From Abstract to Tangible
An Approach to Learning Loops in Programming

Thomas Bengtsson
thomas.nilsbengtsson@gmail.com

Interaction Design
Bachelor
22.5HP
Spring 2023
Supervisor: David Cuartielles
Abstract

This thesis seeks to advance interaction design practice by exploring the potential of interactive tangible prototypes to enhance university students' understanding of abstract programming structures, such as loops. It addresses prevalent challenges, including the difficulties students encounter when initially learning about loops, and scalability issues inherent in tangible teaching tools within this research area. Two distinct tangible learning methodologies are considered: unplugged programming and tangible computing. The primary contributions of this thesis lie in the creation of innovative tangible learning tools, the emphasis on the scalability of such tools, and the illumination of how physical prototypes can inspire digital software design.

Keywords: tangible tools, tangible computing, unplugged programming, programming, loops, mental models, metaphors.
Acknowledgements

I want to extend my gratitude to everyone who has been a part of this thesis journey. Special thanks to my supervisor, David Cuartielles, for his guidance and input that helped shape the project. I am also grateful to my classmates for their support throughout this process. Lastly, thanks to everyone who participated in the user tests, your time and insights have been invaluable to this project.
# Table of content

Table of content ............................................................................................................ 4

1 Introduction .................................................................................................................. 6

1.1 Context ....................................................................................................................... 6

1.2 Aim ............................................................................................................................ 7

1.3 Delimitations ............................................................................................................. 7

1.4 Ethical considerations ............................................................................................... 7

1.5 Thesis Structure ....................................................................................................... 8

1.6 Research Question: ................................................................................................. 8

2 Theoretical Background ............................................................................................. 9

2.1 Teaching programming ............................................................................................ 9

2.1.1 Mental models .................................................................................................. 10

2.1.2 Metaphors ....................................................................................................... 10

2.2 Computational thinking ......................................................................................... 11

2.2.1 Unplugged programming .............................................................................. 11

2.2.2 Tangible computing ...................................................................................... 12

2.3 Theories of learning ............................................................................................... 12

2.4 Summary of Key Theoretical Perspectives ......................................................... 13

3 Related design ........................................................................................................... 14

3.1 UP in education ..................................................................................................... 14

3.2 Tangible computing devices in education .............................................................. 15

4 Mapping ..................................................................................................................... 17

4.1 Designing for loop comprehension ...................................................................... 17

4.2 Adapting complexity ............................................................................................. 18

5 Methods ...................................................................................................................... 19

5.1 Research through design and user testing ......................................................... 19

5.2 Sketching and prototyping ................................................................................... 20

5.3 Semi-structured interviews ................................................................................ 21

6 Design process ........................................................................................................... 21

6.1 Interviews with educators ..................................................................................... 22

6.1.1 Benjamin ........................................................................................................ 22

6.1.2 Daniel ............................................................................................................. 22
1 Introduction

1.1 Context

As technology continually evolves and society's dependence on it deepens, the demand for programmers continues to grow (Melcer et al., 2017). Despite this increased need for programmers, the dropout rates of students in their first introductory programming courses are high (Luxton-Reilly, 2016). There are a wide variety of reasons why these students are struggling. A few of the reasons are related to students’ ability to understand abstract programming concepts, problem-solving, logical reasoning, and lack of both motivation and self-confidence (Gomes and Mendes, 2014). When students struggle with the novelty that programming presents, this can lead to negative self-belief, which in turn creates barriers to learning (Kinnunen and Simon, 2010).

Traditional methods for teaching programming imply students being passively engaged in the learning experience. Arawjo et al. (2017) argue that it is not an effective learning environment, and present an alternative teaching method that has shown to be beneficial for novice programmers, namely to use tangible teaching materials related to the subject that the students are learning.

Loops is one of the fundamental programming concepts. It is a type of control structure that allows executing a block of code repeatedly based on a specific condition. Since loops are one of the basic programming concepts, it is important that people develop the skill to effectively implement loops in their codes. However, people who are new to programming can face difficulties when trying to learn this specific programming concept (Cetin, 2020). The following three stages have been outlined by Cetin (2015) to describe the process of learning about loops in programming:

1. Pre-action stage: The aspiring programmer is developing an understanding of the programming concept, for example, when it can be utilized. However, they are unable to correctly write the syntax.

2. Action stage: The student is able to write functional code but they have only obtained the knowledge to solve simple problems using loops.

3. Process stage: The programmer has developed an understanding of how loops execute but also further knowledge on how to handle the data coming out from a loop such as being able to print the result. This results in the programmer having the ability to solve more complex problems by utilizing loops.
1.2 Aim

The aim of this thesis is to explore how to help university students who are new to coding to develop their understanding of programming structures by utilizing tangible tools. To fulfill the aim I will also explore how the design and implementation of tangible tools can be scaled to support a contemporary university classroom setting.

Furthermore, I will be exploring specific problems novice programmers face when learning about loops. Cetin (2020) argues that when developing teaching approaches in regard to programming concepts it is important to entertain questions such as “how to help students develop their understanding,” “how to help them overcome their misconceptions,” or “how to help students so that they do not construct misconceptions” (p.3). By exploring questions such as the latter it is possible to create teaching approaches that help people understand at low level how loops work.

1.3 Delimitations

For this thesis, I will be focusing on the first stage of programming learning (Pre-action stage) as presented by Cetin and mentioned in section 1.1 (2015). Therefore, the focus of this thesis will be lying on developing a basic understanding of how loops work and not, for example, other aspects such as understanding the syntax of a loop.

An additional delimitation of this study lies in the composition of the student participants engaged during the research. The ideal demographic for this study, given that the tangible tools are designed for novice programmers, would have been students with limited programming experience. However, due to accessibility constraints, the pool of participants surveyed primarily consists of students who already possess a certain level of programming experience. This factor potentially limits the scope of the findings. These students, having more experience, might interact with and perceive the tangible tools differently from absolute beginners. Therefore, the results and insights obtained might not fully capture the experiences and challenges faced by novice programmers, which is the targeted user group. However, Section 6.2.2 discusses the benefits associated with the chosen participants.

1.4 Ethical considerations

Due to the nature of this thesis focusing on working within the realm of educational research, students were involved throughout the design process as participants in user tests. This thesis project followed the eight ethical rules and data handling guidelines presented by the Swedish Research Council (2017) to ensure that ethical research principles were followed. All of the participants of the user tests signed consent forms which guaranteed them confidentiality (appendix 1). The data which I collected from the user tests are presented in a way in which the participants cannot be identified.
Furthermore, before signing a consent form the participants were informed about the purpose of the research, what to expect from their participation, and if they wish to withdraw their consent to participate in the project they can do so at any moment and have their data deleted.

During the thesis, information was also gathered from two experts in the field of programming through interviews. Both confidentiality and anonymity were offered, however, they both gave consent for the interviews to be recorded and for them to be able to be identified in connection with the information they provided. They were assured that all of the data gathered was used only in the context of this thesis and that the interviews were to be deleted after the project was complete. Leaving the only trace of the information gathered from the interviews is what is mentioned in this paper.

It is important to acknowledge the ethical implications of digital tools created for education. Selwyn (2016) argues that the implementation of technology in education can create educational inequalities. Such as the gap in accessibility of tools for students depending on their access to modern digital technology, described by Van Dijk (2006) as a “digital divide” (p.1). Furthermore, education is a fundamental right and one should strive to design a tool to be inclusive, aiming to minimize barriers to access and use (Warschauer and Matuchniak, 2010). Therefore, while creating digital prototypes the implications of a digital divide and access to the technology were taken into account while making design choices. I strove to design tools that consider broader societal implications and aim to promote social equity.

1.5 Thesis Structure

Chapter 1 outlines the project’s aim, context, and research questions. The following chapter (Chapter 2) explores approaches that can be used to teach programming by investigating both physical methods but also theoretical approaches. Chapter 3 explores existing related designs which are relevant to this project followed by Chapter 4 where I reflect on what I learned from the previous two chapters with the purpose of finding a gap in the current landscape in regard to teaching loops. Moving on I explain different methods -in Chapter 5- that will be used to guide my design process, which will be presented in Chapter 6. Following my design process Chapter 7 will focus on discussing and reflecting on both my design process and findings. To end the paper, Chapter 8 will provide a conclusion of the findings, what my thesis project contributed to the field of interaction design, and future work.

