Experience Programming: an exploration of hybrid tangible-virtual block based programming interaction

Clarissa Correa de Oliveira
“The search for new integrative disciplines to complement the arts and sciences has become one of the central themes of intellectual and practical life in the twentieth century. Without integrative disciplines of understanding, communication, and action, there is little hope of sensibly extending knowledge beyond the library or laboratory in order to serve the purpose of enriching human life.”

(Buchanan, 1992, p. 6)
ABSTRACT

In less than a century, programming languages have assumed many forms in adapting to system’s needs and capacities, of which our cognitive systems are a part. One variation, tailored specifically for the cognitive processes in children’s education of computational concepts, and nowadays successful among novice adult learners too, is that of visual block based programming. From the pool of available block based programming environments, Scratch is the most popular for users, and therefore becomes a good topic for researchers interested in contemporary educational discussions, including that of coding as a curricular activity in schools. Although inspired by the educational philosophy of using abstract physical blocks in foundational learning, the mainly visual interface of Scratch is made for keyboard and mouse mediated interaction with the digital content on-screen, producing audio-visual feedback. This research is a case study of Scratch, where the shortcomings found in interactions with its environment motivate the investigation of a potential hybrid technology – tangible and visual – for enhanced learning of foundational concepts in block based programming. The investigation is characterized by progressive cycles of conceptual design, supported by prototyping and testing. The results from its design process present the benefits and challenges of this hybrid concept to inform and inspire the development of new technologies for learning, as well as it should inspire designers of Tangible User Interfaces (TUIs) for learning and the educational community of computing to challenge the current ways of learning. The work here presented is concerned with acknowledging and building onto strengths of existing technologies, rather than substituting them with disruptive ideas.
ACKNOWLEDGEMENTS

First and foremost, I wish to thank Malmö University for the talented personnel, state-of-the-art tools, and incredible support structure provided to the students. And I would like to give a special thanks to Professor Per Linde who supervised me through the course of piecing together this thesis: Thank you, Per! You were inspirational. You allowed me to understand the investigatory process of Interaction Design research as a craft unique to the development of each individual project, revealed in – not ruled by – the use of known methods and tools.

I am immensely grateful to the knowledgeable and experienced contributions that my research benefited from, specifically from Professor Daniel Spikol of the IOTAP Lab of Malmö University, IT Strategist Per Falk of Väsby Makerspace, Coder Dojo Malmö, and Professor Francisco Lopez Luro of the Department of Creative Technologies (DIKR) of Blekinge Tekniska Högskola. I also thank Andy Wood, Martha Sedgewick, Ian Woodbridge, Simon Belshaw, John Stout and his students, Catherine Elliott, Dr. Sue Sentance, and the Computing At School (CAS) Exeter Hub. All those mentioned shared a willingness to participate in this study in one form or another, with curiosity for the unusual topic and its potential outcomes, making it possible for me to advance with every step of the project.

Last, but not least, to my family, and to my dear Rick Latham Lechowick, Thiago Ferreira, Olga Matsarina and Kevin Ong: you cared for me in different moments of this adventure in the way that only the greatest of friendships can do, and this was probably more important to me than you might think. Thank you!
DEFINITIONS OF KEY TERMS AND CONCEPTS

Artefact – “may be a commercially released product or any end-result of a design activity such as a concept system developed for research purposes” (Houde & Hill, 1997, p. 3)

Block – “A compact usually solid piece of substantial material especially when worked or altered to serve a particular purpose”, according to Merriam-Webster online dictionary. (Merriam-Webster, 2018)

Block-based coding - is coding within a programming language where instructions are mainly represented as blocks. Scratch is a block-based programming language. (Lifelong Kindergarten Group, 2018)

Coding Literacy - the ability to identify, understand, interpret, create, communicate, and use rules that shape and reshape information in order to participate fully in the creation of new information. (Betts, 2011)

Computational Thinking: refers to a pool of abilities necessary to abstract away a problem, and represent it in a format amenable for computers to process (Bonani, et al., 2018, p. 66).

Experience – a personal process or practice based on, or formative of individual knowledge. “The fact or state of having been affected by or gained knowledge through direct observation or participation”. (Merriam-Webster, 2018)

Inclusive education - a process of strengthening the capacity of an education system to reach out to all learners. It is, therefore, an overall principle that should guide all educational policies and practices, starting from the belief that education is a fundamental human right and the foundation for a more just society. (UNESCO: International Bureau of Education, 2009, p. 9)

Manipulatives - “Objects (such as blocks) that a student is instructed to use in a way that teaches or reinforces a lesson”, according to Merriam-Webster online dictionary. (Merriam-Webster, 2018)

Tangible - capable of being perceived especially by the sense of touch: palpable; substantially real: material, according to Merriam-Webster online

Tangible Multimedia – term suggested by Tsong, et al. (2012), referring to multimedia augmented by physical objects.

Visual Programming Language (VPL) - is any programming language that lets users create programs by manipulating program elements graphically rather than by specifying them textually.
Wizard of Oz - Wizard of Oz prototypes are prototypes with faked functions — for instance, interactivity that comes from a human rather than an algorithm or software code, with users believing the latter is the case — that you can use to test with your users. (Dam & Siang, 2018)
1. INTRODUCTION

This is the materialization of a project that started in my first year of this education. The project then, in 2017, consisted of validating my claim that “the choice of Scratch as the main resource used in primary schools to teach the fundamental logic of programming presents a potential threat towards human rights to equal opportunities in education faced by blind students, for the use of Scratch seems to rely highly on visual abilities.” (De Oliveira, 2017, p. 5) In doing so, one of the activities carried out during the exploration phase was the test of Scratch by a blind participant. Immediately, it became clear that this was not accessible to the blind, in any way whatsoever.

Have we, really, made the best use of our technological resources – knowledge, skills and materials – to develop tools that enhance learning of subjects such as computer programming? Does the fact that Scratch is not accessible to a blind person have something to say to us about the limitations imposed by graphical content and the senses that it fails to engage, which could be of relevance for blind and non-blind learners?

When I say that this project materializes the former, that of 2017, I mean to say that what I am developing now has a more tangible outcome than before, because it offers us a concept in the form of a prototype, whereas previously my focus was on methods for producing design guidelines. There is a both a link and a differentiation between these research projects. The link is that of conducting research in Interaction Design for better educational practices, in order to facilitate the dissemination of computing concepts and skills. They differ, especially, in their target user groups, their methods and their intended outcomes. The previous work aimed at blind children and proposed participatory design research methods to generate a set of guidelines for designing programming tools for the blind. The current work has a broader audience of school aged novice learners of computer programming, and has prototyping as a method of exploration of tangible user interface (TUI) design for interaction with graphic block based programming languages to evaluate the possible learning benefits of an integrated educational system in the apprehension of basic programming concepts.

What follows is a compendium of the current research that comprises the context of this investigation, followed by my attempt to provide a fundamental overview of the key domains that inform my design exploration: computing education, the case of Scratch as a popular choice of block based programming language, and tangible user interfaces and learning. Information on these three domains have guided my exploration of the benefits of a hybrid tangible-visual approach for learning programming, building on the shortcomings of Scratch as design opportunities with the potential for improvement beyond mere software design, explored and evaluated through cycles of prototyping.
1.1 Problem Statement

It is without a doubt, that the dissemination of knowledge and skills in computer programming through social spheres not long ago seen as ‘too young’, or ‘not of the right niche’ (as so called “computer geeks”), is no mere trend, but an opportunity to better enjoy the benefits of the digital world by understanding how it works and why it is useful today. Be it for making personal creative projects feasible, or for gaining that extra skill for better job opportunities, or other relevant reasons as such. Big (visionary) steps, such as that of introducing coding in schools as part of their curriculums, show contextual understanding of the changing landscape of education in our contemporary world - a world permeated by pervasive technologies. In this scenario, it is important to acknowledge that the tools developed for such education are brilliant in the logical, creative and interpersonal learnings that they leverage. These are tools such as Scratch\(^1\), Micro:bit\(^2\) and HelloRuby\(^3\), so different from one another in terms of approach, yet similar in their purpose of teaching kids computational concepts, and in their efforts to become accessible to many, through affordable prices, user-friendly interfaces, and widespread availability of support and learning material.

In general, the first steps in the course of learning foundational block based programming languages follow a progressive ‘concepts first, electronics later’ structure. Loops are the first taught. Then Events, with If/Then Statements (which need a loop to run on Scratch, for example). Thirdly, data handling through Variables. From here, it continues progressing into complexity, always relying heavily on the visual aspects of programming, from development to results.

Some experts in education argue that we should forget about visual programming, and teach text based coding from the beginning. Although both types of programming environments are suitable for teaching, block based was developed specifically for learners new to the world of coding, while textual environments were around in the computer sciences way before we even considered that diverse communities, such as those of special needs and of primary schoolers, could gain the skills. The nature of block based languages is inclusive in itself by facilitating syntax construction, focusing the learning in the giving and receiving of instructions rather than a perfect structure in a line of code. In this sense, children with learning disabilities, such as Dyslexia, have better chances of participating in the learning.

\(^1\) As described in its official website, Scratch is “a free programming language and online community where you can create your own interactive stories, games, and animations”. Source: https://scratch.mit.edu/ (access on April 29, 2018)

\(^2\) As described in its official website, Micro:bit is “a tiny programmable computer, designed to make learning and teaching easy and fun.” Source: http://microbit.org/ (access on April 29, 2018)

\(^3\) As described in its official website, Hello Ruby is “the world’s most whimsical way to learn about computers, technology and programming. The story started with a book, and now Ruby continues her adventures in exercises, games and apps”. Source: http://www.helloruby.com/ (access on April 29, 2018)
However, it is important to highlight that outside of a classroom setting, as a self-taught activity, programming in both block based and text based languages have been reportedly unsuccessful in sustaining people’s interest. As I present in the subchapter *Tangible user interfaces and learning*, this lack of enjoyment and engagement might be mitigated by introducing tangible interaction in the use of computer environments – a problem of design.

With foundational programming educational tools either solely tangible or visual as available today, the solution may lie somewhere in between, where virtual worlds and matter meet, to augment the purpose and the potential of programming in a learner’s life. This might be a case of Universal Design, if defined as design that intends to be accessible to all, not just culturally or geographically, but also in terms of usability. Inspired by this movement towards providing programming education for all, I dedicate this study to the development of more resilient educational tools for adaptation to diverse contexts.

### 1.2 Research Question and Aims

Having presented the roots of this project, I do not wish to condemn the visual aspect of these programming environments, but to understand how tangible interaction can collaborate with them to enhance knowledge retention. I wish to rethink some of its visual components and imprecise mouse-mediated interactions.

In doing so, the main aim is to attempt to answer the following question: **can we introduce manipulatives in the interaction with visual block based programming environments to enhance the learning experience at a foundational level and to improve accessibility? If so, then how and what are the implications for interaction design?**

The main aims of this study are to:

1. Provoke reflective thinking upon the practice of Interaction Design for solving complex interface problems, moving across disciplines, to cause positive impact within the social sphere, thus improving lives.
2. Explore and propose a tangible alternative (in the form of a concept) to visual block based programming languages, using Scratch as a case study for experimentation through prototyping and testing.
3. Evaluate my proposals by engaging target users in proof-of-concept activities.
4. Share my findings with the Interaction Design research community, extended to all other research communities and practitioners interested in improving design for learning, especially for computing education.
The purpose of this study does not entail:

a) Developing the technological solutions to my proposed design. Questions, such as what the cost is, are not answered, although some suggestions on how to implement it are made.

b) The production of a high fidelity prototype.

c) The substitution of existing block based programming environments with a completely different interface. The focus is on working in favour of the existing systems that are already successful in learning and accessibility (geographically speaking) to considerable extents, appropriating the potential of shortcomings as opportunities for design exploration.

d) The participation of children in experiments, because the meta learning aspect of this investigation is better assessed by educators who are able to evaluate a design idea critically during prototype sessions, than by students immersed in the learning themselves.

e) Targeting special education specifically, understanding that although special education has played its part in motivating this study, the focus is on building on such shortcomings as the accessibility of Scratch, to explore how tangibility can improve learning in this specific domain of foundational computing education.

f) The improvement of the learning per se, although the tendency is that adding a tangible layer to a visual system will, invariably, incur positive impact.

Shortcomings in the accessibility of Scratch are therefore the case study for this investigation. Further exploration of tangibility is then built onto Scratch itself to allow for comparative studies between its current and preferred (preferred according to my findings) state. Scratch is also a good programming tool to conduct research on due to its popularity that enables my access to a wide range of resources (literature, projects and people).

The target audience of this study is designers of Tangible User Interfaces (TUIs) for learning and the computing education community. This audience also extends to, more indirectly and not limited to, the special educational needs and disabilities (SEND) community, and industries of digital innovation interested in developing new media.

The target user group of a final product based on this study is that of learners of foundational concepts in programming who are transitioning from Tangible Programming Languages (TPLs) to Block Based Programming Languages like Scratch. These are secondary school learners in general, who are better prepared to make sense of abstract concepts, but can benefit from the help of a TUI to fully understand them.
Even though this study is aimed at learners as final users of its possible future materialization into available instructional products, it focuses on the experience of educators within the computer programming community to provide insights and evaluations of the design proposals here developed. The reason for this choice lies on the belief that educators have better experience in reflecting on ‘how students learn’. By metacognition – the process of thinking about thinking -, they are better used to clearly and orderly reporting their conclusions from experiencing learning, due to their instructional mind-set, than young learners are. It is, in fact, difficult for students to assess meta learning in the midst of learning. In addition, the need of high fidelity prototype testing, when testing with end users, reinforces my choice of participants in this design process. The need to produce such prototype could cause prejudice in this study by: taming its investigative potential outcomes, by dismissing promising ideas that present a high complexity level to fully produce in the form of a product; or by taming its experimental nature in face of the need to provide very fine design specifications for building at a fully working prototype; shifting its focus from an interaction design study to a product design process, due to the previous first two reasons mentioned.

2. BACKGROUND

The context presented is at the intersection of Computing Education and Tangible User Interfaces and learning. These topics have been extensively discussed by others, therefore there is much to learn from their experiences, previously to developing new ideas of my own.

