twigg-Smith, Jasper Tran O'Leary, and Nadya Peek. 2021. Tools, Tricks, and Hacks: Exploring Novel Digital Fabrication Workflows on #PlotterTwitter. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3411764.3445653
[85] Bret Victor. 2013. Bret Victor - The Future of Programming. https://vimeo.com/71278954
[86] Tom Yeh and Jeeeun Kim. 2018. CraftML: 3D Modeling is Web Programming. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–12. https://doi.org/10.1145/3173574.3174101
[87] Qiang Zou. 2022. Parametric/direct CAD integration. https://doi.org/10.48550/arXiv.2203.02252 arXiv:2203.02252 [cs].

APPENDIX

Base questionnaire used in the semi-structured interviews.

(1) Are you at least an Advanced Beginner with OpenSCAD? (Are you capable of creating designs and understanding the code of a model?)
(2) What is your gender?
(3) How old are you?
(4) What is your academic background?
(5) What is your current job?
(6) Do you have experience with 3D printing?
(7) If yes, tell me about your experience with 3D printing. When did you start? What were your motivations?
(8) How often do you 3D print?
    - Daily
    - Weekly
    - Monthly
    - Every two months
    - Every semester
    - Less often

Here, we define the terms of direct manipulation and programming-based paradigms that we use in the rest of the interview.

(9) What **direct manipulation CAD** applications have you used before, and what is your experience in each? Name of the application and skill level
    - 1 - Novice
    - 2 - Advanced Beginner
    - 3 - Competent
    - 4 – Proficient
    - 5 - Expert
(10) Other than CAD, what other **programming languages**, in general, have you used, and what is your skill level in each one? Programming language name and skill level
    - 1 - Novice
    - 2 - Advanced Beginner
    - 3 - Competent
    - 4 – Proficient
    - 5 - Expert
(11) What is your skill level in **OpenSCAD** ?

- 1 - Novice
- 2 - Advanced Beginner
- 3 - Competent
- 4 – Proficient
- 5 - Expert

(12) What motivated you to learn/use OpenSCAD specifically? How did you start? Did you try other applications? Why OpenSCAD and no others?
(13) Let's talk about the last three objects you 3D printed. Describe the object and motivation.
(14) Would you say that, in general, you 3D print for the motivations mentioned before, or are there other main reasons you print for?
(15) How did you get the design for those objects? Design them from scratch, Pre-existing models
(16) How do you normally get your models? Design them from scratch, Pre-existing models
(17) What CAD applications did you use to design/edit your last three objects and why?
(18) What were the major difficulties you found in the process of fabricating these objects (Including all the processes, ideation, design, configuration, printing, iteration, etc)? What is the most time-consuming part? What brings more uncertainty? (What makes you iterate more?)
(19) In general, what are the most difficult parts of the fabrication process (including all the processes, ideation, design, configuration, printing, iteration, etc)? What is the most time-consuming part? What brings more uncertainty? (What makes you iterate more?)
(20) What factors bring more uncertainty or usually make you iterate more times?
(21) Specifically in the model design part, what is the most difficult and time-consuming part? Is it something related to the software? Is it different when you use direct manipulation than programming-based?
(22) If different than the previous answer, Specifically in OpenSCAD, what is the most difficult and time-consuming part?
(23) Do you need to measure physical sizes to transfer them into the digital design? How do you do it? What tools and strategies do you use? How do you verify the correctness of the measurements?
(24) Tell me about measurement difficulties, Linear measurements, Curved and organic shape measurements
(25) Generally, when you 3D print, how often do you design the models you print from scratch? Why?
- (0%) Never
- (1% - 20%) Rarely
- (20% - 40%) Often
- (40% - 60%) Sometimes
- (60% - 80%) Frequently
- (80% - 99%) Very frequently
- (100%) Always

(26) Generally, when you 3D print, how often do you use a pre-existing model for the models you print (previous projects, friend's model, website model)? Why?
- (0%) Never
- (1% - 20%) Rarely
- (20% - 40%) Often
- (40% - 60%) Sometimes
- (60% - 80%) Frequently
- (80% - 99%) Very frequently
- (100%) Always

(27) What motivates you to design from scratch or to use a pre-existing model?
(28) Do you know model-storing websites such as Thingiverse? What others?
(29) If yes, what do you think about them? Do you use them? Do you find them useful?
(30) If you know Thingiverse, have you used the Customizer tool? Talk to me about your experience with this tool.
(31) When you use pre-existing models, in what format do you get them? (stl, obj, code...)
(32) When you re-use a **non-coded** pre-existing model, how often do you need to edit it? What types of modifications do you make?
- (0%) Never
- (1% - 20%) Rarely
- (20% - 40%) Often
- (40% - 60%) Sometimes
- (60% - 80%) Frequently
- (80% - 99%) Very frequently
- (100%) Always

(33) When you re-use a **non-coded** pre-existing model, how difficult is it to edit it? Explain what applications you use and why the level of difficulty you selected.
- 1- Very easy
- 2- Easy
- 3- Neutral
- 4- Difficult
- 5- Very difficult

(34) When you re-use a **coded** pre-existing model, how often do you need to edit it? What types of modifications do you make?
- (0%) Never
- (1% - 20%) Rarely
- (20% - 40%) Often
- (40% - 60%) Sometimes
- (60% - 80%) Frequently
- (80% - 99%) Very frequently
- (100%) Always

(35) When you re-use a **coded** pre-existing model, how difficult is it to edit it? Explain what applications you use and why the level of difficulty you selected.
- 1- Very easy
- 2- Easy
- 3- Neutral
- 4- Difficult
- 5- Very difficult

(36) Did you bring some of your previous OpenSCAD projects? Talk to me about one of them, How was the process, how many iterations did you need, and what was the most difficult part of the process?
(37) **(Hands-on exercise)** I will ask you to localize in the code the specific statements that create a part that I will point

out in the view. Share aloud the thinking process you follow to find it. Is this a task you normally do when designing: looking for a specific part in the code based on the view? What is the hardest part of doing it? What strategies do you use normally?

(38) How difficult was the task?

(39) In OpenSCAD (and programming-based), how easily can you link the output in the view to the code?

(40) What would you say is the best of OpenSCAD and the worst? What would you say is the best of programming-based CAD and the worst?

(41) What are the advantages and disadvantages of direct manipulation and programming-based applications like OpenSCAD? When do you prefer to use one or the other?