1.6 Research Question:

The research question for my thesis project is the following:

- How can interactive tangible prototypes help university students develop their understanding of how abstract programming structures -such as loops- work?
To help me unpack the research question I created following sub-questions during the process of this thesis, the first two questions are explored in depth in section 4.2 and the third one in section 6.5:

- What are the struggles novice programmers at university level face when learning about how loops work?
- How can the combination of tangible tools and their respective metaphors be utilized to help students develop mental models of how loops work?
- How is it possible to create a tangible learning tool that can be provided to a large number of students but requires a low amount of resources?

2 Theoretical Background

This chapter delves into the limitations of traditional programming education strategies, accentuating the value of mental models, metaphors, computational thinking, alongside interactive methodologies like unplugged programming and tangible computing. The chapter further investigates influential learning theories that prioritize active participation, the act of creation, and social interaction as vital components of the learning process.

2.1 Teaching programming

Traditional teaching methods using static materials such as books or slides to teach dynamic programming concepts are not optimal teaching approaches. There is a need to use visualization tools that can project mental representations of the concepts at hand (Gomes & Mendes, 2007; Zhang et al., 2013). Ben-Ari (2001) argues that traditional ways of teaching do not provide adequate ways for students to learn since these methods tend to lack the ability to challenge students’ previous beliefs and neither do they facilitate the creation of usable mental models. Teachers should not assume that their students have the required underlying mental models to understand different programming concepts. Hence, there lies an urgency in helping students build appropriate mental models at an early stage of their learning programming journey.

The traditional teaching methods also lack the ability for students to actively collaborate with each other. When teachers use a masterclass model with materials such as slide-show presentations as a teaching method, the students are mostly passively engaged in the activity. On top of that, reading books -another traditional learning technique- is more of an individual activity. These bring along negative consequences since one of the reasons
that students fail to reach an understanding of programming concepts is due to the lack of active involvement in learning sessions (Ismail et al., 2010).

2.1.1 Mental models

The concept of mental models was first proposed by Craik (1943). He argues that humans perceive the world actively rather than passively through the use of mental models. The human mind creates models of reality with the purpose of clarifying things, predicting future events, or logical thinking.

It is important to note that for a mental model to be effective, its structure must be similar to the real-life phenomenon it represents. If the structure does not carry any similarities there is an increased risk of misconceiving the phenomenon. A model does not need to be as complex as the original phenomenon it represents, a rather simplified mental model can be more effective (Sasse, 1997).

When it comes to programming, mental models can be effective for students when trying to understand abstract concepts which are both invisible and untouchable. However, due to the inability to see in real-time what is happening when a program is executed, people are not able to construct their own-or a correct mental model (Ma, 2007). Novice programmers tend to have poorly developed mental models (Bednarik, 2012). It is important to consider the construction of proper mental models in the early stages of learning a programming language (Sleeman et al., 1986).

2.1.2 Metaphors

Metaphors are used as tools to describe an object, idea, or event by drawing on concepts or ideas that are not associated with the latter but rather create a comparison or analogy between the two (Indurkhya, 2013). Lakoff (1993) describes the usage of metaphors as the main way which allows people to build an understanding of abstract concepts. By taking an abstract concept and describing it through a metaphor it is possible to turn it into a more concrete subject.

When utilizing metaphors it is important that a few things are taken into account to optimize their use. The object or event that is being described is called the target; there should be an underlying similarity between the target and the metaphor. Since metaphors are used externally to a source, it is important for those using a concrete metaphor to have some background knowledge of the target so that a connection can be made between the two (target and metaphor) (Hamilton, 2000).

Though metaphors offer positive opportunities to navigate the world as mentioned above, there is also an inherent risk of metaphors yielding negative results. Such as, if a metaphor is used in an incorrect situation where it can be seen as either misleading or simply inaccurate. Or, where the
metaphor is poorly constructed leading to the full scope of the target is not conveyed rather only a limited scope is shown to people (Black, 1979).

2.2 Computational thinking

Computational thinking (CT) can be understood as a problem-solving approach that allows people to envision problems and their respective solutions in a way that a computer could understand and execute (Wing, 2006). Bell et al. (2009) argue that the increase in people's awareness of the importance of both coding and data manipulation can lead to a decrease in people's interest in engaging in the field of computer science. With this paradox in mind, society has turned to make a stronger emphasis on developing students' CT in education. There is a growing emphasis on STEM (acronym for Science, Technology, Engineering, and Math) activities, involving the development of engaging, hands-on learning activities aimed at keeping students interested and actively involved in their learning process (Freeman et al., 2014). In this section, follow several approaches to learning about CT, namely unplugged programming and tangible computing.

2.2.1 Unplugged programming

The concept of unplugged programming (UP) is used in the context of developing people's CT and understanding of programming concepts. It is an approach that allows people to develop their CT without the need to work with any electronics or have an understanding of how to code (Aranda and Ferguson, 2018). This is beneficial since when learning programming concepts at first, the computer can be a distraction due to learning how to code can create a barrier for some people due to the difficulties it presents. Furthermore, this can lead to a lack of motivation and low engagement in learning activities (CSUnplugged, 2015).

There is a wide variety of learning activities that can take place when engaging in UP; two examples are embodied interaction and tangible interaction (Aranda and Ferguson, 2018). Embodied interaction harnesses physical engagement, body movement, and at times, tangible tools to facilitate the understanding of various concepts, providing a unique, immersive learning experience. Tangible interaction in the context of unplugged programming involves the use of physical objects that learners can manipulate to represent and understand programming concepts, providing a concrete, tactile learning experience that bridges abstract computational thinking with the physical world. The two mentioned examples provides an environment where students are physically engaging with the learning material, therefore UP tends to take upon a constructivist approach to learning (CSUnplugged, 2015).

Since UP is being used to develop CT, which includes the understanding of programming concepts, it is important to note that at some stage a student
must move away from UP and venture into coding practices to advance in their quest of learning how to program (Aranda and Ferguson, 2018).

2.2.2 Tangible computing

With the rapid advancement of technology in the late 1990s there was an effort made to move Human-Computer Interaction (HCI) away from the screen and into the physical world. To describe this phenomenon the word "tangible" was adopted by Ishii and Ullmer (1997), a term which became popular when describing the activity of using either physical artifacts or interactive surfaces to manipulate and represent digital information. Interaction design, as a discipline, has later coined the term "tangible computing" to describe a paradigm of computing where interactions happen beyond the keyboard and the mouse, the most common human computer interfaces until the arrival of touch interfaces. The traditional way of manipulating data with inputs such as a mouse or keyboard does not fall into the area of tangible computing because of their limitations as input mechanisms. One of the benefits that tangible computing brought is the ability to provide learning experiences that incorporate both physical artifacts and the ability to move through space, creating a relationship between the physical artifact and the student that references sensorimotor schemes which is the foundation for abstract thought development (Macarana et al., 2012).

2.3 Theories of learning

Ackermann (2001) discusses the difference between the theory of constructivism (by Jean Piaget), constructionism (by Seymour Papert), and social constructivism (by Lev Vygotsky). All three of these theories are related to learning but are distinct educational philosophies. Constructivism is a theory that suggests that to gain knowledge students must actively engage with the experiential world with their pre-existing cognitive structures, rather than gaining knowledge through a passive medium such as lectures or textbooks. Constructionism argues that people learn by creating something tangible that can be shared with others and, at the same time, reflect upon their creative process. Social constructivism proposes social interactions and collaboration with other people as the core of the education process.

Though these theories offer different perspectives on how learning happens they all focus on “building knowledge structures” (Papert and Harel, 1991, p.2). It is important to note that these theories do not “compete” with each other, in fact, they can complement each other. They offer different lenses through which to examine and understand learning. Providing insights into diverse elements that shape learning.

Papert (1980) highlights the benefits of an early immersion of children into the world of computer literacy in his book "Mindstorms: Children,
He argues that when children are encouraged to explore, experiment, and physically construct their ideas, they not only grasp the subjects at hand better but also develop critical skills such as creativity, problem-solving, and self-confidence. From this perspective, introducing programming principles like loops at a young age could be beneficial. This initial exposure might serve as a stepping stone, paving the way for a stronger understanding when learners encounter more intricate programming constructs in their university courses. The impact of these factors on the effectiveness of the tools I developed in this project is further discussed in Section 7.4.

2.4 Summary of Key Theoretical Perspectives

The theoretical aspects discussed in this chapter outline various methods for teaching programming effectively. Traditional teaching methods such as the usage of static materials fail to actively engage students in the learning experience and are not optimal methods to facilitate the learning process for students learning programming. Strategies, such as the use of mental models, metaphors, UP, and tangible computing, were found to be methods that are efficient in helping students learn programming.