Reviewing how Computing Education has developed over time, why it is relevant and what the challenges are helps to understand the purpose of learning programming, as well as where we stand today with visual block based environments at the forefront of this domain. To take a closer look into this context, I also present a detailed review of Scratch, breaking it down into component features and interactions, and identifying its shortcomings which will then be the main conducts of design exploration into tangibility. In addition, Tangible User Interfaces (TUIs) in the context of education, is another topic at the core of this research.

To introduce this chapter, it made sense to review the study from 2017 that led me to here: my own dissertation on blind programming. It is somehow difficult to divide all of this background information into specific categories, due to overlap of congruent topics, but I hope that the main sections in which I have divided this will lead to a systematic read, therefore also an easier one.
2.1 Building on previous research

To refresh the context: in 2017, I conducted experiments with blind users of computers, where I tested the accessibility of Scratch to understand the limits that its interface presents to this group. I also interviewed a blind primary schooler about the experience of being blind in a regular Computer Technology classroom, to gain further validation of the existence of weaknesses regarding visual block-based programming environments, and in fact, this student confirmed the inability to participate in related learning activities. The student’s capability of learning computational concepts and applying them was in alignment with that of other non-blind students, not only creatively, but also flawlessly performative of instruction-based activities with tangible objects. The fact that learning, in this case, was inaccessible solely due to the technology of choice for interacting with these environments, sheds light onto the need to improve such visual programming tools, in detriment of making better use of what state-of-the-art technology can do for educational practice. Both previous and current researches are concerned with improving the outreach of these tools, and relate improvement to the intrinsic embodiment quality of tangible interaction, proven beneficial for learning.

The former dissertation contributes to the latter with some insights. These insights touch upon aspects of tangible manipulation in learning foundational programming, based on “instructions given in audio formats, performed physically with the help of physical objects” (De Oliveira, 2017, p. 34). They include the ideas that: time for accomplishing a task in a tangible programming format may be longer (De Oliveira, 2017, p. 34); “when thinking spatially, elements that are not at a ‘reach’ distance automatically become secondary or unimportant” (De Oliveira, 2017, p. 35); “logical skills are important for understanding how the world works and that things happen through logic, not through magic” (De Oliveira, 2017, p. 36); and that a collaborative approach to learning programming is a way to build common shared understanding in learning programming (De Oliveira, 2017, pp. 36-37).

2.2 Computing Education

Most discussions on the topic of Computing Education inevitably targets mostly school aged learners, therefore end users to my study are likely to be of this same age group and condition. According to Sue Sentance, who is a senior lecturer in computer science education, Computing Education will enable children to be better prepared to face ethical challenges, such as Cybersecurity and Artificial Intelligence (Sentance, 2018). Sentance is based in Britain, the first amongst European countries, to introduce coding in its national education.

---

4 Sue Sentance lectures in the School of Education, Communication and Society at King's College London, and was recently awarded the BERA/SAGE Public Engagement and Impact award for her work in computing education with teachers. (Sentance, 2018)
educational curriculum, in 2014 (Dredge, 2014), followed by other nations such as Estonia, Sweden, Finland, and others (School Education Gateway, 2015).

I believe that not only does this education prepare for ethical challenges; it also gives these future adults the knowledge and skills necessary to make better choices and use of technology in general. I see this as a sort of freedom, an advantage that kids have over the previous generation, to make informed decisions as to how they interact with technology, due to better knowledge and experience. This advantage extends to their professional lives, in that they can have more freedom to compete for technology related jobs, if they wish to.

Karen Brennan, associate professor in the Harvard School of Education, says that learning computing requires “creating – writing code, designing programs, and building Systems –” and connecting – “sharing “learning experiences with others: hacking with peers, getting advice from instructors and teaching assistants, and drawing upon the expertise of colleagues near and far” (Brennan, 2013, p. 52). Creating and communicating computing knowledge and experience, can be performed from an enclosure such as a home, with communication through online community forums, hackathons, workshops, as well as in live hackathons, workshops, meetups, and other social events. Such events were once particular to experienced computer scientists, now attended by makers of all levels, interests, background and ages.

The ways of learning are constantly changing, and with computing education it is no different, and as pointed out by Alan Bundy, Professor at the school of Informatics at the University of Edinburgh, the new questions in this field are more profoundly addressed in less direct approaches than regular direct ones (Bundy, 2007). There is no better way to illustrate this, other than with Seyomur Papert’s educational philosophy derived from the Brazilian ‘samba school’ system, which inspired his creation of Scratch. Papert, in his article Some Poetic and Social Criteria for Education Design (1976), explains that in the case of a Samba School, people gather because of one common interest: samba. Denominated “School”, as a literal translation from the Portuguese Escola, in reality, these are samba associations, or in other words clubs. Papert’s inspiration from a trip to Rio de Janeiro is as he described it:

“[…] the Samba School has another purpose then the fun of the particular evening. This purpose is related to the famous Carnival […] at which each Samba School will take on a segment of the more than twenty-four hour long procession of street dancing. This segment will be an elaborately prepared, decorated and choreographed presentation of a story, typically a folk tale rewritten with lyrics, music and dance newly composed during the previous year. So we see the complex functions of the Samba School. While people have come to dance, they are simultaneously participating
As he described it, the value to be acknowledged for Instructional Design, is the formation of small groups, with no specific age range, level or format of instruction, duration, spatial configuration, to which he adds “the details are not important: what counts is the weaving of education into the larger, richer cultural-social experience of the Samba School” (Papert, 1976). In his case, he transferred the positive qualities of this learning collected in Rio, into traditional learning contexts of subjects such as maths. Papert suggests, and I agree, that the rigidity in traditional education is understandable from the perspective of the traditional technology used in schools, limited to pen and paper. This is why I argue that we must reconsider how we are teaching foundational programming at large (the worldwide scale): other than implementation barriers such as financing, nothing prevents designers, educators and software developers, from exploring improvements for providing learners the best experiences and outcomes. We do not have as many technological barriers in present times, neither have we many restrictions in gaining access to information regarding technological innovation.

The creation of Logo\(^5\), in which Papert actively participated, as a programming language accessible to children, is the biggest proof of his vision of an education that adapts to the available technologies, allowing active participation in the elaboration of projects at all levels of complexity. For more than a half century now, Logo has become many more elaborate versions of itself (there is around 300 learning environments based on Logo today, including LEGO\(^6\)Logo, MicroWorlds\(^7\), Programmable Brick\(^8\), Logo Blocks\(^9\), StarLogo\(^10\)), with its most popular variation represented by Scratch\(^11\) (Logo Foundation , 2015), as further described in the following section.

---

5 Logo is a Programming Language designed as a tool for children to learn programming in 1967, for through activities in maths, language, music, robotics, telecommunications, and science. Logo is “used to develop simulations, and to create multimedia presentations and games”. (Papert, 1976)
2.3 Scratch: the popular choice of block based programming education

Product of a collaboration between the Lifelong Kindergarten Group at the MIT Media Lab and Yasmin Kafai’s group at UCLA, Scratch is based on the educational concept of Logo, with a tweak of drag and drop of blocks of code (inspired by LogoBlocks and EToys\(^\text{12}\)) instead of typing of code (Maloney, et al., 2008, p. 367). As described by its creators:

“Scratch emphasizes media manipulation and supports programming activities […], such as creating animated stories, games, and interactive presentations. A Scratch project consists of a fixed stage (background) and a number of movable sprites. Each object contains its own set of images, sounds, variables, and scripts. This organization enables easy export and exchange of sprites.

Programming is done by dragging command blocks from a palette into the scripting pane and assembling them, like puzzle pieces, to create “stacks” of blocks.” (Maloney, et al., 2008, p. 367)

I the next section, I provide my own, comprehensive breakdown of Scratch, in terms of constituent elements and interactions, presented according to the following categories of interest: environment, main categories and types of blocks, and interactions, culminating in a review of what experience with Scratch has revealed so far. This was necessary for the identification of potential weaknesses for exploring improvement through tangible interaction.

2.3.1 Environment

The user interface of Scratch consists of a Stage, a Blocks Palette area and a Scripts area (Fig.2). The user selects blocks from the Blocks Palette, to build a script on the Scripts area, and the Stage is where results from that script are tested. The purpose of this user interface, in the words of its creators is that of making “the key concepts of Scratch as tangible and manifest as possible. Having the command palette visible at all times invites exploration”, where the exploration is “supported by having the palette, scripting area, and stage simultaneously visible, providing the user with a process model of how their scripts are interpreted by the computer” (Maloney, et al., 2008, p. 368).

On Figure 2, other features of this environment are shown, of which the following are of relevance for basic scripts built by beginners. These features are:

- The Green Flag and the Stop signs, used to run and to stop a script;
- Blocks Tabs, used to show specific categories of blocks, in the Blocks Palette;

- XY Position, used to show the x and y coordinates of the mouse cursor on the Stage area.

2.3.2 Main categories and types of blocks

The blocks of code are divided into 10 categories of functionalities (Dadiwiki, 2014):

Motion: control movement of a Sprite in x and y coordinate positions, and change its direction through rotation, in degrees.

Looks: change appearance of the Stage or the Sprite, its size and its visibility too.

Sound: play sounds, in musical notes, and use instruments sounds with instruments blocks.

Pen: draw on the Stage, by holding the left mouse button and clearing by pressing the ‘c’ key on the keyboard.

Data: store data, in text or numbers (variables).

Events: triggers actions, with most events blocks marking the beginning of a script, and a few used to send signals/messages between objects.

Control: sets a desired organized or unexpected flow of a project when ran (ex.: loops)
Sensing: detect conditions related to objects, such as two objects touching each other, or a key pressed, for example.

Operators: script math equations, through Boolean and Reporter blocks.

More Blocks: create more blocks, with a chosen script and data input.

In addition to these categories, each type of block also has a specific shape. In total, there are six specific block shapes to use in specific cases. The 6 types are as follows: Hat, Stack, Boolean, Reporter, “C” and Cap (Fig.3). Their roles are:

Hat – to introduce a script.

Stack – perform an action if activated.

Boolean – state conditionals true or false (1 or 0), and are usually attached to “C” blocks.

Reporter – report information.

“C” – loop the blocks inside the “C” block.

Cap – end a script or stops it.

Figure 2 - Types of blocks of code, from Scratch. Source: Researcher's own archives.
2.3.3 Interactions

To perform the main activity of building a script from scratch on Scratch, the interactions with the blocks of code are keyboard and mouse-based, and visual. The user builds scripts manually by stacking up blocks of code, relying on immediate feedback of testing these out in a combined visual and manual (mouse and keyboard) interaction.

Examples of keyboard and mouse operations are the moving of blocks of code from the Palettes to the Scripts area with drag and drop motion, and changing values of certain blocks (coordinates, time, and colour) with key-pressing motion. Both mouse and keyboard are also be used for testing scripts that require user input for moving elements placed on the Stage, in projects such as games.

All of these interactions constantly engage the user in visualising results, in a system built to encourage and support screen based graphical experimentation, of which drawing scenarios and characters are a part.

2.3.4 What experience with Scratch has revealed so far

From Creative Computing, an introductory computing curriculum using Scratch: “the activities encourage exploration of key computational thinking concepts (sequence, loops, parallelism, events, conditionals, operators, data) and key computational thinking practices (experimenting and iterating, testing and debugging, reusing and remixing, abstracting and modularizing)”. (ScratchEd, 2014, p. 2).

In a study conducted by Resnick and other researchers at a Computer Clubhouse – an urban after school of youth ages 8-18 -, where Scratch was introduced to novice programmers over a period of 18 months, one of the findings was that most of them “did not identify scripting in Scratch as a form of programming” (Maloney, et al., 2008, p. 370). When asked what programming was to them, they were unable to formulate a response. At first, the researchers were concerned with the missing link between use of the educational tool and their expectation of a successful ‘learning’ of programming, but upon reflection, the essence of Logo and the samba schools were dominant features of success, as expressed:

‘[…] not seeing Scratch as “programming” may have helped Scratch catch on, allowing youth to see Scratch as being in line with their identities as kids, as something “cool”, and as a central part of the Computer Clubhouse culture. After all, the point of engaging youth in computer programming is not to turn them all into hackers or programmers, but because being engaged in the full range of technology
The study also concludes that the digital multimedia aspect of Scratch, along with its simplified mechanics and elimination of syntax errors, facilitated engagement in activities. Added to this, the freedom of creating\textsuperscript{13} played a big part in sustained interest, as also concluded by Brennan \cite{Brennan2013}, after interviewing Scratch users on the main motivating qualities of Scratch, along with the motivations of reward upon successfully achieving an intended goal, and encountering challenges along the process. Another important remark that Brennan makes is that the connection with others, through the online community, in the school or home setting, for breaking down or solving problems together, or for getting expert help, provided support for developing ideas and overcoming challenges. Together with the learning aspect, these findings also suggested that the ‘audience’ aspect of connecting with others was an incentive in itself. The audiences that Brennan mentions, according to those who she spoke to about Scratch (mostly 8-16 year olds), at the time, were mostly of parents or tutors with backgrounds in computer science, programming and engineering.

I agree with Sue Sentence that other fields must provide research that can be directly applied by teachers, to help make the teaching of coding more accessible, in the face of so many difficulties in adaptation with the new subject in schools \cite{Sentence2016}. Scratch and other similar tools of the same concept, have proven to be efficient in conveying computing knowledge, as well as the underlying creative, social and communication skills that it encourages, but it is can still present some challenges to adapt to by those not so familiarized with digital interfaces. This does not mean that we must eliminate digital interfaces and make programming a tangible activity. Proposing a completely different system to teach programming ‘just because its current state seems imperfect’, would mean to waste all the research and experience, and the success, that a product such as Scratch embodies. This we can effortlessly confirm by looking at what the positive experience with Scratch has revealed so far:

- That it is successful in teaching foundational computing concepts and providing the environment for development of programming skills.
- That it has given widespread opportunity of learning programming to many, through its web-based access and online community forums.

\textsuperscript{13} The study at the Computer Clubhouse showed that Scratch Programs that had popular characters (from images pulled out of the internet), were linked to projects of longer engagement, whereas projects with generic characters were more quickly left aside. \cite{Maloney2008, p. 371}
That it only requires a computer, a mouse and a keyboard – which in the case of a tablet device, for instance, has it all embedded in one device only.