Understanding and teaching programming can greatly benefit through the conscious use of mental models and metaphors at the time of creating the course materials. Mental models are internal constructs that provide simplified representations of how things work and can aid students in grasping abstract programming concepts which are otherwise invisible and intangible. Due to the abstract nature of programming concepts, it can be a struggle for students to develop correct mental models. Metaphors can be used to facilitate a student’s comprehension of a programming concept by relating them to familiar ideas. However, it is crucial that a metaphor offers an accurate and relevant comparison to prevent misunderstandings about the programming concepts being taught.

To develop students' computational thinking the usage of UP and tangible computing have emerged as significant tools in this endeavor. UP allows students to understand programming concepts without the need for electronics. This approach avoids the distraction computers can cause during the initial learning phases, making the learning process less intimidating and more engaging. Tangible computing offers a more physical approach to interaction with digital data. This approach combines physical movement and abstract thought to enhance the understanding of programming concepts.

The different theories of learning by Jean Piaget (constructivism), Seymour Papert (constructionism), and Lev Vygotsky (social constructivism) provide different perspectives on the process of learning. These theories complement each other, emphasizing the importance of active engagement with the world, the process of creating, and social interactions in learning, respectively.
This chapter thoroughly explores theoretical aspects beneficial for teaching and learning programming. It sets the foundation for my project, I will be developing artifacts to connect the physical world as metaphors that will help illustrating abstract programming concepts, thus helping building mental models around CT. The upcoming chapter will delve into existing designs in UP and tangible computing, building on the solid theoretical foundation laid in this chapter.

3 Related design

Once set the theoretical framework of the thesis it becomes relevant to explore existing design work that could be used as academic reference and inspiration in the making of this project. The following chapter explores examples in the field of unplugged programming (UP) and tangible computing aimed at helping students develop an understanding of programming concepts.

3.1 UP in education

Curzon et al. (2014) present an UP prototype that aims to introduce students to variables and assignment sequences. The prototype includes multiple boxes which are labeled as variables and within the boxes there are strips of colored paper that represent different values. To simulate that a variable can be changed, overwritten, or copied, the boxes have both built-in shredders and copiers. When a user is given instructions that say that a variable has been overwritten, the user proceeds to remove the colored paper and replace it with a paper that represents the value. This is a hands-on approach for explaining how variables and valuables can be manipulated.

A simple way in which conditionals can be demonstrated is through the use of a card deck. The website Code.org has developed an exercise that focuses on if/else statements (Code.org, n.d.). The exercise requires learners to draw a card from a deck and, depending on the number on it, there will be a specific output (Figure 1). For example, if a card with the number 10 has been drawn then the other participants must clap, however, if number 10 is not drawn then the person who drew the card must shake their head. Though the activity itself is not complex it demonstrates the core principle of conditionals which is that the outcomes will be different depending on whether a specific condition has been met or not.
In the context of teaching programming through the use of tangible computing devices, LEGO MindStorms\footnote{Official LEGO MindStorms product page (product has been discontinued): \url{https://www.lego.com/sv-se/product/lego-mindstorms-ev3-31313}} is a popular tool (Figure 2). It is a toolkit that can be reconfigured in multiple ways to become a robot capable of performing tasks programmed by the students. The device allows students to create tangible prototypes through the use of standard LEGO bricks, sensors, and motors. Through the use of a graphical programming interface, students can create programs to control their robot. LEGO MindStorms can be used by students of a broad age range, from secondary school children and all the way to university. As a matter of fact, Klassner and Anderson (2003) argue that the product will be more suited for university level in the context of teaching students programming concepts. Another benefit of MindStorms is that many people have played with LEGO when they were younger leading to a spark of interest to work with the latter in an educational setting.
Yet another tangible computing device following a different paradigm than LEGO is a puzzle game created by Melcer et al. (2017). The objective of the puzzle is to connect wooden blocks to create a program that will guide a virtual robot through a maze (Figure 3). The wooden blocks represent different types of programming concepts and the puzzle has a variety of skill levels. When the levels increase, the complexity of the puzzle also increases, forcing students to learn further to enlarge their repertoire of programming concepts and solve the following puzzle.

Figure 3. The wooden blocks that are used in the puzzle (Melcer et al., 2017).

It was hard to find many examples focused solely on teaching loops, which is the main computing concept explored in this thesis. Code Jumper², a tangible computing device is one of those. It was developed to support children in learning programming concepts such as loops (Figure 4). The device consists of individual pods, each representing a line of code and a physical board that serves as the connection point for the pods and the interpreter of their settings. Students can create and execute programs by connecting the pods to the board, enabling tasks such as repeating a command a specific number of times or until a condition is met. Each pod is equipped with a dial and a button, which students can manipulate to alter the specific code, enabling changes to specific commands and conditions. The board also produces audio feedback, indicating the current position within a loop in the program.

4 Mapping

As an outcome of the research phase, it became clear that both UP and tangible computing tools can help users develop an understanding of how programming concepts work. The purpose of this chapter is to map my findings to find a gap to explore in the existing design space.

4.1 Designing for loop comprehension

When analysing existing designs that use a tangible aspect of teaching programming concepts, the focus on teaching loops is commonly a passive part of the design. An example of this is the puzzle that Melcer et al. (2017) present. Students can perform actions that represent a loop such as making a robot repeat a certain move a specific number of times. While this gives students a taste of the repetitive function that loops can perform, it may not be sufficient for a full understanding of the underlying principles and structures that govern loops in programming. The loop in this context is abstracted into a simple repeat action, which may fail to introduce key facets of loop constructs, such as initialization, condition evaluation, and incrementation, as well as the termination of the loop.

Throughout researching existing designs it was clear that there do not exist many artifacts which were created with the sole purpose of learning about loops. The one example I found, Code Jumper, has children and not university students as target group, thus, the complexity of the design is adapted to a certain level that doesn’t correspond to what I am up for with this thesis. From a programming metaphor perspective, Code Jumper can barely depict the very basic idea of a loop as the iteration over a sequence of commands.
4.2 Adapting complexity

Developing teaching elements for university students requires considering not just an adaptation of the language used in the education materials, but also the kind of interactions, pedagogical methods, and the like. This correlates to what Klassner and Anderson (2003) argued regarding LEGO MindStorms. Despite the educational kit’s original design for children, they argue that it can be effectively adapted for university-level education if the taught content is appropriately adjusted in complexity, making it both engaging and sufficiently challenging for university students. Consequently, when employing tangible tools to learn about loops or similar concepts, it’s essential that, regardless of the simplicity of the tool itself, the educational content is tailored to match the academic level of the learners.

Sub-questions

While most of the teaching tools that I have researched so far have a passive approach to teaching loops, Code Jumper is an exception, offering a more in-depth understanding of the concept. However, Code Jumper is primarily designed for children, implying that the complexity of loop concepts it teaches is adapted to a level a child can understand. Given my aim to develop tangible prototypes for university-level students, it is essential to identify the specific challenges this target audience faces when learning about loops. This will ensure that the prototypes I create are both suitable and appropriately adapted to my audience. Therefore, I have formulated the following sub-question:

- **What are the struggles novice programmers at university level face when learning about how loops work?**

Given that existing designs tend to teach loops passively, potentially hindering a student’s ability to develop mental models of loop functionality, I am driven to create tangible tools that provide a learning environment where mental models of how loops work are actively developed. I will utilize metaphors to help in this endeavour of making abstract concepts more concrete. Therefore, I have formulated the following sub-question:

- **How can the combination of tangible tools and their respective metaphors be utilized to help students develop mental models of how loops work?**
5 Methods

This chapter provides a comprehensive overview of the research methods integral to this thesis project, including Research through Design (RtD), semi-structured interviews, as well as the creation of sketches and prototypes. Adopting such an approach allows for extensive material explorations, purposely tailored to translate the abstract programming concept of loops into tangible and comprehensible forms. The forthcoming sections will elaborate on the rationale behind choosing these particular methods, their practical application in this research context, and their distinctive contributions to the overall study.

5.1 Research through design and user testing

When following RtD it is important to realize that designers should not aim for standardization but rather explore and produce new, conceptually rich artifacts (Gaver, 2012). Durrant et al. (2017) acknowledge RtD as a method where the process of creating and testing prototypes is where knowledge is generated. Therefore, user testing will be used in this thesis since it provides designers with feedback on the prototypes by having people interact with them (Bastien, 2010). Furthermore, the knowledge generated from the user tests will affect the design decisions made during the development of the prototypes.

Though RtD has been actively used within the design community, there is not one set definition of what the method actually entails or what outcomes a research project should have when using it in a design process. Zimmerman et al. (2007) present four criteria with the intent of formalizing RtD and providing a way of evaluating an interaction design research contribution. What the following criteria state is relevant to the goals of this thesis and will therefore be used as guidelines throughout the whole design process:

**Process:** The design process must be well documented so that others can reproduce the process and the chosen design methods should be justified.