On the other hand, the negative experience with Scratch is what makes this research relevant, and so I here mention that they reveal:

- That it is not accessible to all due to its main character of being mainly visual.
- That the mouse - its main physical interface for digital manipulation – is an imprecise tool, placing an intermediate connection (mouse) between user and final goal (screen) that too often misleads movement or poorly represents it, therefore inadequate for the precise activity of programming computers. A simple exercise placing any marker in the middle of a text, with a mouse, is a good enough experience of what I mean by imprecision of the mouse, for the marker might easily not end up where intended.
- That it might be somewhat incoherent in contributing to a society better prepared for accepting and interacting with pervasive technologies by conveying the message through a system deprived of more relatable tangible experiences.

Given these shortcomings in the use of Scratch, it most probably can gain from a more tangible approach to learning, targeting a learning of distributed cognition and the embodiment of concepts transposed to the material world, to help make sense of meaning by association with the real, multidimensional perceptions of humans.

2.4 Tangible User Interfaces and Learning

Most research conducted for TUIs in learning contexts suggests improvements in learner’s engagement and cognition, with little empirical validation of such claims - reasons being range from the difficulties in producing high fidelity end user tests, to the lack of reliable frameworks for assessment of successful outcomes factored into the realm of the cognitive sciences. Meanwhile, designers continue to try out possible solutions to educational problems with the tools available, producing plausible and promising theoretical and practical advancements informed by experienced meta perspectives, individually or collectively contributing to the small number of ideas that eventually become realities in real classroom contexts. Tangibility in education is no breaking news, but however obvious this may seem to us, remembering to make use of our physical abilities to make sense of abstract information (such as computer languages) is a seemingly naïve rediscovery that must not be undermined in such a fast developing digitalized educational reality in need of ways to get information across diverse ‘unmappable’ humane cognitive system. Our immediate solution is to diversify these bridges for communication using the knowledge that we have so far.
It is not surprising that Resnick’s Lifelong ‘Kindergarten’ Scratch project, is “inspired by the ways children learn in kindergarten: when they create pictures with finger paint, they learn how colours mix together; when they create castles with wooden blocks, they learn about structures and stability.” (Berry, 2013) Scratch is an extension of the kindergarten as a style of learning, where “learners of all ages continue to learn through a process of designing, creating, experimenting, and exploring” (Berry, 2013). What strikes me as questionable here is the shift in focus from the hands-on activities mentioned, to the mouse and keyboard mediated interface that Scratch came to be.

Introduced by German educationalist Friedrich Froebel, in the 19th century, the Kindergarten system and the Spielgaben tool set (Spielgaben, 2018), for manipulation of objects in learning, has been around for quite some time now in the educational scene, promoting meaningful learning up to this date, as supporters of Distributed Cognition (Dcog) theory – presented later into this section - would agree. Froebel’s toolset consisted of different materials and construction kits, of which a set of geometric building blocks was probably the most famous, inspiring the creation of Lego, among other popular construction kits. Through manipulation, children learned the proprieties of form, space, matter and composition, as well as they developed creative expression, collaborative and fine motor skills. Although Scratch translates these qualities of learning through digital manipulation of code blocks to build scripts, it does, however barely mimic the real physical interaction of a Spielgaben.

Thinking with the hands is not the same as thinking with a mouse pointer. The amount of information conveyed through tactility, makes real object manipulation more efficient than the manipulation of representations of that object. It is very much a matter of perception, not just a matter of what things are. With the hands, the trying out of explorative thinking has a greater opportunity to materialize virtual blocks, than by solely making mental connections.

To design for learning, it is crucial that one understands the dynamics of learning processes, and the learning in Human Computer Interaction (HCI) is commonly seen through the lens of pioneering professor of Cognitive Science Edwin Hutchins’ and his theory of Distributed Cognition (Dcog). According to Hutchins et al., Dcog shifts the focus of HCI from the confined desktop perspective, to the complex network of information that surrounds us and to computer-mediated interactions (Hollan, et al., 2000, p. 174). Hollan et al. talk of ‘intelligent use of space’ (HOLLAN, et al., 2000, pp. 190-192) - how we manipulate objects in patterns that illustrate a distributed cognition process, and aids the achievement of our cognitive goals. We find these patterns in the grouping of puzzle pieces to facilitate cognition, as much as we find them in the preparation of the set-up of ingredients of a recipe, to facilitate a cooking activity. Such ‘set-ups’ or ‘organizations’ are a cultural construct that we embody over time, through active use of interpretations based on interactions with our surroundings.
For Professor of Interaction Design and Learning Technologies Michail Giannakos, skill and motivation are equally important factors in learning, and the diversification of activities is key in developing computational thinking. Researchers Peta Wyeth and Helen C Purchase, say that performance and engagement have shown to be directly proportional in learning simple programming. Lesley Xie, Alissa N. Antle and Motamedi mention enjoyment and engagement as “integral and prerequisite aspects of children’s playful learning experiences” (Xie, et al., 2008, p. 192). They make the observation that many authors have relied on quantitative studies that measure engagement by frowns, smiles, yawns and such among participants during a playful learning experience, and criticise this method for its subjectivity. Instead, the authors prefer to evaluate engagement by means of monitoring participants’ ‘on-task’ activity time and ‘repeat play’ (Xie, et al., 2008, p. 197). As for enjoyment, they base it on the Self Determination Theory (SDT) which relates enjoyment to its “intrinsic motivation, where intrinsic motivation is a product of a person’s natural inclination toward spontaneous interest and exploration that is essential to cognitive and social development” (Xie, et al., 2008, p. 192). Intrinsic motivation can be measured through a questionnaire that can be tailored to suit a specific study. Aware of the lack of empirical studies to support their claims, the authors still find reasons to believe that “tangible style interfaces, which rely on direct physical manipulation and support face-to-face collaboration, are more “natural” and thus more enjoyable and engaging for children than desktop environments” (Xie, et al., 2008, p. 191).

What these authors have in common is they all argue that in learning programming, performance and engagement are co-dependent. As previously mentioned in this dissertation, users of Scratch have reportedly lost interest in it in an out-of-classroom setting, and this is enough reason for us to believe that Scratch is not engaging enough, therefore performance in learning may suffer negative consequences. These authors also agree that tangible interaction is important for both performance and engagement in the learning of programming. For Giannakos, the affordances of predictability and function in Tangible User Interfaces play an important part in helping to make the abstract “more concrete and manageable for young students” (Papavlasopoulou, et al., 2017, p. 1812), than the affordances of virtual programming. Tangible Programming Languages (TPLs) have the ability to mimic how we interact with everyday objects, and the ability to reduce children reluctance with coding” (Papavlasopoulou, et al., 2017, p. 1814) by facilitating ‘programmability’ through the communication between blocks and their ‘apparent’ behaviours and the exposure of, otherwise, hidden signals (Wyeth & Purchase, 2002). In the case of

---

14 Michail Giannakos is an associate professor at the Department of Computer Science of the Norwegian University of Science and Technology.
older children, not understanding the nature of the behaviours in programming has shown to make debugging harder; manipulating TPLs help to understand programming behaviours because the physical aspect aids a more straight-forward understanding of syntax (accessibility) (Wyeth & Purchase, 2002).

Xie, et al., believe that single user access and the indirect interaction mode constrained to a 2D space of a graphic interface makes learning more difficult, judging by their own observation of time for completion and of the initiative to repeat of children playing with three different types of interface in puzzle games: graphical, physical and tangible. For these researchers, collaboration is an important variable because “children communicate and learn through social interaction and imitating one another” (Xie, et al., 2008, p. 192). Testing in pairs has helped them to observe collaboration and weave these insights into their work.

In my understanding of the importance of manipulating objects to interact with block-based programming environments for learning basic programming concepts, it has much to do with life-long acquired ability to recognize meaning in associations with real life. Independently of the nature of the process by which cognition is established in a person’s mind-brain system, this person is always using pre-established knowledge to perform these processes. Pre-established knowledge and skills in computer literacy do not happen overnight, and as a product of a life-long construct of associations that we make between the abstract and the concrete, commonly relying on real world scenarios to make sense, conceptual learning must be better understood through physical representation. As Michail Giannakos suggests: “educators should develop learning concepts and scenarios aiming deeper social and personal purposes that users engage in.” (Papavlasopoulou, et al., 2017, p. 1814)

There is much research done in the topic of manipulatives for computational thinking, of which many aim at the learning of programming through tangible interaction with objects. Most TPLs are for very young children who run the risk of getting lost in solely creating when handling physical representations, rather than trying to program (Horn & Jacob, 2007, p. 4). Older children have better ability to cognitively access the underlying concepts and perform a more structured learning of instructions. For this reason, Horn and Jacob also suggest that programming activities be targeted at second graders, for they are more developmentally appropriate for that age group. Giannakos, on the same thread, believes that there must be a bridge between tangible and virtual interfaces, with commands clearly mapped out, to facilitate the process of moving on from TPLs to more traditional IDEs, as a learner progresses. I see this as a call for designers to explore the possibility of hybrid interfaces for programming education, where physical and virtual collaborate into facilitating access to abstract concepts, possibly resulting in better learning through sustained engagement and development of computing skills.
2.5 State of the art

My main purpose with this section is to identify patterns in design decisions and findings from exemplars of TUIs integrated with on-screen technologies, reviewing existing reports on conceptual development and practical results from prototype tests conducted, and relate findings to theories of tangibility in learning mentioned in the previous section. This specific combination that I chose to review reinforces my intention to develop a similar concept that will preserve the successful qualities of Scratch as a tool for on-screen foundational learning of programming, and make use of qualities of TUI interaction for enhancing the learning outcomes with Scratch, motivated by the opportunities presented by the shortcomings with its existing platform and the academic encouragement to use design for developing other ways to teach computing.

Figure 3 - GUI (left) and TUI (right). Source: Are tangibles more fun? Comparing children’s enjoyment and engagement using physical, graphical and tangible user interfaces. (Xie, et al., 2008, p. 196)

Xie, et al., draw their conclusion of TUIs as beneficial for learning from observations in their comparative empirical study on physical (traditional), graphical and tangible interfaces for school-ages kids (Xie, et al., 2008, p. 191). The Physical User Interface (PUI) was a traditional cardboard puzzle from the games industry, for children older than 5. It had a poster of the final image that was used as an underlay (a guide) and took in average 15 minutes to complete. The Graphical User Interface (GUI) was also a traditional puzzle game, but on the desktop (already developed by a software company), mediated by the use of a mouse (to rotate, and drag and drop puzzle pieces), with correct matches amplified by audio and visual feedbacks. In the GUI, a reference image could be extended for reference or hidden at any time, and the average time seemed a little longer due to the size of the workspace on screen. The Tangible User Interface (TUI) – made specifically for use in this study - was a hybrid of the tabletop version of the GUI with physical puzzle pieces as in the PUI, tracked by fiducial markers (placed on the edges of intersecting pieces) recognized by the program through a ReacTIVision engine. Audio and visual feedback were also triggered with correct
connections, just as in the GUI. This TUI resulted in a tool that embodied proprieties and functions of both TUI and GUI (see Fig.3).

From their observation of collaboration, they report that single-mouse use in GUI required taking turns, preventing both participants (tests were conducted in pairs) to be fully engaged at all times as seen on a table top space with multiple access points, players and full-body interaction invited for collaboration. In observing physical manipulation, they also found that children preferred to use a more direct style of contact/interaction, such as touch-screen, over the mouse interfaced interaction, as it gave them more control of movement (precision). In addition, having a reference guide seemed helpful in the completion of the task – in their case it was the image to be formed with the puzzle -, providing an anchor for consulting when in doubt, like a manual. Observing special strategies, the authors observed that in PUI and TUI tests, children picked up pieces randomly and matched them to the reference image. In the GUI, this did not happen, probably because of the fact that the image was a pop-up feature that was not always there to guide the players. Observing preference over one mode or another, children seemed unable to pick a preferred type, but complained that the GUI has too many pieces, when in fact it had less pieces that the PUI and the TUI, which I believe to be once again a problem of not being able to relate realistically to a 2D environment. Overall the indirect interaction mode constrained to the 2D environment of the GUI presented difficulties in cognition and barriers in accessibility and collaboration.

Figure 4 - The TangiLearn concept and experimentation. Source: retrieved from article Designing Tangible Multimedia for Preschoolers based on Multimedia Design Theories (Tsong, et al., 2012). Access on April 28, 2018.

Tsong, et al., (2012) have also experimented with hybrid technologies for educational purposes. Their concept of TangiLearn (Tsong, et al., 2012) was part of a case study for exploration and demonstration of the need for design theory and guidelines on Tangible Multimedia, with the belief that the sharing of cognitive loads through diversified media is beneficial in learning. In TangiLearn, the Learning, Quiz and Problem-Solving on-screen sections are designed to accommodate the use of tangible objects. Binding between these tangible objects and their virtual representations occurs when the learner grabs an object and aligns its QR code marker with the computer camera, for the Adobe Flash Professional CS5 ActionScripts 3.0 to identify the object (see Fig.4).
The learning flow with TangiLearn is made to be cyclical and sequential, going "from tangible object exploration, followed by cycles of conceptualization, reinforcement (through quizzes) and application of the concepts (through problem-solving activities). This format of learning ensures “lasting acquisition of knowledge” (Tsong, et al., 2012, p. 2). The authors base their design choices on principles of multimedia instructional design from well-established cognitive theories such as Mayer's Cognitive Theory of Multimedia Learning\textsuperscript{15}, the Dual-Coding Theory\textsuperscript{16} and the Cognitive Load theory\textsuperscript{17} for their design of tangible multimedia. These theories support the following design choices: to provide more sensory channels that are not solely visual and audio, adding the tactile channel to better distribute the cognitive load, therefore freeing up space on each channel for learning given that each channel holds a limited amount of information; to provide up to 7 units of information only, which in the case of TangiLearn result in the use of up to 7 objects in the interaction with the proposed activities; to provide several stages of learning that extend the basic behaviourist model of teach-test to the proposed model of cycles of explore-conceptualize-reinforce-apply of TangiLearn, as a way to mitigate the forgetting curve in humans by repetition; to provide all relevant material placed close to each other (adjacent), because according to the spatial contiguity principle this avoids the split attention effect that divides a learner’s attention as this helps children to make connections between elements of a learning activity (in TangiLearn all elements are restricted to a small learning area that comprises the computer and an object manipulation area just in front); to avoid irrelevant material, as stated by the coherence principle that says that children absorb content better in the absence of extra and redundant information, leaving simultaneous multichannel information as a feature of recognition activities (where learners learn about an object, for instance) and single channel input in problem-solving activities.