**Invention:** The results from the research project should be clearly positioned in regard to related work.

**Relevance:** There should be a clear distinction between the current state of the world and the state that the researcher wishes to achieve, also how the work that was conducted by the researcher helped achieve their preferred state.

**Extensibility:** It should be clear what possibilities there exist for future work to build upon the project.
5.2 Sketching and prototyping

Throughout my design process, both sketching and prototyping were methods used when working with design concepts. Moussette (2012) argues that sketching can be seen as a rapid way of prototyping but the two methods are used for different purposes. Buxton (2010) states that the difference between the two can be found when it comes to cost, timeliness, quantity, and disposability. Sketches compared to prototypes are more cost-efficient and require less of a time-investment as visualized by Figure 5. Thus, designers have the opportunity to dispose of concepts having invested less time, money, and effort. Sketches can be created in higher quantity compared to prototypes.

There are different types of ways to sketch, such as the use of pen and paper or Sketching in Hardware (Mousette and Dore, 2010; Moussette, 2012). Moussette and Dore (2010) introduce Sketching in Hardware which is the process of creating sketches through the use of physical materials. This method is useful when designers want to grow their knowledge about the material they are working with. For my project, I chose to use Sketching in Hardware instead of using pen and paper. This is because I was focusing on developing tangible tools and using Sketching in Hardware allowed me to try out different design concepts in a format where they already are tangible at the sketching phase.

The benefit of counting with a prototyping phase in a project, is that it allows for design concepts to be refined (Lim et al., 2008). Following the initial lo-fi physical sketches, I refined my design concepts towards the following design of the tangible educational tools. Moving to the prototyping phase allowed me addressing the limitations I found in the interaction with the sketches. This leads to the possibility of being able to improve aspects such as the functionality of the sketches in their transition to prototypes. Furthermore, prototyping transforms the sketches into a more comprehensive and detailed representation of the design. This opens for a deeper conversation with the study participants, since a refined prototype can demonstrate the nuances of the design concept it comes from much more effectively than a lo-fi sketch.
5.3 Semi-structured interviews

For this thesis, semi-structured interviews will be used to gain insights from the perspectives of educators with experience teaching programming at a university level. According to Kvale and Brinkmann (2014) this method enables the collection of qualitative information rooted in a person’s experience. The questions employed are flexible and not strictly scripted. Guiding questions are utilized, but as an interviewer, one has the opportunity to ask follow-up questions and probe emerging areas of interest (Wilson, 2012). While the goal of this thesis is to create tangible learning tools, it is beneficial to interview people who have experience in teaching programming. This is because it allows me to gain insights in regard to what they experience novice programmers struggle with when learning programming (including loops).

6 Design process

This chapter outlines the design process, tracing the path from the initial interviews with educators to the prototyping stages. The process begins with gaining insights from interviews, followed by planning and defining the research focus. The ensuing stages involve user testing and sketching in hardware, leading to iterative improvements. Finally, the prototyping process encompasses unplugged programming and tangible computing methods, to further refine and explore design concepts. The following sections chronicle this journey, highlighting the encountered challenges, exploited design opportunities, and critical decisions made.
6.1 Interviews with educators

My design process started by interviewing two educators who are experienced in the context of teaching programming at a university level.

6.1.1 Benjamin

Benjamin Maus is a lecturer at the Faculty of Technology and Society at Malmö University. He teaches at both undergraduate and graduate level. When it comes to undergraduate students, he teaches several courses in the Bachelor of Information Architecture. At the master’s level, he is a lecturer in Innovation for Change. Occasionally, he also teaches at other programs such as the System Development and the Business and IT ones.

Benjamin is a part of courses where students learn HTML, CSS, and JavaScript at an introductory level. In these courses, he partakes in more of the UX and UI modules but works closely together with another lecturer who is focused on teaching the programming elements of the course. This provides him with an overall understanding of the approach that his colleague uses when teaching programming. He is also involved with teaching programming in a basic introduction to Arduino (an abstraction of C++), in the form of workshops where Benjamin focuses on teaching how it is possible to code simple applications (working with sensors and LEDs).

When it comes to students learning programming, a problem that Benjamin has observed is that students may lack the motivation to learn about the subject. This can be for a variety of reasons, for example, students find that it is a difficult subject to learn, or they are simply not interested because they see it as a too small part of their education.

To help students that are struggling Benjamin suggests that there needs to be explorations for a more personalized teaching experience, one that adapts to the pace of the student. However, since there may not be enough resources available at universities, this opens up questions such as, how can we integrate digital resources? How can we make sure that resources adapt to the style of the students?

In regards to teaching programming, two things that Benjamin has found beneficial is letting students engage practically with what they are learning instead of passively listening. For example, when learning to create basic applications with Arduino he invites students to actively try to make sensors work. The second activity is collaboration because it puts the students in the situation of learning from each other.

6.1.2 Daniel

Dr. Daniel Spikol works for the University of Copenhagen in the Department of Computer Science and also the Department of Science Education. Furthermore, he is one of the members of a new research center called the Center for Digital Education. His interests are, to an extent, related to
computational thinking and how people learn programming. He teaches a variety of classes, some being programming-related classes at the bachelor's level. He has been teaching programming for around 16-17 years at university level. In regards to programming languages, Daniel has experience in teaching Python, JavaScript, and C++ (Arduino platform).

Daniel has experienced that students face a wide variety of struggles when learning about programming concepts, one of these struggles is understanding the abstract nature of the concepts. In regards to loops, it has been a common problem that students struggle to understand the difference between how a for loop and a while loop work. Moving towards more complex loop concepts, students struggle with understanding how nested loops work, more specifically the idea that there can exist a loop inside of a loop.

Using tangible tools is a method of teaching which Daniel thinks would be beneficial to implement in programming teaching environments. Previously he has used physical objects to demonstrate programming concepts such as loops but he believes that it has not been beneficial. The reason is that the students have not actively interacted with the objects, rather he has been the one demonstrating. This is because it is difficult to implement tangible tools in teaching practices where there are 80-140 students attending the class. Providing enough resources so that all of the students can be actively engaged in the learning experience is a challenge. However, he believes that using tangible teaching tools to help students understand abstract concepts can work and is something he would be interested in implementing in his teaching.

6.1.3 Summary and Design Guidelines

The insights from Benjamin and Daniel’s interviews revealed key challenges and opportunities in teaching programming. Students often grapple with motivational issues, understanding abstract programming concepts, and the specific task of comprehending loop structures. The use of teaching material that promotes active engagement from students could enhance individual motivation, comprehension, and foster collaboration. Further, tangible tools present a potential means to elucidate abstract concepts, including loops. However, their effective implementation often clashes with the realities of large class sizes and limited resources. From these insights, a set of design guidelines can be established: create tangible prototypes to facilitate a hands-on learning environment with emphasis on the comprehension of loop structures; provide opportunities for collaborative learning; ensure the scalability of these teaching strategies for larger class settings.
6.2 Planning

Section 6.2 delves into the planning phase of this research, a critical stage that sets the foundation for the subsequent design and implementation processes. The planning phase consists of three major tasks: defining the research focus based on insights from educator interviews, selecting user testers who will provide valuable feedback throughout the design process, and organizing user test sessions to ensure a smooth, effective evaluation of the prototypes. This section will walk through the rationale behind each decision, highlighting the importance of strategic planning in developing tools for novice programmers.

6.2.1 Defining the Research Focus

Following the interviews with educators I decided to focus on creating tangible tools which can aid novice programmers in understanding what the difference is between a for loop and a while loop, and also develop their understanding of how a nested loop works.

The tangible tools that I will be creating will not be focusing on helping students understand the programming syntax of the different loops, rather I will be focusing on developing an understanding of the following:

- A **for loop** executes a specific number of iterations and is complete when all of the iterations have been executed.
- A **while loop** does not iterate a specific number of times rather it keeps iterating until a specific condition has been met, which could be external to the actual loop such as the arrival of certain command via a communication channel, a time interrupt, etc. When the condition is met the loop will be complete.
- The **nested loop** is a loop within another loop and when executed the inner loop iterates completely for each iteration of the outer loop.

6.2.2 Selecting User Testers

Throughout the whole design process, I user-tested with the same five participants. The ages of the user testers range from 22-25 years old, whereas two of the participants are female and the remaining are male. All of the testers are currently studying for a bachelor’s degree where they all have completed at least one programming course and have continued developing their skillset after that course. Three of the participants are experienced in the programming language Java and the two others have experience in both Javascript and C++. In terms of the languages that the participants have learned, they are all relevant to this project since the underlying principle of how the different loop concepts work is the same. There is nevertheless a difference in regards to the syntax but in the context of what I am exploring that is not relevant.
All of the participants defined themselves as beginners in programming. However, due to having gained basic experience such as having a clear understanding of different data structures, they are no longer considered novice programmers. All of the participants have developed an understanding of the loop concepts which are being explored in this thesis. Conducting user tests with participants who already have experience in the concepts being explored can be seen as beneficial for the following reason:

- Being experienced with the concepts allows the participants to provide a better assessment of whether the tangible tools are conveying the desired meaning.
- Identify gaps in both the design and functionality of the tangible tools.
- Having an understanding of the concepts allows for deeper reflections and conversations during user tests.

It is important to note that people who have acquired years of programming experience can provide similar information as a person who is a beginner. However, conducting user tests with participants who are closer in experience to novice programmers can provide insights to further develop tangible tools that are more suited for novice programmers. This is because more experienced programmers may take certain aspects for granted in contrast to novice programmers. Hence, engaging participants with slightly higher skills seems to be a better fit for the purpose of this thesis project.

6.2.3 User test sessions

There were four individual user tests conducted with each participant. I decided to take this approach instead of conducting collective workshops due to participants' schedules colliding which resulted in difficulties in planning. Choosing to conduct user tests individually allowed me to have each participant test each prototype throughout the whole design process. This also gave me the freedom to conduct user tests on short notice, which provided me with the opportunity to still be able to receive feedback on prototypes that were iterated quickly.

Drawing on the advice of Houde and Hill (1997), during the user tests, I prepared the participants by providing a clear explanation of the prototypes. I made explicit both the design questions that were being explored and, those that were not, when they interacted with the prototypes. After trying out the prototypes I would ask the participants questions to get them to reflect on their experiences. While I had received clearance from the participants to record the sessions I decided not to. The reason was that some of the participants stated that they did feel pressure to “perform” while their voice was being recorded, hence I decided not to record to create a comfortable environment for them. Instead, I wrote down insights that I thought were valuable through the use of pen and paper. I do acknowledge that there are downsides to not having recorded the user tests such as if there were insights
that I did not write down but which could have become relevant later in the project.

6.3 Sketching in Hardware

The creations from this initial process are defined as “sketches” to differentiate them from the more developed “prototypes” that emerge later. Sketches, being less formalized, are faster to implement and easier to discard, making them ideal for the exploratory phase of design where rapid ideation and iterations take place. In contrast, prototypes are used in later stages where designs are more concrete and require a higher level of detail and stability.

6.3.1 Iteration 1: the pipe

The first sketch which was created during the design process was a tangible tool developed with plastic pipes (Figure 6). The choice of using non-transparent pipes was due in part to the limited resources available at the time. With this sketch, a user can place a ball inside of the pipes and then proceed to move the ball around by twisting and turning the object. A user can start the loop by entering the ball on one side and then get the ball to advance until it falls out through the other hole of the pipe.

The aim for iteration was not to demonstrate how a loop functions but rather to receive feedback on whether using pipes to create a tangible tool is an effective way of demonstrating how loops work.

Figure 6. Tangible tool made out of pipes

6.3.2 Feedback from the user testing 1

Through user testing of the first sketch, it was confirmed that using non-transparent pipes as the material would not be effective. The main feedback that I received was that since the participants cannot see the ball running inside of the tube, they had to focus on “feeling” where it was in the pipe. One
participant stated that “my focus is solely on not losing the ball rather than reflecting on what I am interacting with and what I am supposed to learn from the tool.” In other words, the sketch lacked a visual feedback dimension which took away from the experience. Though there was positive feeling about the hands-on interaction that takes place when engaging with the artifact, there were other factors that would get me to reject further pursuing this line of work. A significant issue was the inherent complexity of working with pipes as a material due to their physical properties. This would likely pose difficulties when trying to create more intricate designs, such as illustrating different types of loop constructs.

Two participants made the same analogy of creating a tangible tool that took inspiration from a race track. One of the participants stated that it would be a good metaphor because “the loop would open up in front of the user which would provide a visual aspect to the tool while interacting with it”. Furthermore, they thought that this would help users build a mental model of how loops execute. Building on the metaphor proposed by the participants, parallels can be drawn with the concepts articulated by Calliot (1993). Much like a race track, loops can be likened to a closed circuit where the flow of control (akin to the flow of electrons) is confined within a set path and cannot escape until certain conditions are met.

6.3.3 Iteration 2: carving foam

For iteration 2, I aimed at creating a sketch built using the feedback I received from the first round of tests. Furthermore, I developed three different distinct sketches representing the *for loop*, *while loop*, and *nested loop*. While developing the sketches I used metaphors to describe the different loop concepts to help users create mental models of each one of them. The sketches were created using styrofoam. I carved out “tracks” in the material so that the ball could be pushed around through the tracks. This interaction was not only visible, but also slower than having a marble inside of a pipe.

**For loop**

The *for loop* demonstrated has both a starting position and a finish point, there are also three holes in the top corner where a piece of paper can be used to count how many iterations the loop has executed (Figure 7).

The metaphor which was used to describe how a *for loop* works is comparing the concept to a racing track where the ball can be pushed through for a specific number of laps, for each lap that has been completed a counter will increase (or the participant will have to make it increase) until the determined number of laps have been executed. When all of the laps have been completed the ball is positioned at the finish point.

The mental model which is being developed for the user is the understanding that *for loops* runs for a predefined number of iterations and, when all of the iterations are completed, the *for loop* ends. By also implementing a feature
that tracks the number of iterations that have been completed, I highlight the importance of the counter in the \textit{for loop}.

\begin{center}
\includegraphics[width=0.5\textwidth]{figure7.png}
\end{center}

\textit{Figure 7. The sketch that illustrates a for loop.}

**While loop**

To demonstrate a \textit{while loop}, I implemented the same design as the previous sketch but with a twist. Instead of there being a counter there is a bolt attached to a pencil by using a rubber band (Figure 8). This construction is used as a metaphor to show that the loop will start executing when the movement of the screw is initialized. A user will continue to complete laps with the ball until the movement of the screw has stopped and before starting a new lap a user must check so that the screw is still moving.

By using this metaphor, which uses the movement of the bolt as a sort of timer representing the loop’s ending condition, the loop will run and continue to execute as long as a specific condition is true. This allows users to develop a mental model that can help them understand the condition-driven nature of a \textit{while loop}.

\begin{center}
\includegraphics[width=0.5\textwidth]{figure8.png}
\end{center}

\textit{Figure 8. Illustration of the while loop.}
Nested loop

Nested loops can have multiple layers of loops which are nested. Due to the complex nature of a nested loop, I will focus on demonstrating a one-layer nested loop, meaning demonstrating the relationship between the outer loop (main loop) and a single inner loop. Within a nested loop, there can be different loop constructs such as a for loop, however, for this iteration I do not implement any specific loop constructs; rather the focus lies on developing a student’s understanding of the relationship between the outer loop and the inner loop.

To illustrate a nested loop through a metaphor, I created a sketch featuring two tracks that were interconnected, the tracks having two different sizes (Figure 9). To showcase the process of the loop execution, each track had its own ball which can be moved around. Before the ball on the bigger track has completed one lap the user must have completed laps on the smaller track. The mental model that I am trying to build for users is how a nested loop executes.

![Figure 9. The sketch that demonstrates a nested loop.](image)

6.3.4 Feedback from the user testing 2

All of the participants gave positive feedback in regards to them being able to “see” where in the loop process they were. However, with this lo-fi sketch, they had to manually push the balls throughout the user testing because of the rugosity of the foam used and the friction between it and the ball. Therefore, they could not pick up the sketch with both hands and twist and turn it to control the loop. One participant stated that “the idea is good but I should be able to move the ball by turning the sketch, pushing the ball around was inconvenient.” The same was stated by the other participants but using other words.

It became clear that, through engaging with the sketches, the participants could gain a clear understanding of how the for loop and while loop work and why they are different. It also came to light that all of the participants did experience difficulties understanding the difference between the two types of
loops when first learning about them. Therefore, using a distinctive way of demonstrating the concepts, to show clearly what the difference is, was well received by everyone. However, instead of using two different sketches, some of the participants felt that it would provide even more clarity if the concepts were demonstrated using the same board but still utilizing the same metaphors.