![Figure 5 - Fiducials on conventional toys, system setup, and single or multiplayer interaction with NIKVision. Source: retrieved from the developer's official website http://webdiis.unizar.es/~jmarco/. Access on April 28, 2018.](image)

Much of this is also found in other tangible multimedia systems, such as NIKVision (Marco, 2009), a tabletop prototype for play with tangible toys while interacting with digital multimedia (see Fig.5), and Osmo Coding, (PlayOsmo, 2014) a set of tangible blocks magnetically

snapped together, for small children to assembly of sequences of instructions for Awbie – main character of the game - to follow (see Fig.6).

In NIKVision, conventional toys with fiducial printouts attached to their bases are placed on a tabletop setup with reacTIVision technology recognition, where projected scenarios invite for tangible interaction. The tabletop in NIKVision is the input arena that triggers visual and audio outputs either on the projection itself or on a computer screen placed onto one of its sides, depending on the game played. In this exemplar, there is an even balance between visual, audio and tactile elements, with affordances related to the pressing of buttons, movement and matching of elements, stamping and so on intuitively encouraged by graphic instructions and feedback. Just as in Osmo Coding, which consists of snapping tangible blocks of code together to instruct a character called Awbie to follow certain paths that lead him to eat as many strawberries as possible in the virtual world game of Coding Awbie (on an iPad): the tangible blocks of code are clearly indicative of how to snap, when to turn, and the blocks’ functions indicated by graphic symbols. Osmo Coding, just as NIKVision and TangiLearn ensure that all elements related to its learning activities are places adjacent to one another. It uses Reflective Artificial Intelligence technology, which consists of reflector (a mirror) attached to the IPad’s camera and computer vision algorithms that process the data scanned of the surface beneath it, which consists of the combinations of blocks of code built by the user(s). Its lose tangible elements allow for multiplayer interaction and special exploration, just as we find in the other exemplars here presented.

Although there is no extended empirical evaluation on the success of Osmo Coding and of NIKVision, thus we are left to evaluate it through observation and comparison with other similar systems for tangible multimedia learning, reports on TangiLearn provides us with insightful test results. Its researchers conclude that its system is successful in its purpose of making multimedia learning more enjoyable, engaging and efficient by its creators. Their evaluation was based on interviews, observation and questionnaires, and what its

---

18 “The reacTIVision system is software for tracking specially designed fiducials (markers) in a real-time video stream”. (Bencina & Kaltenbrunner, 2005)
researchers found is: that all 6 participants/evaluators reported enjoyment with the activities and presented high levels of engagement in the learning throughout the test session all the way until it reached its end; that the level of difficulty experienced in problem-solving varied according to participant’s own skills and difficulties; that introducing tangibles in interactions with this visual and auditory system did help to distribute the cognitive load and make learning more effective; and that they grasped content in a near immediate fashion when handling the objects and without hesitation interacting with the system. The researchers also found that the children moved around TangiLearn, thus presenting a full body exploration of spatiality in their experience of learning. As said by Tsong, et al., “In TangiLearn, tangible objects and multimedia objects are designed to complement each other meaningfully” (Tsong, et al., 2012, p. 4), where problem-solving, for instance, is aided by clues given by the multimedia objects, as to what types of actions to perform with the physical objects. In choosing the objects to include in this system, the researchers suggest, by experience, that objects be closer to ‘toy’ like figures (animals and people), rather than ‘object’ like figures (books, cubes), and that some measure of balance between tangible objects, visual objects and audio cues be considered so that none of the different technologies stand out from one another, inducing the user to engage with individual parts of the system.

2.6 A summary of considerations for conceptual development

As quoted in the beginning of this thesis,

“The search for new integrative disciplines to complement the arts and sciences has become one of the central themes of intellectual and practical life in the twentieth century. Without integrative disciplines of understanding, communication, and action, there is little hope of sensibly extending knowledge beyond the library or laboratory in order to serve the purpose of enriching human life.” (Buchanan, 1992, p. 6)

I quote Buchanan because of his ability to describe the essence of Design Thinking as a complex practice that has the ‘power’ to produce knowledge applicable in practices. Solutions to design problems are found at the confluence of varied specialized sciences, such as the Computer, Social and Natural sciences. Interaction Design is, perhaps, the most complex science of communication, in my opinion, for it is tailored to handle communication problems of diverse natures, whether the communication of a message, data, feelings, materiality, functionality, and use, just to mention a few. Independently of the field of Design, whether Interaction, Graphic, Architectural, or other, the aim is always some sort of communication, and in this process the designer needs to understand, in depth, the object(s) of communication and the message(s) to convey.
In the case of this research, the objects of communication are bodies, mind-brain systems, tangible objects and visual block based programming environments, and the main message consists of programming concepts, programming with blocks of code and the presence of computer programming in our physical world.

From my findings of previous research on programming education and accessibility, I have learned that tangible elements and instructions must all be part of a whole that surrounds the learner, making learning elements close enough for the learner’s exploration, and for them to maintain the coherence of a system in which all of its constituent parts are equally necessary and work together for the final outcome of a programming activity. This information is strengthened by all of the main related fields of research discussed in this chapter.

Another important factor is collaboration, because it is through collaboration that concepts gain shared meaning, as learners explore concepts and their purposes, just like a block based language is meant to convey universal messages regarding programming. This claim is also made by experienced professionals involved in computing education, as it also is in reports from experiments with Scratch and the state-of-the-art exemplars here presented. Collaboration is also known for increasing motivation in learning activities, and this factors in the search for interactions that better engage learners, concomitantly increasing and providing a good setting for the development of better performance; an idea shared many authors on TUIs and Learning, as I have exposed in subchapter 2.4.

Currently, Computing Education – and other domains of education - , is pointing towards less direct approaches in the learning of programming, suggesting that we consider weaving this domain of education into the larger social sphere, beyond the regular two-dimensional computer screen. The most immediate transfer of this understanding of a more immersive education is found in experiments that use tangible objects as an extension of virtual worlds into our real world, as I have presented in subchapter 2.5, with Osmo Coding, TangiLearn and NIKVision, for instance.

Moreover, including three dimensional tangible interactions in the learning of on-screen programming, can: reduce children’s reluctance to coding, by providing them with the affordances of predictability and functions of tangible objects, to make sense of the abstract on-screen elements (subchapter 2.4); make debugging easier to follow for older children; second graders are more prepared to understand the underlying concepts in programming activities than younger children; distribute the cognitive load by adding the tactile channel to the visual and audio ones; bridge the transition from the currently available TPLs to the currently available visual programming interfaces.
3. METHODOLOGY

The overarching methodology to guide this study is that of Design Based Research (DBR) for learning from the practice of iterative multidisciplinary explorations during concept development, constantly embodying the diverse progressive findings, for improved knowledge and present a final artefact that reflects better possibilities to address the current problem exposed. Multidisciplinary information then contained in both process and final concept becomes a knowledge resource at the disposal of other researchers and practitioners, not only in the field of Interaction Design. Educators, software and hardware developers, special needs teachers, students, and possibly others looking to grasp new insights on block based programming education, are all prone to find new knowledge contributions to their fields.

Design Based Research – also known as Research Through Design – can be defined in literature as "a research approach that employs methods and processes from design practice as legitimate method of inquiry", where DBR is "an inquiry process revolving around the making of a product, service, environment, or system" (Zimmerman, et al., 2010, p. 310). DBR is effective in close study of learnings through experimentation in a context of theoretically inspired innovations that unfold through multiple iterations, establishing “making as a method of inquiry” (Zimmerman, et al., 2007, p. 496) in order to address complex problems.

The problem with complex problems, for design - and as such, for the practice of design in DBR -, is that given the extent of its problem space - the design space on which the designer must act to solve a problem -, becomes ‘ill-defined’ (Simon, 1973). In this situation, with the design space not given by the problem as commonly perceived, “the designer need to delimits his/her design space according to how he/she finds necessary for the attempt to find solutions for a given problem” (Schön, 1992, p. 11)

Much of what constitutes the structure of a DBR process and product is defined according to the designer’s best judgement of what it may contain and how it must be done. Expected results of DBR are new theories, artefacts and practices (Barab, 2006, p. 153) that describe a vision of a preferred state to that presented at the start of the investigation, “increasing the chance for knowledge transfer to the research, practice, and education communities” (Zimmerman, et al., 2007, p. 498). Although there has been much discussion on the subject and there is no agreed upon method to document the results that emerge from this type of research (Zimmerman, et al., 2010, p. 310). According to Zimmerman et al., knowledge produced from a DBR research process can be, implicitly expressed by the resulting artefact, or even “an endless string of design examples” (Gaver, 2012, p. 938).
Communicating design research implies communicating an open-ended design process and final outcomes, according to my interpretation of Gaver’s use of the word ‘endless’, combined with the understanding of making – an activity that expands ideas, rather than concludes them - as an ideal method for design research, as expressed by Zimmerman et al., as above mentioned. Design Research must point to solutions at the same time as it makes space for further growth through future appropriation by others. Added to this, I like to keep in Gaver’s understanding of DBR as a practice that is generative by nature, and needs moderate rigour for success in “exploring and speculating, particularising and diversifying,” and manifesting “results in the form of new, conceptually rich artefacts”. (Gaver, 2012, p. 937) In this exploration, I turned to literature and state-of-the-art review, reflection-in-action, interviews, informal brainstorming, and prototyping and testing, as methods for my diversified design exploration in order to produce ‘new, conceptually rich artefacts’.

3.1 Methods of design exploration and concept development

3.1.1 Literature and State-of-the-art Review

Nothing that we create is completely new; we all have a repertoire from which we build ideas. As a practice that accompanies the research throughout most of its process, the review of relevant literature and state-of-the-art projects served as an informative basis. This included mostly published works in the fields of Computing Education and Tangible Interaction in learning, available in the form of articles, books, online forums, videos and websites. The importance of this review has allowed me to gather useful information on past and the current state of and discussions within Computing Education, and on the development of programming languages over time, along with the evolution of learning systems - from tangible to digital -, as technological advancements progress. Academic and practical explorations that have been reported on the topic of Tangible Interaction with multimedia for learning were insightful in tracing a panorama of what we know today, how it became known (in terms of methods and technologies), what worked and what did not work, and why it is relevant to continue exploring within this topic.

3.1.2 Reflection-in-action

For Schön, “design knowledge is knowledge in action, revealed in and by actual designing” (Schön, 1992, p. 3), in which “knowing in action involves sensory, bodily knowing. The designer designs not only with the mind but with the body and senses.” (Schön, 1992, p. 5) One way for designers to make use of this knowledge in and for design, is to place themselves in the mode of doing. To this practice it is given the name Reflection-in-action (RiA). In RiA, active, sensory appreciation of a design material enables the designer to assess what is and is not relevant in that material for that design space for the purpose of
design, “creating a ‘design world’ within which he/she functions” (Schön, 1992, p. 4) by means of “seeing, moving and seeing again” (Schön, 1992, p. 5).

Designing requires conversation with the materials of a design situation, whether real or virtual, and in all stages of design. For the design researcher concerned with learning something new from his/her own explorations within a topic, it is as important to experience phenomena of interest and observe it from the performer’s (the designer) perspective, as it is to observe phenomena performed by a participant of the research. Reflection-in-action is, as I see it, implied in any design research, but for this one, specifically, I have interest in this experience of phenomena by the designer, meaning that the designer must experience proposed interactions at all stages of the process, as a catalyst of learning, by doing, for knowledge formation in design.

3.1.3 Interviews

Edwards and Holland, experts in social research methodology, state that interviews range from structured to unstructured, through semi-structured interviews. They explain that structured interviews are for use in surveys and result in quantitative comparative data (Edwards & Holland, 2013, p. 2). On the other hand, semi-structured and unstructured interviews have higher flexibility and lack of structure (Edwards & Holland, 2013, pp. 2-3). Both types of interviews were of use to this project.

Semi-structured and/or unstructured interviews were beneficial in generating discussion, whilst maintained focused on the research problem, questions and aims. This investigation provides space for insights, answers and new questions to arise from its open format, as opposed to the constraints imposed by the question-answer structure of a structured questionnaire. The sessions were in person and through video calls, and mostly related to the third prototyping stage of Experience Prototyping, always aiming for gathering information for identifying strengths and weaknesses of my research, according to target users.

3.1.4 Informal Brainstorming

Originally introduced by Alex F. Osborn (Osborn, 1953) as a strictly structured method for ideation, Brainstorming can also be informal, according to Jeffrey Baumgartner (Baumgartner, n.d.). Baumgartner states that, although the traditional formal structure of a Brainstorm is that in which “you post a problem on a whiteboard or chalkboard and invite a group of people to suggest ideas”, an informal type of Brainstorm may occur as an unstructured ideation activity, in a smaller group of participants, as a way to produce as many ideas possible to solve a given problem. Arguably, brainstorming can also be performed individually, as I do, in recording thoughts and grouping or shuffling them around, and
sketching ideas, to then visualise the whole, letting the flow of thoughts draw out conclusions, in a process that can be considered by Baumgartner as Informal Brainstorming.

3.1.5 Prototyping

Prototyping is “any representation of a design idea, regardless of medium”, which “includes a pre-existing object when used to answer a design question” (Houde & Hill, 1997, p. 3). The prototypes of this research are of low-fidelity, in a sense that they represent (do not present) a final, fully functioning version of the proposed concept. Added to this, they are Integration Prototypes (Houde & Hill, 1997, p. 12), with all main aspects of the system - look and feel, role and implementation – are present for exploration and evaluation of the relations enabling the intended outcomes.

The choice of a low fidelity mind-set has the purpose of maintaining focus on investigating interactions within a proposed system, prioritising the experience, and by prioritising experience the choice of integration prototypes provides a testing ground with no specific focus, where all elements – look and feel, role and implementation - are harmonized to prevent tendentious starting points to exploration and evaluation.