Though the participants felt like the nested loop sketch developed a sense of understanding of the concept, there were flaws in the design which could lead to confusion. First of all, though the idea of having two tracks interconnected may represent the basic relation between the outer loop and inner loop, it does not portray the execution flow accurately. As mentioned previously before the outer loop executes its next iteration the inner loop must have completed its entire cycle (Figure 10). However, when using one ball for each track the idea of a sequential execution is not clear.

![Flow chart of a nested loop's execution](image)

**Figure 10. Flow chart of a nested loop's execution.**

### 6.4 UP prototyping

After the second iteration of Sketching in Hardware the design concepts for the *for loop* and the *while loop* were advanced enough for me to start my prototyping phase where I would focus on refining the teaching tools. Even if the design concept for the *nested loop* was not up to par I decided to directly create a prototype that had a different design concept. This proved later to be a mistake. In retrospect, I should have continued creating sketches for the *nested loop* until I had reached a more developed design concept. Why this was a mistake will be explained in the next section.

#### 6.4.1 Prototype round 1

**For loop and While loop**

My aim for this first prototype round was to create a tool that allowed the users to control the process of the loops by simply twisting and turning the
object. I took this design approach as a response to the feedback I received after the second iteration of the sketches. Mainly, I focused on not having to push the ball around. Furthermore, since the participants thought that the for loop and while loop should be demonstrated on the same board, I created one single prototype that has the potential to illustrate both concepts by modifying its setup.

For the prototype which can be seen in figure Figure 11 I used a laser cutter to engrave forms in MDF. Then repeated the process with multiple layers of MDF, gluing them together, to create a track that has enough depth for a ball to move around by simply turning the prototype. There is a “cage” in the center of one side of the object that can be opened and closed so that the ball can roll out in the track when the loops are being executed, but also be put back in when the loops are completed. To make sure that the ball will not fall, there is an added layer of transparent plastic that functions as a “lid.” As previously mentioned, the design concepts for the two loops are still the same. For this prototype, there are holes in the corner of the board so that a user can put a stick in the different holes to count the current lap to demonstrate the for loop. For the while loop, the users use the same construction as in the corresponding sketch, which is a timer built by a bolt/pen/rubber band. However, the construct is not connected to the board itself, it is meant to be put on the surface close to the user. This is because the bolt would otherwise never stop moving if it was connected to the main board due to the user moving around while physically interacting with the tool.

![Figure 11. Picture of the for/while loop prototype.](image)

**Nested Loop**

As mentioned earlier, I made a mistake in the conceptualizing of this prototype. Figure 12 shows what I created, this was not a nested loop rather it was a loop with branches (implementing decision-making logic). The idea behind the design was that a user cannot move the ball through the main track, rather they have to move the ball through the smaller tracks. The original idea behind this design concept was to force a user to complete the iterations of the smaller track before they can complete the main track.
(represents the outer loop), to demonstrate the relationship between the outer loop and the inner loop. To enhance this further, I added sticks to block parts of the main track (Figure 13). However, this design concept is not a valid metaphor of the nested loop because when the ball enters one of the small tracks it cannot stay there to complete even one iteration. The design concept which I created is demonstrating how branching works in the sense that a user can move the ball through either the main track or the other tracks depending on which track is barricaded by the sticks. For example, if a stick is blocking the main track then it could represent the following statement: if the stick is placed on the main track then the ball must roll through the small track instead.

It was not until after user testing this prototype with all five participants that I realized that the design concept was providing an incorrect metaphor. However, this led to an interesting discovery. Although all of the participants fully understood the concept of nested loops, during user testing, nobody thought that the prototype did not demonstrate a nested loop. This suggests that while individuals may comprehend the functionality of nested loops, they may still encounter difficulties when it comes to accurately visualizing this concept.

![Figure 12. The prototype that incorrectly demonstrated a nested loop.](image)

![Figure 13. An enhanced picture of the "barricade".](image)
6.4.2 User testing of the prototype 1

The purpose of this test was only to explore if the tangibility and affordances of the prototype were effective for the participants to “control” the loop. While testing, all of the participants felt like the tangible aspect of the prototype offered a meaningful experience. One participant stated that “compared to the second iteration of the sketches, the interaction between me and this prototype is both intriguing but at the same time straightforward which allows me to fully focus on what I am supposed to learn from the experience.” Combined with the metaphors which were used for each one of the two loop concepts, it is clear from the user tests that the participants believe that the UP prototype can be used as a learning tool. However, there are still improvements that can be made, as one participant felt that it was difficult sometimes to exit the loop, stating “it is hard to control the speed of the ball at times, that is why I went past the exit part of the loop.”

As mentioned in the previous section, I did conduct user tests on the prototype which incorrectly represented the nested loop. Due to its flaws, I will not provide further insights from those user tests due to it being irrelevant. In section 6.5.3 it is further discussed what happened with my design exploration of the nested loops.

6.5 Tangible computing prototyping

After creating the first iteration of the UP prototypes I faced a dilemma. I could either continue to refine the UP prototypes or solve a problem that came to light during my interviews with the educators. Namely, the limitation that tangible tools present in regard to scalability. Daniel mentioned how providing enough resources in the form of tangible learning tools to a full course of students is a challenge, extent which was further confirmed by Benjamin by saying that there are not always enough resources at universities. Therefore, I decided to explore how it would be possible to make a tangible learning tool for loops that is accessible without the requirement of having to spend many resources to deploy it in the classrooms. Moving away from UP prototypes, created with only physical materials, I started to explore how it would be possible to solve the problem through tangible computing prototypes instead. It was during this part of my design where the following sub-question was created:

- How is it possible to create a tangible learning tool that can be provided to a large number of students but requires a low amount of resources?
6.5.1 Exploring tangible computing

While researching related designs for tangible computing devices within the realm of teaching programming, I found that the existing designs face the same problem with scalability as the UP prototypes. Even with the implementation of electronics each computing device had a limitation in regards to accessibility, requiring dozens of these devices to create a learning environment where each student can equally participate at once.

I had two aims while creating the tangible computing prototype, the first being how to solve the scalability problem? How can the tangible aspect of the UP prototypes be transferred over to the tangible computing prototype? This means that a user should be able to “control” the loop by rotating the prototype itself. The method that I found that would make it possible to fulfill both of the aims was to utilize a smartphone, a ubiquitous device present in the pockets of most - when not all- students at the university where the study was conducted.

Smartphones have an IMU sensor (Inertial Measurement Unit, a combination of gyroscope, accelerometer, and magnetometer) that precisely detects the orientation of the phone by means of filtering algorithms. This sensor provides data that represents the relative movement of the phone. The data can be accessed through code and then mapped to a visual object (UI element) on the smartphone’s screen, creating the possibility for a user to move around that object by rotating the phone. Comparing this to the UP prototypes that I created, this method provides the same aspect of tangible interaction as a way to control the ball. From a scalability perspective, this method also solves the problem related to the limitations of resources. This is because the only resource that the university needs to provide is the code since the majority of students use smartphones today.

6.5.2 Prototyping on smartphones

The first step of creating my tangible computing prototype was to figure out how to access an iPhone’s orientation data (for simplicity I chose to prototype on an iOS device). The ways of accessing the sensor data can vary depending on the operating system, for example, the steps required will be different between an iPhone and an Android. I used an iPhone to create my prototype and due to Apple’s security policy, you need to ask the user for permission to access the data due to it being sensitive information. This step can be done with code; however, it is important to note that the user needs to perform an action such as pressing a button on the UI to approve the notification that asks for permission to access the sensors. Due to the data being classified as sensitive information, the web page that hosts the UI representing the loops must reside on an HTTPS server. For this prototype, I used the platform CodePen3 to create the prototype due to it providing an HTTPS server for all

---
3 The website that hosted my digital prototype: https://codepen.io/
web pages created using their site. The codes that were create for the digital prototypes can be found in appendix 2.

6.5.3 Creating the digital prototypes

Following an unsuccessful attempt at accurately visualizing nested loops with an initial prototype, I made the conscious decision to halt its further development in the digital prototype phase. This decision stemmed from a prior mishap where I prematurely advanced to the prototyping stage without a successful sketch to guide the process. The absence of a clear and comprehensible sketch led to difficulties in visualizing the nested loop concept, thus resulting in an ineffective prototype. To avoid repeating this error, I resolved not to pursue the creation of a functional prototype without a successful preliminary sketch of the concept. Instead of rushing back into sketching, I channelled my efforts towards exploring how while loops and for loops could be advanced within the context of the elements they encompass. The objective is not to only enhance their interactive capability, but also to refine and evolve the underlying metaphors.