Regarding the use of the term low-fidelity when speaking of prototypes, I would like to dedicate a paragraph to a personal view of disagreement, although referring to my approach, not to my prototypes themselves, as an approach with a low fidelity mind-set. I see ‘low-fidelity’ as a poorly administered judgement of value in the design practice that requires clarification of what aspects of design we are referring to as low-fidelity and parameter of reference we consider as high fidelity in the particular case. These terms trouble me with regards to their purpose which may often be that of downgrading a design work, leading to biased and misconceived undermining of its value. I assess my prototypes in this research as concomitantly high and low-fidelity, and I will explain what I mean by this. My research uses prototypes to explore interactions for a conceptual programming environment, and my prototypes need to provide a system that enables integrated manipulation and visualization to perform certain activities. To maintain focus on interactions, I chose to build the simplest prototypes possible in order to make a point of experimentation and space for iteration of interaction design. Simple prototypes are commonly perceived as ‘not fully functioning final prototypes’, therefore my choices of prototypes would be considered low-fidelity by default, when in reality they are low fidelity prototypes in materiality only – with no final hardware and software configuration, for instance -, and high fidelity prototypes in their purposes of conceptual experimentation and evaluation.
However divergent my opinion is, I find is necessary to mention these terms, to indicate that I purposefully do not provide a polished, fully functioning final prototype in this study, with the prerogative of promoting "the attitude that it is the design question that is important, not the tools and techniques that can be brought to bear". (Buchenau & Suri, 2000, p. 431)

![Figure 7 - Setup of shooting the Experience Prototype video. From left to right: laptop with tangible objects on screen, ready for building Scripts; tripod, camera; 4 LEGO pieces, small magnets, doughnut shaped metallic 'Ref base'; palettes, sticky rubber for attaching LEGO's to the screen, pile of LEGO. Source: Researcher's archives.](image)

It is tricky to be succinct about one or another approach to prototyping because to build and execute such artefacts there is mixed media like paper mock-ups, sketches, role-playing, enactment, wizard-of-oz and video recording involved. According to Mackay et al., video recording a Wizard of Oz simulation (see Fig. 7), to fake a demonstration of the computer’s response to computer mediated interaction "is a particularly powerful video prototyping technique because it gives the user a real sense of what it might actually feel like to interact with the proposed tool, long before it has been implemented. Seeing a video clip of someone else interacting with a simulated tool is more effective than simply hearing about it" (Mackay, et al., 2000, p. 78). Video prototyping a wizard-of-oz also allows the mechanisms used to fake a human-computer interaction to stay hidden, providing a more realistic representation of the idea than would be possible in person. In addition, this format of representation of an idea is in itself a source of inspiration and information that can be posteriorly analysed multiple times, and can simplify physical barriers in communicating the idea with others. For Mackay et al., their experiences with video recorded prototyping sessions show that participant’s feedback had more detailed descriptions, encouraged by the awareness that the video would later be reviewed and analysed (Mackay, et al., 2000, pp. 81-82).

If ‘seeing a video clip of someone else interacting with a simulated tool is more effective than simply hearing about it’, physically interacting with it is even more powerful. Experience
Prototyping can engage all senses. For IDEO Design Directors Marion Buchenau and Jane Fulton Suri, Experience Prototyping is “a form of prototyping that enables design team members, users and clients to gain first-hand appreciation of existing or future conditions through active engagement with prototypes” (Buchenau & Suri, 2000, p. 424). Going back to Reflection-in-action (RiA, as referred to in p. 31), it has the same quality of direct engagement in experiencing something new as found in Experience Prototyping. It is with focus on ‘what they prototype’ that I call the prototypes used in this research Experience Prototypes, for they allow the testing and evaluation of an experience with a design artefact, by experience, and stimulating reflection-in-action. The main purpose of Experience Prototyping is in facilitating the design process towards a more informed development of the user experience and the tangible components that create it. In this stage of design, this should provide inspiration, confirmation or rejection of the concept or parts of it, according to Buchenau and Suri (Buchenau & Suri, 2000, p. 431), providing integrative answers and feedback that weave together all three aspects of the design of interactive artefacts, role, look and feel, and implementation. This approach places designers and users in this position thorough understanding, providing common ground for establishing a shared point of view (Buchenau & Suri, 2000, p. 431). What follows is the report of my conceptual development and prototyping stages within this design process (see Fig. 8 for an overview of the process).

4. DESIGN PROCESS

![Figure 8 - Overview of the design process. Source: Researcher's own archives.](image)

4.1 Interviews

I interviewed several professionals involved in computing education during all stages of the process, to ensure that my ideas evolved in alignment with their experienced perspectives. In this section I present three of these interviews; the three interviews from the exploration phase, that helped me to reframe this project and to decide on how I would conduct my conceptual development. Conducted in person, these three sessions had no predefined set of questions. They were initiated with an initial problem statement of ‘the audio-visual format
of Scratch being inaccessible to some extent and distant, in format, from its original ideal of an education that is weaved into a larger, richer cultural-social experience and is up-to-date with the technologies available today’, and the assumption that ‘there has to be other forms of interaction with Scratch, with contemporary technologies, that can extend the learning experience into a richer cultural-social activity, and this may be achieved by adding tangible three-dimensional components, as background studies suggest’.

Each interviewed participant was from a related field of interest: one educator experienced in computing education; one researcher of ubiquitous computing; and one professor of creative technologies. My first interview was a small set of informal conversations with Per Falk, coordinator of Väsby Makerspace, in Sweden, who has vast experience with computing education, and is engaged in creating physical and virtual environments for learning. His clarity in the processes of learning and vision of opportunities for research within the field, of the kind that only comes with experience, had a profound impact in my initial ideas. My first thoughts were on making a popular educational physical computing device that is programmed with Scratch (a BBC micro:bit, for example. See Fig.9) more accessible, and more intertwined with the act of on-screen programming itself so that it is not as dissociate as it is today. Per Falk advised me to play with a micro:bit myself, as a first exploration activity.

Before experimenting with micro:bit, I also interviewed Daniel Spikol, Professor at the Internet of Things and People (IOTAP) of Malmö University. Spikol had been working on a similar project focused on tangible computing, and contributed with background literature and state-of-the-art examples. Perhaps, his subtlest contribution was the most helpful, and it was in the form of a warning: to be pragmatic – to prototype ideas, test and learn from them. The reason for him to suggest this was the fact that, from his experience, the path of investigation that I
was setting off for is one that he has done many times. It is time consuming and it requires many resources, so instead of focusing on a final product, I should focus on interactions and conduct experiments for exploration and ideation.

With Falk’s and Spikol’s suggestions, I experimented with Scratch and micro:bit, and in a combination with past experiences with other TUIs for learning programming, I realised that my focus was not on reinventing TUIs for use with Scratch, but on Reinventing Scratch itself. Microcomputers, robots, and such, are all tools that have been successful in their purposes just as Scratch, but still present opportunities for further development (just as Scratch does too). Microcomputers and robots, for instance, present several inaccessible features to the blind, for instance; a micro:bit has LED features that require a visual ability, and BeeBot wonders off out of reach as it moves automated according to instructions given by the user. What mostly became evident to me at this point, was that neither micro:bit or BeeBot types of programming devices seemed to provide a unified programming structure, given their disruptive or inexistent (in the case of BeeBot) positioning in relation to on-screen programming activities, leading an apparent distancing between input-output relationship, and to alternated focus on one ‘or’ the other, not on one ‘and’ the other.

This immediately led me to search for examples of learning tools that merged input-output in a shared physical space. Exemplars that inspired me and began to make sense in a holistic sense – according to my interviews and experimentations, and to theories of TUI design and Learning, as well as the whole history and present state of Scratch – were those that proposed a sort of hybridity of technologies, integrating visual, audio and tactile senses, and presenting an area of research that is much less saturated than that of purely programming by Physical Computing.

As I gained interest in the possibility of exploring physical three-dimensional interaction with two-dimensional worlds, I had the chance to exchange ideas with Professor Francisco Lopez Luro, from the Department of Creative Technologies (DIKR) of Blekinge Tekniska Högskola, for insights of technical nature, discussing current technical solutions available, from analysing state-of-the art technical solutions to similar hybrid concepts. His suggestions informed the possibility of relying on systems of recognition for enabling integration of objects in the software design of computer environments. The main reference that Professor Luro spoke of was Reactable (see Fig.10), a touchscreen table for playing music with interactive tangible objects. These objects are identified by the system though fiducial tags attached to them. Reactable is part of a team effort to “further develop the instrument and its possibilities” (Reactable Systems SL, s.d.), from research at the Pompeu Fabra Univerisity, in Barcelona.
Reactable represents the sort of integrated multisensory experience that I would like to see Scratch users immersed in. It is intuitive, there is a balance amongst its sensorial components that sustain engagement with the whole system instead of diverging interest onto one or another element, there is immediate real-time feedback transferred to that social cultural context via spatial connections with physical objects, and there is collaboration with all participants provided with the same tools and space for interaction.

### 4.2 Conceptual Development

Just as the Reactables team did with music, I set off to further develop Scratch as an instrument for learning basic concepts of programming and its possibilities beyond its two-dimensional environment, motivated and inspired by the history and shortcomings of Scratch, and by Reactables itself and the related State of the Art research presented, which support Tangible interaction for learning and present studies on hybrid Tangible-Visual systems for learning. Yet, I have not come across a study of hybridity for the specific learning domain of programming.

My process started with a close look at Scratch once again, and identifying two main interactions that are dominant for the act of programming itself: input interaction and feedback interaction (see Fig.11). These are both displayed with distinction from its
remaining support features, occupying most of the visual interface of Scratch and are as follows:

- The input interaction of building scripts by dragging blocks of code from a source of blocks, then dropping them on an area where these blocks can be puzzled together to form scripts that the system interprets, to then execute; and
- The feedback interaction of 'staging' a program, by creating visual elements (backdrops and characters) that perform the instructions given, automated (ex.: tutorials) or according to user input (ex.: games).

There are Tangible variations commercially available today, that attend both main interactions, although isolated from one another. For the input interaction, Osmo Coding (see subchapter 2.5) is a good example, with its tangible blocks of code that snap together. In the case of Osmo Coding, the feedback is solely visual. Osmo Coding depends on a Screen but has its interactions physically disconnected from one another. For the feedback interaction, plug-and-play toolkits, such as micro:bit (see subchapter 4.1), are a good example of a system that transfers code to physical material. In the case of plug-and-play toolkits, the building of scripts is then exercised visually, in virtual environments such as Scratch, but physically disconnected from it with the exception of a USB cable connection.

Hence, my initial proposal of a concept focused on exploring the physical manipulation of Scripts elements and of Stage elements, whilst maintaining the default steps of learning
activities in Scratch. For this stage it was kept in mind that this would then, in future stages of exploration, be merged with the digital interface of Scratch, to result in a hybrid system.

Figure 12 - Recorded notes and sketches from brainstorming for prototyping the initial design concept. See Appendix A for more notes and sketches. Source: Researcher's own archives.
My assumptions, at this stage, were that:

- This might serve as a conductive thread between tangible programming and screen-based programming, because staging and building scripts are already the main features of existing TUIs. Thus, making this distinction in a hybrid Scratch might motivate a user with previous experiences and basic understanding of programming from interaction with TUIs to interact with Scratch to acquire extra and deeper understanding of programming concepts for screen-based programming;
- This might should help a novice learner’s in making a clearer distinction between the crucial input scripting and output testing activities of programming, and how Scratch can, as a computer, offer extra support features such as store and organize data for automation; and that,
- Both of the above mentioned will aid understanding of programming concepts by means of transferring abstract two-dimensional concepts to concrete physical forms.

My initial concept of a Hybrid tangible-visual block based programming learning environment based on Scratch, was then the subject of brainstorms and sketches (see Fig.12 and Appendix A) for the development of a prototype of this concept that I tested to produce initial insights, validate initial assumptions and strengths to further improve on, and to spot weaknesses to reconsider in further development.

4.3 Tangible Experience Prototyping

Figure 13 - Tangible Experience Prototyping. Source: Researcher’s own archives.
This Tangible Prototyping Session follows on conceptual development presented in the previous subchapter, and as there mentioned, was intended for my personal exploration of the physical manipulation of Scripts elements and of Stage elements, whilst maintaining the default steps of learning activities in Scratch. In addition, Block Palettes were also explored in a physical form as it is part of the action of building Scripts.

For representing this first concept into a prototype with off-the-shelf materials, I built a paper version a simple ‘Maze Game’ from the Scratch library (see Fig.13). I also decided to look for inspirations for possible automated technologies for the Stage Area, given that simply placing objects in this Area – which in Scratch are graphic characters - , was only part of the interaction. I needed these objects to be able to move around the Stage when a program is ran, just as in Scratch. The two main technologies that I came across were: InForm (Leithinger, et al., 2013): “a state-of-the-art 2.5D shape display that enables dynamic affordances, constraints and actuation of passive objects” (Leithinger, et al., 2013, p. 2), developed at the MIT Media Lab (see Fig.14); and GaussBits (Liang, et al., 2013): “a system of the passive magnetic tangible designs that enables 3D tangible interactions in the near-surface space of portable displays” (Liang, et al., 2013, p. 1391). It is equipped with a thin magnetic sensor grid on the back of the screen, enabling 3D positioning and ‘partial’ 3D orientation, through a bi-polar magnetic field tracking technology (see Fig.15). In Form seemed like a far off reality, given the complexity of the structure needed for its pins to go up and down, which could result in a hybrid system much like plug-and-play kits where a separate stage structure would be attached onto the computer to perform. However, it would be interesting to experiment with building maze walls with its dynamic surface by manually pulling up the pins needed for this. The magnetic system of GaussBits would not have a three-dimensionally dynamic surface as an InForm like pin board, but it could be built into a screen where Scratch run and for this reason provide a closer to ‘hybrid’ system and experience of programming. In this sense, GaussBits was more promising from the start, therefore I decided to first experiment with my own prototype version of InForm, to make a final assessment of its pros and cons.

*Figure 14 - InForm experimentation. Retrieved from the MIT Media Lab official webpage, on https://tangible.media.mit.edu/project/inform/. Access on April 28, 2018.*
Figure 15 - Magnetic sensor grid, tangible flight simulation, Sensed magnetic field image, and magnetic unit that goes inside the toys. Retrieved from the article GaussBits: Magnetic Tangible Bits for Portable and Occlusion-Free Near-Surface Interactions (Liang, et al., 2013, p. 1391). Access on April 29, 2018.

The resulting system proposal (see Fig.13) resulted in:

A stage area of consisting of a display made out of pins, that could be shaped to represent the backdrop of the maze, and shaped to move sprites around. This setup included:

- Tangible objects of choice to represent sprites.
- Palettes of blocks of code with detachable blocks to mimic the palettes tabs of Scratch.
- An A3 sized sketch of the layout of Scratch to represent the screen.
- A constraint to keep in mind: although manipulated during programming interaction, all scripts must run by themselves, meaning that the Stage must still react to them through automation or through a user’s input with keyboard or mouse interaction, typical of game play in Scratch.

4.3.1 The Maze Game

To replicate the systematic basic sequence of interactions necessary to perform a Scratch Maze Game from beginning to end, it was necessary to observe this process and record all of its stages. Here is my breakdown of these interactions in step by step instructions, based on the Maze Starter game available in the ‘Projects’ page of Scratch (Scratchteam, 2013):

Description - the player must take the ball to the goal (from opposite corners), avoiding touching the edges of the maze along the way.

Main sequence of interactions – 1) draw the maze backdrop (walls) and create sprites (characters); 2) build Scripts; 3) play (see Appendix B for full sequence).

1) Draw maze backdrop (walls) and create sprites (characters) (see Fig.16):

   a) Navigate environment with mouse to open the ‘Backdrops’ tab and select the Rectangle tool. Draw maze.

   b) Once done, navigate sprites window to find ‘Paint new sprite’, and draw a ball. Then, ‘Paint new sprite’, and draw a goal. Place goal at the end, and ball at the starting point of the game.
2) With the Backdrop and the Sprites ready, build Scripts and 3) Play (see Fig. 17).

   c) Drag blocks of code from the palettes in the categories tab, and drop them in the 'Scripts’ area, to build scripts.
   d) Build the script for the ball. First, state the start position of the ball, in coordinates, for every time that the game begins (when the Flag is mouse-clicked). Build scripts for every ‘arrow key’ pressed on the keyboard, to move the ball up, down, right or left. Change values on blocks, according to direction (in degrees), and how far (in steps)
the ball should go if when an arrow key is pressed. Also add a script for what happens if the ball touches the edge – it must bounce back, the same distance it took to touch the edge (in steps).
e) Build a script for the goal. Select the ‘goal’ sprite, in the ‘Sprites’ window, to write a script for when the ball reaches (touches) it. When the ball touches the goal, but for it must ‘play sound’ (select a sound to be played from the library of sounds).
f) Click the Green Flag to start the game, and play it with the keyboard arrow keys.

4.3.2 Tangible experience of The Maze Game

I used the guide of interactions observed in The Maze Game to conduct my experiment of the same game on my paper and pin-display setup, away from the original Scratch. The original interactions mapped out had to be adapted as I experienced these activities with this prototype. I placed one palette of blocks of code in the middle of my A3 paper layout of Scratch (see Fig.18), the dynamic display (the stage) on the upper left corner and the Sprites area on the bottom left corner, and drew a Scripts area on the right hand side of this. I did not build the palettes for all of the 10 categories of blocks of Scratch, I only reproduced those needed for the Maze Game specifically. Instead of the original main sequence of activities 1) draw the maze backdrop (walls) and create sprites (characters); 2) build Scripts; 3) play, tangible interaction required a slight change in the sequence, resulting in: 1) ‘draw’ the maze Backdrop (walls); 2) ‘draw’ the Sprites (characters); 3) build Scripts; and 4) play, where the ‘drawing’ activity became a ‘build’, ‘create’, ‘assemble’ one.

![Figure 18- Initial layout of Scratch on A3 paper and remaining Palettes of blocks, per category. Source: Researcher's own archives.](image)

With the interactive dynamic shape display – the pin board (see close-up in Fig.19 and position within the prototype setup on top-left corner of Fig.18) -, my vision was that of the user manipulating the pins to construct the ‘goal’ and the background of the game. The choice of a dynamic shape display was also an attempt to mimic, to a certain extent, the creative nature of Scratch, in providing a blank canvas on which to create scenarios and change them.
as wished, erase and start from scratch, and so on. The experiment presented some
difficulties in manually handling the pins of the display, for it required steady hands and
precise movements. It somehow felt inaccessibly equivalent to a mouse interaction. I also
observed that this process slowed down the interaction that is usually faster on the original
Scratch. It felt discouraging, therefore not enjoyable and/or engaging enough to enhance the
learning experience.

![Image of the 'Stage', in a tangible experiment with a dynamic pin-display. Source: Researcher's own archives.]

Adding the moveable Sprite - the 'Ball' (represented by a rubber band ball. See Fig.19) -, showed that having a display of coordinated pins may allow the user to choose a coordinate on which to place the sprite, much like in the original Scratch coordinate positioning system. The user could also place the sprite randomly on the pin display, and let the system tell which coordinates it was placed on, much like Scratch once again. Moreover, in this tangible system, sprites could be any object of the learner’s choice, provided there was a way to manually attach digitally traceable references to them to be identified and tracked by the system when placed onto a pin. One option for these bases would be to have fiducial markers, as used effectively on NIKVision, TangiLearn and Reactables.

Reflecting on the placement and tracking of the sprite, I conclude that there are feasible solutions. The challenge that this pin system presents is in the movement of the sprites. How could we have automated move on a pin board? Although I do not intend to specify the technologies necessary for such a product, exploring the moving of the Sprites on the physical Stage area presents trouble regarding look and feel aspects. For the question of how to make it move, I imagined that magnetic pins could be activated sequentially to move
a metallic sprite around, for instance, but it would still result in a system far from seamless and smooth, possibly noisy, and as already expected, a plug-and-play like add-on to Scratch.

As I explore further the whole experience of manipulating this dynamic surface of pins, it proves more and more inadequate, and the qualities ‘merge’, ‘embed’, ‘integrate’ are not met in this proposal of stage interaction. However, they became an evident goal to pursue in my next cycle of conceptual development, exploring with an improved prototype based on the learnings from this first attempt.

The staging interactions (output) was one of the two main interactions that I had aimed to explore. The building scripts interactions (input) was the other, and my exploration started off with observing and feeling the physical blocks of code and their palettes in my hands. Physical Scratch blocks were surprisingly playful in itself. They looked ‘cute’ and ‘fun’, and they had different shapes with bits sticking out from the edges that encourage grouping them up with other pieces intuitively. They are like candy to the eyes, and spark curiosity. You cannot help but, at least, fiddle with them. Having them on palettes, my immediate attitude was to place these side by side to get an overview of all that I had available. When using blocks from one of the palettes, I placed replaced the palette on the layout with the one currently in use, to mimic the original Scratch interactions. Physically, this interaction did not make sense. I did not need them all displayed at all times, and I certainly did not need to place the whole palette on the layout in order to then grab a block from it for use. It became unnecessary to maintain a space for palettes on the layout and so this area of the layout was dismissed to avoid redundancy that may cause unnecessary lost in content absorbance, as observed in the coherence principle mentioned in the TangiLearn report. As a result, this would be one element of the original environment of Scratch that would have to be changed in a Scratch environment for hybrid tangible interaction, and blocks of code would just be and handled manually like external objects, as seen in Osmo Coding. Naturally the only rule that I created for better efficiency was to place blocks back in their spaces on the palettes after use, whenever I could. It felt good to have some control and freedom of choice over how I prefer to organize my tools for learning to suit my learning necessities, instead of depending on a complete and universal given layout as Scratch provides us with today.

When handling the blocks, however focused I was on the interaction of drag and drop only, many other insights came to surface. Physical blocks present limitations that digital blocks do not. For instance, blocks are endlessly reused in Scratch. Once you picked up a digital block to drag it to the Scripts area, a replacement of that block is automatically generated in its palette. This would mean having various copies of every one of Scratch’s physical blocks. Another challenging issue is “C” blocks, for they need to be stretched to accommodate other blocks inside them. Added to this, it is also relevant to comment that some blocks need to have variables stated and it must be decided if the physical blocks can offer that or if there
is another way to add and to alter values. Last but not least, blocks must have a practical way to attach to and detach from their palettes – a sturdy feel and efficient mechanism that could be easily solved by magnetic contact such as in Osmo Coding.

My final remarks on this experiment are related to the Scripts Area. I noticed that depending on the length of the script, this area would not be big enough if speaking of a physical format. This area should accommodate multiple scripts and lengthy programs, just as in the original Scratch. Meanwhile, what I could do in this prototype to improve this shortcoming, knowing that it would not solve it, was to extend this area into the area that used to be for Palettes but was now taken out, making the Scripts area a little bigger (see arrow indicating change in Fig. 20). Although this does not overcome the limitation of fitting larger numbers of scripts in the area, it does start to slowly reshape the original interface of Scratch towards another, similar environment to suit a tangible hybrid concept.

![Figure 20 - Modified layout of the system, at the end of the prototype experiment. Source: Researcher's own archives.](image)

Conclusion from this prototyping experience:

The purpose of this prototype was to explore an initial concept in a very simple, solely physical format. In experiencing a tangible interaction with Scratch and the Maze Game, I felt more in charge of how I choose to organise my learning tools in order to perform the intended activity. I also felt that the possibility to pick any item to be my sprites and to ‘construct’ or ‘sculpt’ my background brought excitement to the activity which may be linked
to the creative potential that this contextual quality provides to the experience. As a tool for a next cycle of conceptual development and prototyping, this experimentation generated the following main insights:

- Pin surfaces are not a promising technology for this. The magnetic technical solution proposed by GaussBits, mentioned previously, might be a better way to go and must be explored instead.
- Blocks of code need ‘multiply’ in order to be reused in longer scripts, and to ‘expand’ when needed for accommodating other blocks inside.
- Palettes do not need to be on the work surface, they can be external, manipulated like colour swatch palettes.
- An A3 format is a good tabletop size for this activity so far into this study.
- Moveable sprites will need a base for reference and to be ‘moved around’ automatically by a future technology of choice. This points to a combination of technologies necessary in order to function as needed.
- A modular background for referencing (such as that of coordinates) to inform location specific values of the sprites in the scripts has been useful and should be considered in the next experiment.

4.4 Hybrid Experience Prototyping, in person and in video

Iterating on the first prototyping session, this second prototyping session consisted in producing and testing a hybrid Wizard of Oz – tangible and virtual - improved version of the tangible Maze Game prototype. The evaluation of this Wizard of Oz experience occurred both in person and in video (used to reach out for physically distant participant’s input).

Participants were divided in two groups. One is the group that experienced the prototype in person, performing the full Maze Game in the hybrid prototype, with the computer feedback faked by myself as I explain later in this section. The other group consists of people who experienced following a Wizard of Oz demonstration of the experience visually, by watching a video recording of it and reflecting on the concept based on their professional experiences with computing education and programming. In the process of producing and evaluating the experience, I also experienced the interactions myself.

This section presents the development of this prototype, from ideation, through planning and experiencing, to results. The video shared with participants is available on YouTube, through the following link: https://youtu.be/l1mzhhHOQLg.
Figure 21 - Brainstorming notes of scripts processes and mechanisms for Wizard of Oz Demo Prototype (Prototype 2). Source: Researcher’s own archives.

Figure 22 - Brainstorming notes of stage processes and mechanisms for Wizard of Oz Demo Prototype (Prototype 2). Source: Researcher’s own archives.
The planning started with brainstorming on scripts and stage processes and mechanisms (see Figs. 21 and 22), and a sketched script of interactions (see Fig. 23), aiming at repeating what worked and modifying what did not work in the previous, tangible prototype experiment. The instructions of the Maze Game remained the same (see subchapter 4.3.1), slightly adapted to a hybrid reality, where the interactions now include a screen and an inbuilt Stage system inspired on GaussBits. The following list of materials was used to build this prototype:

- A script with the sequence of interactions used to perform the Maze Game in the original Scratch, but adapted to the proposed modified system.
- A stage area with a modular grid as a permanent background.
- Tangible objects to represent sprites, and ‘reference’ bases (metallic) to fake an attachable reference tracking and mobile system, with the use of magnets for motion.
- LEGO bricks for building the maze background (conceptually, it could be any object).
- Palettes of blocks of code to mimic the palettes tab of Scratch, with detachable blocks – the same palettes used in the previous prototype, but with laser-cut wooden blocks, not just paper cut-outs of blocks.
- A sequence of frames of a modified Scratch environment, illustrating what the screen would look like when reacting to each step of the sequence of interactions in the script.
- A computer screen to use as a reactive digital table.
- A keyboard.
- Strong magnets to fake automated movement of sprites inside the Stage area.
- A camera and a tripod (for later video recording of Wizard of Oz)
- A constraint to keep in mind: although manipulated during programming interaction, all scripts must run by themselves, meaning that the Stage must still react to them automated or through user’s input with keyboard or mouse interaction (which might become other interfaces in future iterations of this prototype).

I then began to experiment with Scratch’s original screen environment, given that it would be adapted to support the new system and its interactions. This second layout presented more modifications in comparison with that of the previous prototype, as a result from reflection-in-action in making itself and with learnings from the previous one. The goal was to produce a frame by frame sequence that would mimic every user input and computer response at each stage of interaction with the hybrid system. By doing so, whenever a user registered a Sprite in the system, for instance, the computer screen would change accordingly but not through a real computer program but a fake one, manipulated by myself. The focus was once again on the main interactions in the Stage and the Scripts areas, making them tangible, with other support Scratch features kept on the screen environment that senses the tangible interactions and react accordingly. The expectation was to find a balance between tangible and virtual that collaborate in a unified manner, and ‘complementing each other meaningfully’ as Tsong, et al. (2012) suggest.

Here are some clarifications to specific aspects of this prototype. Firstly, the blocks of code used were the same as in the original Scratch not just because it is successful as a block based language in terms of education, but especially because too many learners are already accustomed to them as they are. Hence, maintaining their visual identities helps learners to naturally relate to the hybrid system. This also makes communication between these and novice learners easier, for experienced Scratch users – especially educators – can then immediately use their knowledge to help others less acquainted. Secondly, I am not addressing the curricular learning aspect of Scratch; I am challenging the interactions that it requires and allows for in the learning of programming. Thirdly, not only code blocks, but also most of the environment was kept the same, for the same reasons of future compatibility (keeping in mind accessibility to all) and the aim or achieving a result that is still recognizable as Scratch – just another version of it.
4.4.1 The resulting Prototype

The resulting prototype is a Wizard of Oz hybrid version of the Maze Game. With this prototype I experienced the interactions of this hybrid concept by performing the sequence of instructions for the Maze Game using screenshots of the intended outcome on a regular Scratch (see Appendix B) as guide of scripts to build and of how the maze and sprites should be placed on the stage. The goal of this activity was to create a maze in with 2 characters, HIPPO and CAT, placed on opposite corners of the maze, and HIPPO has to get to CAT avoiding the walls of the maze. If the walls are touched, a sound is played, but the character is not affected in any way. As HIPPO reaches CAT, a ‘cat meow’ sound is played. The layout’s main areas are the Sprites area (top left) and the Stage area (bottom left), and on the right side are the editing areas of Backdrop, Scripts and Sound, divided in tabs that can be individually activated by tapping on them. The differences between the original and the hybrid versions of layout are presented in Figures 24 and 25.