I have developed distinct digital prototypes to visually represent different loop constructs: one for the for loop and two for the while loop. Each prototype of the while loop delineates a unique aspect; one prototype underscores the inner condition, while the other illuminates the outer condition of the while loop.

While the prototypes are unique in their representation, they all share a common foundational structure. This shared base provides a unified and coherent starting point, onto which additional elements are grafted to form each unique prototype. As these elements are added, they imbue the prototypes with distinctive attributes that accurately depict the concepts of different loops. This approach ensures that despite their differences, all prototypes retain a basic level of consistency and familiarity, which simplifies the task of understanding and differentiating between the various loop constructs.

The foundational structure

Figure 14 illustrates the foundational structure of the digital prototypes and can be describe as the following: a hollow circle with another circle nested within it. A ball, stationed at a specified starting point which is the line in the top of the structure. The users are given control over this ball by first activating the 'start the loop' button and then tilting their phones to move it around. To maintain the ball within the boundaries of the two circles, collision detection logic has been employed. When the loop has been completed the user will receive an alert as demonstrated in Figure 15.
A full circuit of the ball around the circle, returning to its initial starting point, signifies one complete iteration of the loop. Enhancing this foundational structure are three distinctive objects situated outside the circular structure. These objects, each bearing a unique shape, symbolize different types of commands, highlighting the fact that loops are not void constructs; they contain and process data through commands once the loop is activated.

Users are given the autonomy to drag these objects into the circular structure and arrange them as they please, demonstrating that data can be processed in various sequences of commands within a loop, rather than following a rigid, predetermined order. A special feature has been added to these objects, where in they change color when the ball collides with them, signifying the command execution. This interactive change not only adds an element of engagement for the users but also visually reinforces the purpose and function of loops in running tasks.

![Figure 14. The foundational structure of the digital prototypes.](image)
Figure 15. The alert message will display upon the completion of the loop. However, the specific name displayed in the alert will vary depending on whether it’s a ‘while’ loop or a ‘for’ loop.

**For loop**

The core illustration of the *for* loop utilizes similar metaphors that were used in the physical version. Furthermore, it has the same foundational structure mentioned in the previous section but with the added counter as an condition. For each complete circuit of the ball around the loop—signifying one full iteration—causes the loop counter titled "Laps" to increment by one. This real-time loop counter is a visual representation of the number of iterations completed within the *for* loop (Figure 16).

Crucially, the *for* loop embodies a predetermined number of iterations. Therefore, in this prototype, once the *for* loop has executed three iterations, denoted by the counter reading 3/3, the loop is considered complete. At this juncture, an alert pops up on the screen notifying the user that the loop has been successfully completed. This immediate feedback mechanism provides an interactive way for users to learn and understand the concept of *for* loop and its defined iteration property, significantly enhancing the learning experience.
**Figure 16. Demonstration of a ‘for’ loop**

**While loop (inner condition)**

To effectively illustrate the inner condition of a *while loop*, a unique auditory feature has been incorporated into the digital prototype. Each of the objects within the loop structure has been associated with a specific sound, as the ball collides with an object, its corresponding sound is triggered. The execution of the *while loop* continues until a particular melody is formed. This melody is generated by a specific sequence of sounds, which correspond to the ball colliding with the objects within the loop structure. Once the melody has been played, the loop continues with its iterations until the next check of the condition is due. If the melody has been completed, an alert appears when the ball touches the starting line again, signifying the end of the loop's execution.

The design of this prototype allows the ball to collide with each object twice, resulting in a total of six sounds to compose the desired melody. To aid the users in tracking the number of sounds played, a “melody counter” has been included in the prototype (Figure 17). This counter increments by one each time the ball collides with an object, offering a visual confirmation of the auditory sequence. This innovative approach not only clarifies the concept of inner conditions in *while loops* but also provides an engaging, multi-sensory learning experience for users.
While loop (outer condition)

To elucidate the concept of an outer condition in a while loop, a timer feature has been incorporated into the corresponding digital prototype (Figure 18). Positioned outside the loop structure on the screen, the timer activates when the "start the loop" button is pressed and begins counting down from ten. As the timer runs, the user is free to continue iterating the loop with the moving ball. When the countdown reaches zero, the user can still continue the current iteration of the loop, guiding the ball until it completes its current circuit.

Upon completion of the circuit and touching the line, the loop is deemed complete and the alert pops up to confirm this. The timer reaching zero causes the loop to terminate, as the condition associated with the timer (the countdown) is now false. This digital prototype thus effectively illustrates how an outer condition in a while loop can determine the termination of the loop, providing a visual and interactive demonstration of this fundamental programming concept.
6.5.4 Feedback from the user testing

From the user tests it became evident that the tangible interaction provided by the phone's IMU sensor was well received. Additionally, the use of a phone as the tangible tool proved to be comfortable for users, providing them with a familiar and accessible medium to interact with and learn from (Figure 19). Participants found the ball control intuitive and engaging, giving positive feedback on how the prototypes leveraged the smartphone's existing technology to create an immersive learning experience.

![Figure 19. Picture taken of a user tester trying the prototype.](image)

Two of the participants, however, pointed out some difficulties in understanding the purpose of the objects representing commands in the loops. Although they could grasp the physical interaction, the metaphor of processing commands when the ball collides with the objects wasn't immediately clear to them. This suggests that while the tangible interaction is a strength of these prototypes, the representation of abstract concepts such as command execution might need further clarification or more explicit signalling.

The demonstration of the three different loop concepts were proven to be effective. Participants found that both the for loop and the while loop (outer condition) prototypes effectively demonstrated their respective concepts in a clear, unambiguous manner, fostering a deeper understanding without inducing any confusion. The auditory feature in the while loop (inner condition) prototype was in general well received by participants. They found the creation of a melody as the loop's condition to be a creative and engaging teaching approach. However, although the loop's technical execution remains unaffected, user-driven variations in the ball's speed and the arrangement of the sound-emitting objects create considerable inconsistencies in the produced melody. This resulted in there not being one standard melody that could lead to some confusion at first but the participants still understood the concept.
With the completion of the last user tests, we conclude an intensive phase of creation. During this phase, I have thoughtfully developed and refined a multitude of tools with the objective of simplifying the concept of loops for novice programmers. This process included design, thorough testing, and the incorporation of insightful user feedback, leading to multiple iterations of prototypes. Figure 20 offers a visual representation of this creative journey, outlining the creations that took place in this project.

Figure 20. Diagram of the creations.
7 Discussion

In this chapter I delve into a comprehensive discussion regarding the tangible learning tools I developed during this thesis project. I evaluate their role in filling gaps in the current educational landscape, examining their strengths, limitations, and potential applications. Finally, I highlight potential future work, which includes expanding the research, adapting the tools for various programming concepts, and incorporating collaboration elements to further enhance the learning process.

7.1 Filling the gap

The tangible tools that were created in this thesis address the gaps in the existing landscape of tangible learning tools for the specific case of understanding loops. Other existing tools used are commonly designed so that users are indirectly engaging with loops which indicates that the development of mental models is not a priority. There is an absence of tangible tools that are designed with the sole purpose to promote active engagement between the user and loops. Though there do exist tangible tools such as Code Jumper, this tool is developed for children which results in the complexity of what is being taught is not suited for university students.

The tangible tools created in this thesis offer a novel approach for students to develop an understanding of how loops work. Through information collected through interviewing experts in the field, the tools were tailored to students at a university level. User testing showed how the tangible aspects of the tools combined with the metaphors the artifacts convey, a user can develop an understanding of the abstract concept of loops by “taking” it out of the computer and allowing users to physically “control” its execution, making it concrete. Furthermore, I implemented metaphors that are significantly different from each other to clarify the difference between loop types.

Within the existing landscape of tangible computing tools and UP tools, there is a recognized limitation concerning accessibility. Often, the physical nature of these tangible tools is perceived to present a scalability issue, inhibiting widespread implementation. Nevertheless, the digital computing prototype developed within this thesis project suggests a viable solution to this problem. This solution capitalizes on the use of smartphone sensor data, specifically focusing on device orientation. By creating a code that enables access to an iPhone’s sensor data (or any other operating system), it becomes feasible to link this data to a virtual object on the phone's screen. Through further coding, additional UI elements can be generated on the screen to represent different loop concepts. This enables a user to maneuver the virtual object amidst these elements, thereby providing an active learning experience. From the feedback of the user testers, this experience, which combines tangible interaction with metaphorical representations, allows learners to
develop robust mental models of how different loop concepts work. Thus, answering the research question: "How can interactive tangible prototypes help university students develop their understanding of how abstract programming structures -such as loops- work?".