![Figure 24: Original Scratch screen layout with intended modifications for Prototype 2-Wizard of Oz Demo. Source: Researcher's own archives.](image-url)
What follows is a stage by stage description of my experience with this prototype as I performed a full Maze Game in this hybrid Wizard of Oz format. The images presented are Screenshots taken from the Hybrid Experience Prototype video, which was shot as I performed my experience and later showed to participants as a demonstration video.

Stage 1 (see Fig.26) - Creating backdrop, with LEGO bricks on the Stage area (bottom left hand corner).

Instructions: a) The Backdrops tab is activated for editing; begin by typing a name for the new Backdrop. b) Create a new Backdrop with tangible blocks, by placing them on the Stage.

Stage 2 (see Fig.27) - Creating Sprites with toy characters, and attaching reference and tracking base pieces to them, to add them to the system by placing them on the Sprites area and giving them names (top left hand corner). Having done this, they can be placed on the Stage, where the background (maze) is built.
Instructions: c) Select Sprites you wish to add to your project. d) You will need a Sprite Ref base for each of your Sprites. e) To start adding sprites, activate the Sprites area by giving it a tap. f) To add a Sprite, attach a Sprite Ref Base to the bottom of the Sprite and place it inside the Sprites area for the system to acknowledge it. g) Use the keyboard to give Sprites names. h) Repeat these steps for every new sprite.

Stage 3 (see Fig.28) – Activating Scripts area, to start building Scripts for a Sprite. The Scripts area is activated by tapping the Scripts tab. The scripts area is where blocks of code will be positioned to record scripts for that Sprite. Each Sprite has its own Scripts area, activated by ‘tap-selecting’ the sprite of choice in the Sprites area.

Instructions: i) Switch on the Scripts tab. j) Select HIPPO to give it some Scripts.
Stage 4 (see Fig. 29) – Building Scripts using Blocks of Code, creating a program of the Maze Game for the hybrid Scratch to perform when played by the user. To build Scripts, the user must pick Blocks of Code from the palettes and place them on the Scripts area, forming Scripts. Once a Script is complete, the user records it in the digital system by ‘tap-stamping’ it (touching the screen with the tap of a finger). This action creates and saves a digital copy of the assembled blocks, so that these same blocks can be used for new scripts or simply be placed back on their palettes. The user can then also alter the scripts digitally through touch screen and keyboard interactions, changing blocks, adding values, deleting blocks, and so on.
Instructions: k) Take Blocks of Code from the Palettes l) Place Blocks on the Scripts area, forming Scripts. m) Once a Script is built, tap the screen to create and save a digital copy of that Script. n) Once digital, the Script can then be edited digitally through touch screen and keyboard interactions. Blocks can have values changed and be deleted, for instance. o) “C” Blocks must be taken apart to accommodate other blocks inside. p) Use the same rules to create Scripts for other characters.

Stage 5 (see Fig. 30) – Playing the game, once the program for the Maze Game is complete with all its Scripts recorded, Sprites in Starting positions and Background in place. This Maze Game is programmed to be played by using the arrow keys on the keyboard to move HIPPO up, down, left or right, to get him to CAT, on the opposite corner of the screen. To start playing, the program must be activated as it also is in the original Scratch, by tapping on the Green Flag, and then at the end of the game, tapping on the Stop sign next to it to stop the program from running.

Instructions: q) Activate the program by tapping the Green Flag. r) Play the game using the keyboard arrows to move HIPPO up, down, left or right, to get him to CAT. s) Stop the program from running by tapping the Stop sign next to the Green Flag.

4.4.2 Reflection in prototyping

Experiencing this prototype has led me to make many reflections regarding this concept’s role in the teaching of programming concepts, its materiality and the challenges and possible solutions for its implementation as a future final product. In this section I present my own reflections in prototyping this hybrid Wizard of Oz Scratch.
The first thing to notice in making Scratch tangible, is that although the on-screen environment must be adapted to the new interactions, many features can be kept untouched. There can be subtle graphic (visual) feedback from the work pad, such as those that confirm the activation of a window, or of buttons. Some graphic feedback were already tested out in Prototype 1, such as the imprint of an object placed on the screen, to indicate that it has been ‘sensed’ by the system, representing a dialogue between user and computer, establishing a shared language of communication. Another example is the graphic representation that is helpful is that which indicate the activation of Sprites and Backdrop areas, and Tabs, to inform the user of the specific features of Scratch they are interacting with – if building Scripts, then they must activate the Scripts area; if adding a Sprite, they must activate the Sprites area; and so on.

In this hybrid version of Scratch, tangible blocks and features of the interactive screen based interface, all work in a synchronized manner, and all may have a purpose of either input or feedback, or both, depending on the interaction. Sometimes a physical sprite can have a purpose of informing the computer its presence, while at other times it has the purpose of giving feedback by automated motion according to a given script. The existence of the interactive screen based environment is beneficial in the following aspects: by means of visual feedback to inputs; by providing an array of features that would be too large, or technologically impossible to cover through tangible media; and by providing a supportive background to the main interactions of programming, of building Scripts and testing Scripts. This last point is important, in a sense that the focus of this interaction – learning to code - is at the forefront of the system, placing all other supporting features on the background, and prioritising focus on the learning of programming concepts. Moreover, physically having the system in two different layers, distributes its features in a way that it may avoid the feeling of being overwhelmed with options, facilitating its use and avoiding frustration.

Audio feedback contributes to the multidimensional aim of this project. Scratch already has this option with a palette of blocks called Sound and an editing space specifically dedicated to this type of media. For a more immersive experience, speakers could be strategically placed around the screen, to either give a more directional feedback in programming (interaction with scripts area) or in testing a program (interaction with stage). For the Stage, having grid of coordinates as a backdrop can be very good for moving objects to immediately get information of and to inform position to the system. As long as the system can inform the references of the blocks (sprites of backdrop) on the screen to use this information of position in the Scripts, for sensing purposes, having the grid available can be optional.
In the original Scratch, the blocks of code multiply infinitely. Once a block is picked up from its palette, it then drop in the Scripts area, that block is immediately replaced by the same block, for use in future scripts. Added to this, some blocks, such as ‘C’ blocks, can stretch to accommodate other blocks inside. The experience with using physical blocks has showed that as long as they can be reused and can adaptable to accommodate other blocks, it is a valid idea. To solve the problem of reuse, I used what I call ‘Tap-Stamp’. Tap stamp is a term that I invented to refer to the act of tapping the Scripts area make a digital ‘copy’ of the blocks placed on it. Once tap-stamped, the physical blocks used for that script can be reused, or placed back on their palettes. This digital copy of the script can then be modified, digitally, to delete, change values, add blocks, etc. This solution solved two main problems. Firstly, it allows blocks to be reused as many times as needed, because stamping them means that they are free for use somewhere else, and secondly, stamped blocks can have their values edited by tapping on the value and changing it with the keyboard.
As for the adaptability aspect of certain blocks such as “C” blocks, I tested this by breaking them apart (see Fig.32) at the points where adaptation (stretching) occurs. This was an effective solution for an experience prototype, but not for a fully functioning one or for a final product. It needs improvement for better control of manipulation, because dividing them in pieces requires so much from fine motor skills in manipulating them, that it becomes inefficient and frustrating, due to lack of control and of precision in accomplishing the intended task, possibly affecting levels of motivation and learning. Osmo Coding\textsuperscript{19} provides us with a viable solution to this, with their system of blocks that snap together magnetically. Using magnets at the tips of these blocks and of the blocks that they adapt to, ensure that the user has better control of movement in the interaction with blocks, when building scripts.

This is the advantage of working together with the digital environment of Scratch, in favour of expanding possibilities in learning, making up for what the physical materials cannot help us to accomplish by themselves. Having this ‘screen top’ hybrid interaction also points out to full body, multiplayer interaction, in which play happens from different directions in relation to the screen.

\textsuperscript{19} A system that combines tangible coding blocks and a virtual coding environment for children to learn coding. (Tangible Play Inc., 2013)
Manipulating objects for building the Background, for creating and preparing Sprites, piecing scripts together, was in fact an overall engaging and inspirational experience, as much as it was truly useful in planning out and producing a program. It gives a lot of freedom to experiment with different possibilities and outcomes before even building scripts, allowing several iterations of project ideas due to immediate visualization of outcomes via manipulation of the different elements of the project.

I wish to make a point of tangibles and digital technologies working together to make up for the physical challenges of the tangible world which often impose limits to certain curricula. Computers are made for processing data in ways that we cannot in the physical world. To fully learn computational thinking requires the embodiment of this understanding through interactions that can only be digital or else they are not possible. To take away the digital aspect of programming is to take away the exact importance of computers in our lives, that is: to perform as we cannot as humans, and to perform what the physical world cannot perform.

4.4.3 Experience prototype sessions with participants

The experience sessions were conducted in different venues, with the different participants (see Fig. 33), in this distribution:

- Ian and Simon, who teach programming to youngsters. Simon runs a Code Club at the local library;
- Martha who is a teacher, has taught programming to young children, and is knowledgeable in SEND education;
- Andy, who is a software developer and creative technologist.

Figure 33 - Participants of the Hybrid Experience Prototype sessions, Ian and Simon (left), and Andy and Martha (right). Source: Researcher’s own archives.
One full session consisted of the following sequence of events (see Figs. 34 and 35):

Part 1 – Presentation

Presenting the project, the experience prototype and its purpose, and the structure of events to take place in the session. This information was presented in a Form of Consent sent to participants a few days before the sessions, and refreshed by me at the beginning of the sessions.

Part 2 – Building and playing the Maze Game with the original Scratch.

For this, participants were given a notebook and a mouse. On the computer, two tabs opened. One was blank and ready for a new Scratch project. The other had a sample of the original project for consultancy. For extra support, a printout of what the project should look like at the end was handed to the participants.
Part 3 – Building and playing the Maze Game with the hybrid Scratch.

For this, participants were previously presented print-outs of Figures 24 and 25, to introduce them to the new visual environment that they would encounter. After this, the manipulatives – physical palettes with blocks of code, Sprites, LEGO bricks and reference bases for tracking and moving Sprites -, were also presented. Having introduced the elements of this hybrid version, they were given a printout of the script that I had used for my experience of the hybrid prototype. The script was given as the screen-shots presented in Figures 25-29. The participants were instructed to try out the sequence on the given script.

Part 4 – Discussion.

Firstly, participants were given the opportunity to share freely any ideas about the experience. Further discussion was built up on their comments, to answer the research question, as I encouraged the flow of ideas through spontaneous comments and questions.
4.4.4 Experience prototype video sessions with participants

The experience video sessions were conducted in different venues, with the different participants, in this distribution:

- Two Coder Dojo mediators, who teach programming to youngsters (in person);
- Catherine, who supports schools with computing and is a specialist in SEND education and the adaptation of the curriculum to these special learners, and a researcher of computing in these settings (Skype session);
- John, who is a teacher of computer programming for several years, and currently volunteers at an afterschool Code Club (Skype session).
- Two of John’s 12 year old students at the afterschool Code Club (feedback given to John, in person, and transferred to me through Skype, by John).
- Three teachers, members of the Computing at School (CAS) community (in person).

Observation: I was given 15min of the meeting’s agenda to go on stage and present my research to six teachers who were present. Of the 6, 3 participated in discussions.

One full session consisted of the following sequence of events:

Part 1 – Presentation

Presenting the project, the experience prototype and its purpose, and the structure of events to take place in the session. This information was presented in a Form of Consent sent to participants a few days before the sessions, and refreshed by me at the beginning of in person or Skype sessions. This document also had a link for access to the Hybrid Experience Prototype video.

Part 2 – Watching the Hybrid Experience Prototype video while reflecting

Having agreed to the terms in the form of consent, participants were then asked to watch the video and to, meanwhile, reflect on the purpose of this concept, on its interactions, and on the research question (presented on the Form of Consent also). As mentioned before, in the beginning of section 4.4, the video sent to participants is available on YouTube, through the following link: https://youtu.be/l1mzhhHOQLg.

Part 3 – Discussion

Firstly, participants were given the opportunity to share freely any ideas about the experience. Further discussion was built up on their comments, to answer the research question, as I encouraged the flow of ideas through spontaneous comments and questions.
4.4.5 Results from the Hybrid Experience Prototyping sessions

This section is dedicated to the results from experience prototyping for proof-of-concept through participants’ - including myself – experiences, reflections and opinions, to answer the research question. The sessions were all conclusive in a sense that they all result in the validation of the concept proposed in this study and in answers to the research question.

There are some, more general aspects of the activities themselves that I find interesting because they speak for engagement, which is an important aspect in learning both in isolation and in groups. First, I would like to make the overall remark that there is definitely an interest and openness, and a need, from the computing education community, in discussions related to approaches that may enhance their practices. This has become clear through the willingness of participants to try out my prototype and then to extend the session through 1h long conversations. The average duration of the discussions regarding the video sessions, in person or through Skype meetings, was 1h. The average duration of experience sessions in person, was between 1h30min and 2hs, of which 1 hour was also the average time of discussions. Their commitment to and need of improved computing education was also evidenced by how engaged they were during sessions, and the rich up-to-date knowledge that they brought to discussions, regarding educational tools, methods, initiatives, and so on.