7.2 Comparing UP and tangible computing

Through user testing, it is clear that the two different tangible teaching approaches are efficient in developing students’ understanding of how different loop concepts work, however, each has its strengths and weaknesses. The tangible computing prototype is a far more accessible teaching tool when compared to the UP prototypes. The resources required to create the digital prototype are a smartphone and code that creates the user interface. A university interested in using these teaching tools only has to replicate the code, since the usage of smartphones is high among students. However, the UP prototypes which only have physical elements require the university to pay for the material and labor costs to create the tools. This is a problem from a scalability perspective as mentioned by the educators I interviewed. There might not exist the resources it requires to provide, for example, 100 students with enough physical tools to create a learning environment where all students can actively be engaged.

Tangible computing prototypes offer more freedom to create changes to the design concepts compared to UP prototypes. If a lecturer wants to make certain tweaks to a design concept, they only have to change the code of the computing prototype. However, if a change needs to be made for a UP prototype there needs to be a change to the construction which could require creating a new version of the prototype. If there is the need to provide a large number of students with the teaching tools it would require a lot of resources to develop new prototypes. Furthermore, by employing the capabilities inherent to digital technology, we can enrich the demonstration of loop constructs far beyond what physical prototypes can offer. This introduces a layer of complexity that can captivate users' interest longer, combating the quick waning of engagement often seen with simpler, physical demonstrations.

One limitation that the digital prototype presents is the available space for teaching content, due to the screen size of smartphones. Though certain designs such as the ones I created can be implemented on a smartphone screen, more complex examples can be difficult to demonstrate. This is because the size of a smartphone screen is not large enough to demonstrate a concept that has a lot of different elements. However, a UP prototype offers an element of flexibility in regard to the size of the construction. This allows for the opportunity to develop prototypes that can demonstrate more complex concepts.
7.3 Metaphors and mental models

It is crucial to acknowledge that while we have the capacity to help develop a functional mental model with regards to -in this case- programming through these prototypes and metaphors, measuring this impact is more complex. I recognize that my work has the potential to influence a user's understanding of loops, but quantifying this change is challenging and requires venturing other fields such as studies in learning and teaching, or psychology of learning. Instead, the focus remains on creating and refining these metaphors, testing them with users, and gauging their level of understanding. This iterative approach can provide valuable insights into the metaphors' effectiveness and guide subsequent design decisions.

While the development and utilization of metaphors has been a crucial part of the prototypes, this inevitably opens up for further discussion. For instance, how complex should we make the metaphor to adequately describe a loop? Where do we draw the line? Furthermore, what is the optimal environment for such explorations: a scripting language like JavaScript or a game engine designed for more sophisticated interactions? An open question that also arises is: how do we effectively translate these physical metaphors back into actual code?

7.4 Constructivism

The prototypes which I created did take upon a constructivist approach in regards to learning, as mentioned by CSUnplugged (2015) this tends to occur while working with UP. However, I did not mention in my methods section that I would use such an approach in my design process. This is because while creating the prototypes the key components of the creations such as being hands-on interactive tools to construct knowledge, aligned with the principles of constructivism. Therefore it is worth reflecting on why a constructivist approach is more efficient than a constructionist one when creating a tangible learning tool in the context of this thesis.

Following Papert’s constructivist vision of education, students should create their own tools to achieve the representation of loops and while they could reach a valid metaphor, they could as well reach and invalid one, which would render the learning goals for this project useless. Therefore it is a more suited approach to teach novice programmers how loops work by letting them interact with a tool to help them construct a correct mental model of how loops work. As discussed previously the participants of the user tests who are mentioned as beginner programmers, did not notice that the UP version of the nested loop prototype was an incorrect representation. If novice programmers who are less experienced than the participants in this thesis, were to use a constructionist approach to learning, there is the inherent risk of them creating a prototype that is an incorrect representation of the concept. This would lead to them developing an incorrect mental model of
the concept, just as the participants of this thesis did while interacting with the prototype that was incorrectly constructed by me.

Considering the work of Papert (1980), it is important to note that the variability in prior exposure to computational thinking amongst university students. For those who have been introduced to concepts such as loops from an early age, their existing understanding could lessen the impact of the tangible learning tools developed in this study. These students, having already traversed the terrain of loops, may not find their understanding significantly enhanced via the constructivist approach employed here.

During the project, an intention to introduce an aspect of social constructivism emerged after interviews with educators highlighted the benefits of collaboration. However, due to the prioritization of addressing the tangible tools scalability issues, there was insufficient time to incorporate collaboration features into the final prototypes.

### 7.5 Future work

For a more thorough evaluation of the tangible tools developed in this thesis, several avenues could be pursued. Firstly, their effectiveness should be validated in a genuine educational setting. This setting would present a diverse range of student abilities and varying classroom dynamics, offering a more comprehensive view of the tool’s functionality. Evaluating student comprehension before and after tool use, observing teacher-student and student-student interactions with the tool, and soliciting teacher feedback could all offer valuable insights.

Secondly, a dedicated assessment with novice programmers could yield additional data. Adjustments may be necessary to ensure the tool can cater to that specific user group. Further, a longitudinal study observing how novices progress with continuous use of the tools could provide richer understanding of its long-term effectiveness.

Furthermore, while the current tool focuses on loop concepts, programming education extends far beyond that. Future design efforts could focus on introducing other control structures, conditional statements, or even the nuances of function calls and recursion. The current tool could serve as a base to develop these future iterations, creating a suite of educational aids catering to different levels and topics of programming.

Lastly, collaboration has been highlighted as a crucial component of effective learning by the teachers interviewed for this thesis. Therefore, future designs should aim to facilitate this. One could imagine a setup where students work in pairs or small groups, each with their unique role in manoeuvring the digital object through the loops.
8 Conclusion

The guiding research question in this thesis was, "How can interactive tangible prototypes help university students develop their understanding of how abstract programming structures, such as loops, work?" In addressing this, I introduced an innovative approach to bridge a gap in the realm of tangible learning tools for programming loops, thus contributing to the field of interaction design in three significant ways.

First, I developed a tangible tool that offers a hands-on, immersive learning experience for university students. This tool, shaped by insights from expert interviews and user tests, effectively translates abstract programming concepts into a tangible format, enhancing students' understanding of certain loop concepts.

Second, I brought the crucial aspect of scalability to the forefront of the design process, enabling the creation of scalable, adaptable and future-proof designs. This approach led to the development of a tangible computing prototype, which overcame the typical scalability constraint of the existing tangible teaching tools that were explored in the research phase of this project.

Lastly, the significance of physical models in inspiring software design was highlighted. The tangible, real-world elements embodied in the physical prototypes created during this project served as a source of inspiration for the design of the subsequent digital prototypes. This process underscored the potential of tangible elements in creating intuitive and user-friendly software, highlighting an innovative approach where physical design elements are effectively translated into the digital design sphere.

Future work has been identified in several areas, including the need for validation of these tools in an authentic educational setting, statistical assessment of their effectiveness with novice programmers, extension of the tool's scope to cover more programming constructs, and the incorporation of elements in the design to allow for student collaboration.
9 References


Buxton, B. (2010). *Sketching user experiences: getting the design right and the right design*. Morgan kaufmann.


Ma, L. (2007). Investigating and improving novice programmers' mental models of programming concepts (Doctoral dissertation, University of Strathclyde). Retrieved from [https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3c8efb0c95325ac2f6bb38bd3d56fde9004892](https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3c8efb0c95325ac2f6bb38bd3d56fde9004892)


Van Dijk, J. A. (2006). Digital divide research, achievements and shortcomings. *Poetics, 34*(4-5), 221-235. [https://doi.org/10.1016/j.poetic.2006.05.004](https://doi.org/10.1016/j.poetic.2006.05.004)


Wilson, V. (2012). Research methods: interviews. *Evidence Based Library and Information Practice, 7*(2), 96-98. [https://doi.org/10.18438/B89P5B](https://doi.org/10.18438/B89P5B)


Appendices

1. Consent form

I have been verbally informed about the study and read the accompanying written information. I am aware that my participation is voluntary, and that I, at any time and without explanation, can withdraw my participation. The person/s leading the study will strive to guarantee confidentiality in that no unauthorized person may have access to the material. The gathered material will be stored properly and used for research purposes only.

I hereby submit my consent to participate in the above survey:

Date: ……………………………………………………………………………..

Participant’s signature: ………………………………………………………

2. The code

For loop: https://codepen.io/IdxThomas/pen/BaqvxWV
While loop (inner condition):
https://codepen.io/IdxThomas/pen/abQbXLG

While loop (outer condition):
https://codepen.io/IdxThomas/pen/rNqRBxx