From the general aspects to report from the experience sessions in person, first I wish to report the times for completion of Part 2 and Part 3 – trying out the Maze Game on the original and on the hybrid Scratch -, which were recorded to get a sense of how the modified version interferes in the time completion of tasks. For Ian and Simon, the original Scratch was a lot faster (6m50s on the original and 11m00s on the hybrid), while for Andy and Martha the modified Scratch was a lot faster (18m20s on the original and 7m15s on the hybrid). A second general observation was the collaborative engagement of participants in the activities. The four participants instinctively collaborated in both Part 2 and Part 3, whereas in Part 3 there was notably more dynamic and active participation from all participants. Thirdly – and from here on the general observations are related to both experience and video sessions -, although there is a common understanding of learning with tools such as Scratch being suitable for novice learners of all ages, the tendency to relate it to youngsters, when talking about it, is evident. Fourth and last, none of the participants questioned or made comments that questioned the relevance of this study, the structure of the session, or the type of prototype presented, and none made irrelevant or unrelated observations at any moment, without mentioning correctly and by their own initiative, that they were moving slightly off-topic. This, I believe, is a positive aspect to acknowledge, in relation to the solid ground on which participants weaved their observations throughout the sessions.
As important, or more, are the resulting discussions from all the sessions and participant’s answers to the research question. All presented in Table 1, according to the type of prototyping session, user session, and relevance of each of their remarks for each aspect of the integrative prototype that they experienced: look and-feel, role, and implementation. These aspects answer the ‘how and what are the implications’ part of the research question. From the table of results and as already mentioned, all participants (14, including myself) answered yes to the research question ‘Can we introduce manipulatives in the interaction with visual block based programming environments to enhance the learning experience at a foundational level and to improve accessibility’. Answers as to the second part of the research question ‘if so, then how and what are the implications for interaction design (in Look+Feel, Role and Implementation)’ are many, as shown on Table 1 (parts 1 and 2). Note that ‘Researcher’ on the table refers to myself - for Tangible and Hybrid Experience Prototype sessions -, motivated by my approach to writing the requirement specifications for the system not only by adding manipulatives, but also by questioning how this affects graphical interfaces, also regarding my own status of trying to make sense of computer programming through associations with tangible objects.

Look and Feel is the largest and most diverse group of inputs from participants. My most important remarks include having sound feedback for an immersive learning. This system could be played with wireless headphones for a more immersive experience. In relation to the handling objects and interacting with the screen interface, having better control and precision of movements seem to motivate learners because they can achieve results more efficiently. Regarding the manipulation of objects, participants reported that it felt more personal to touch pieces and be able to personalise the set-up around the screen as he/she wishes, instead of being given one predefined environment with a predefined work area for a ‘universal’ user. This seems to be important for the support of individual cognitive patterns. Comments also revolved around the child-like visual aspect of the objects and the environment, possibly because Scratch was tailored for the use of children, and that considering that the end users are young, the blocks of code could present symbol support like in Scratch Jr. (a version of Scratch for very small children), but that this could also appeal to adults if the objects and visuals had a less playful and more serious character. Participants had all sorts of extra ideas of how this new system could be further developed. Some mentioned that this could become a virtual reality system for learning programming in an immersive way, because watching characters move on the screen made it look as if they were alive. This also accounted for their ease in following a program by seeing the objects move around the Stage. The feeling of ease with dealing with block based programming through tangible interaction also brought up the idea that making the Start and Stop buttons could also be a real button, or that this interaction could be performer by pressing the space
bar instead of the screen as it is today, because it feels more precise and dynamic. This also extends to comments on how it feels easier to repeat Scripts with physical blocks and stamping them, rather than redoing piece by piece as the original Scratch, once more because it feels more dynamic.

Another idea was that this could be fun with different sets of blocks that could be bought separately, like collections of LEGO bricks. For example, one could go out and by a HIPPO and a CAT character to play this game. This shows how much participants have extended their relationship with Scratch into a wider social-cultural context by experiencing Scratch in a tangible form.

Moving on to ‘role’, my initial delimitation was ‘to teach basic programming concepts in closer to real experience and to a larger audience’. If judging by participant’s feedback, this hybrid proposal has a lot more to offer, as many of them saw this as a concept that can facilitate learning by making ‘programming easier to follow and appealing because it brings programming alive through real objects on the stage, avoiding misconceptions about how the computer works, enabling collaboration and making abstract concepts more concrete’. Their feedback also reports that this system can help the learner in many ways: it can ‘help accessibility of content with no hidden menus (all readily available for use)’, ‘help with the learning by building on however many years of experience the learner has with the real world’, and ‘help people who are weary of computers (an ice-breaker). Added to all this, it can ‘bridge the popular progression in education, covering the Concrete phase, to move on to the Pictorial (original Scratch), to then move on to Abstract (code)’.

As for Implementation, many of participant’s ideas expand significantly my initial considerations - which included the need for ‘C’ blocks to expand, the need for a technology that can trace and automated move of objects on the screen stage. Participants’ experience say that a final product of this hybrid proposal would probably be costly, making schools choose not to invest on it, however this would not necessarily be made for schools, but the fact that not many would afford it can prevent me from making it accessible to many as intended. Although possibly not viable in schools, further possible materializations to this concept were suggested. One participant suggested that this be installed in public places such as libraries in the form of table top systems for the use of all visitors. Another suggested that this be an open source system to become available to many people through openly shared files and tutorials of how to build your own set of Sprites and Blocks of Code (which can be printed out in 3d printers or laser cut) along with the free download of the hybrid Scratch software. One educator warned me that this project should continue in the direction of being developed as an interactive screen, because, from her experience, desktop computers and laptops are becoming outdates in classrooms, extending to households.
Participants’ conclusions and reflections in their experiences of the prototyping sessions are presented in Tables 1, 2 and 3 below.

<table>
<thead>
<tr>
<th>PROTOTYPE</th>
<th>PARTICIPATION</th>
<th>Results from Experience Prototyping that relate to Look + Feel. Source: Researcher’s own archives</th>
</tr>
</thead>
</table>
| TANGIBLE Experience Prototype | 1 (1) Researcher | • Interaction is slower than on original Scratch. This may be a positive aspect for self-directed learning.  
• Interaction inefficient and imprecise.  
• Blocks of palettes look like candy, pleasant to work with.  
• The combination of physical and digital must provide an intuitive understanding of the combined environment. |
| HYBRID Video Prototype | 10 (1) John | • This tangible aspect makes it a lot easier to change things on Stage and on the Scripting area, because with the original Scratch, it requires a lot of 'sticks' to change things.  
• Seeing the objects move on the Stage according to the program helps to follow the program easily.  
• It is essential to have the coordinates on the Stage, being able to place objects on top to understand what that means.  
• Physically putting things together manually may be difficult for people with poor fine motor skills.  
• Blocks could have symbol support, like in Scratch Junior. |
| | (2) John’s Students (12 year olds) | • 'Cool!'  
• Likened to playing with Lego, saying that people start off with ordinary Lego (just blocks rather than dedicated to a particular model, e.g., Star Wars) and just aren’t sure what to do but gradually learn what they can make with it. |
| | (3) CAS teachers | • 'Fascinating!'  
• Seeing the real object moving in front of you is very helpful in making that connection with what you are doing (referring to programming) |

Table 1 - Part 1 of 2 of results from Experience Prototyping that relate to Look + Feel. Source: Researcher’s own archives
<table>
<thead>
<tr>
<th>Hybrid Experience Prototype</th>
<th>5</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Researcher</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) Ian and Simon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) Andy and Martha</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- There can be subtle graphic (visual) feedback from the work pad, such as those that confirm the activation of a window.
- The existence of the interactive screen based environment is beneficial in the following aspects: by means of visual feedback to inputs: by providing an array of features that would be too large, or technologically impossible to cover through tangible media; and by providing a supportive background to the main interactions of programming, of building Scripts and testing Scripts.
- Physically having the system in two different layers, distributes its features in a way that it may avoid the feeling of being overwhelmed with options, facilitating its use and avoiding frustration.
- Audio feedback contributes to the multidimensional aim of this project.
- The fact that when a block is taken from the palette it is missing from there, is interesting to take notice of what is in use.
- Lack of control and of precision in accomplishing the intended task, possibly affecting levels of motivation and learning.
- Manipulating objects for building the Background, for creating and preparing Sprites, piecing scripts together, gives a lot of freedom to experiment with different possibilities and outcomes before even building scripts, allowing several iterations of project ideas due to immediate visualization of outcomes via manipulation of the different elements of the project.

- Reminds me of VR.
- Having physical blocks may become like LEGO bricks - people buy a set of blocks, then they can buy other sets of blocks according to their evolution in the learning. This might make those making purchase decisions stay with the original version of Scratch.
- Could be good to give the user something that gives ownership, like the drawings created with the old 'floor Turtle' • Stamping blocks solves the problem of reuse of blocks.

- Having the menu of all of the options (palettes of code) in front of the user and being able to pick up the blocks and play is much more accessible than the tabs on the original Scratch and having to find the right section, scroll down, and the rest of it. Because you’ve escaped the screen, you can spread out.
- You can spread palettes and other objects how you want to – what makes more sense to you, whereas on Scratch it’s how MIT designed it. You can reorganize things to make sense in your own mind, so people who don’t think in linear ways can group things together, which would be beneficial. Could quite help people with dyslexia to make connections with things, organize thought patterns more clearly as well.
- Having this stamping option is really useful to repeat whole scripts, because on Scratch you usually build it piece by piece all over again if you wish to repeat it.
- Some of the buttons, like the green flag and things like that, you could have a separate set of easily touchable buttons, for starting and stopping. Just some of those, not all. Or the space bar as the stop/play button - it is easy to find.
- Background with coordinates: moving sprites around manually on the stage and getting immediate coordinate information, changing according to placement, is very good for helping kids understand how coordinates work.
- "I love the idea that you could programme the table to move things around... that’s... Than’t the fun bit!"
- For adults, this could just have a more serious look to it, or even theme, to fit this user group.

Table 1 - Part 2 of 2 of results from Experience Prototyping that relate to Look + Feel. Source: Researcher’s own archives
<table>
<thead>
<tr>
<th>PROTOTYPE</th>
<th>PARTICIPATION</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>TANGIBLE Experience Prototype</td>
<td>1 (Researcher)</td>
<td>- Teach basic programming concepts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Be accessible to more people than currently available visual environments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Provide better support for cognitive processes (Doog), by enhancing learning through tangible manipulation.</td>
</tr>
<tr>
<td>HYBRID Video Prototype</td>
<td>10 (John)</td>
<td>- Tool that makes programming easier to follow, because it's physical constraints do not give the same instant feedback that Scratch provides.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- being able to actually physically pick up a sprite and put it somewhere on the screen, moving things around and acting it out, is essential for learning, making abstract concepts more real, and making the learner think of how things work.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Helps all learners, not only special needs learners.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Could be an 'ice breaker' for adults who are not very confident with computers and have never done any coding before.</td>
</tr>
<tr>
<td></td>
<td>1 (John's Students (12 year olds))</td>
<td>- I think your idea of it being a complimentary alternative set of rules to get started is ideal. If you see it as transition between your physical blocks, the blocks on the screen and then going to text-based programming.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;It is amazing, how far my 17 year old students can get with some really strange ideas about how the computer works.&quot; They find strategies to work around difficulties which work for some time, until they encounter a very simple problem that they cannot solve because of misconceptions. The thing about your concept is that because there are physical things, even if on the screen, some of these misconceptions cannot happen.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- I think that tangibles can help learners to understand programming better because you are building on however many years of experience they have with the real world.</td>
</tr>
<tr>
<td></td>
<td>(3) CAS teachers</td>
<td>- Learning made easier, building on the experience that learners had.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- A tool for all ages to learn basic programming concepts and skills.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- A tool that enables multiuser interaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- A hands-on approach, physically and digitally.</td>
</tr>
</tbody>
</table>

Table 2 - Part 1 of 2 of results from Experience Prototyping that relate to Role. Source: Researcher's own archives
<table>
<thead>
<tr>
<th>HYBRID Experience Prototype</th>
<th>(1) Researcher</th>
<th>–</th>
<th>–</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) Ian and Simon</td>
<td>Yes</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) Andy and Martha</td>
<td>Yes</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- A tool for learning foundational programming at all ages.
- A tool for embodiment of programming concepts.
- A tool accessible to audiences of diverse needs and abilities.

- Appeals to people who have difficulty in one form or another, like in relating to things that are on screen.
- Could encourage parents who are not fond of screens to interact with this hybrid technology, to then have the knowledge to understand better their kid's interactions with programming, and facilitate communication on the topic.
- A way of producing interest in something that is screen based, for somebody at the start of the journey in learning programming.

- It is quite fun, for any age.
- Having tangible things to move around can help small children and people with special needs who struggle with the abstract nature of pictures on screen. They would benefit in the learning, by understanding things more quickly and being able to relate to this much better.
- Would help people who are weary of technology by working with a world that they know rather than the intimidating world of code.
- Having tangibles is good for learning through physical experimentation and asking "I wonder what this does?", or "what happens if I do that?".
- It would immediately engage people. Kids would be ble to put their favourite characters on immediately.
- The thing that really appeals is the magnetic approach (referring to movement of Sprites in the Stage of the prototype). That brings it alive in a way that, you wouldn't.
- No hidden menus. More tools and features are exposed to the user, facilitating understanding of how to interact with this environment better than the original Scratch.
- This hybrid version, together with a tactile map of its environment, could be very good for people with visual impairment.
- This hybrid system would help in the 'concrete' stage of learning - which progresses from concrete, pictorial (original Scratch) and abstract (code) - in computer science.

*Table 2 - Part 2 of 2 of results from Experience Prototyping that relate to Role. Source: Researcher’s own archives*
<table>
<thead>
<tr>
<th>PROTOTYPE</th>
<th>PARTICIPATION</th>
<th>Research Question (RQ): Can we introduce manipulatives in the interaction with visual block based programming environments to enhance the learning experience at a foundational level and to improve accessibility (Yes or No)? If so, then how and what are the implications for interaction design (in Look+Feel, Role and Implementation)? How?</th>
<th>Implementation</th>
</tr>
</thead>
</table>
| TANGIBLE Experience       | 1 (1) Researcher       | - Blocks of code need to “multiply” and “expand”.                                                                            | - Palettes do not need to be necessarily on the work surface, they can be external.
| Prototype                 |                        | - Palettes in this format tested, take up too much space.                                                                        | - An A3 format is a good tabletop size of work surface.
|                           |                        | - Moveable sprites will need a base for reference, to be ‘dragged around’ automatically (can be magnetic, for instance, or ‘suction cup’ to attach to any object) | - Modular background for location of Scripts.
|                           |                        | - Palettes no longer need to be displayed in the visual interface, since they are now physical.                              | - The program that runs Scratch needs to adapt to the needs of a tangible system. It is important that software and hardware development collaboratively in this.
|                           |                        | - I do not recommend using dynamic shape displays. They are hard to handle manually and have limited possibilities of interaction. | - Scripts area must accommodate multiple scripts and lengthy programs.          |
| HYBRID Video Prototype    | 10 (1) John            | - The rules must be different from Scratch (referring to procedures in the activity of programming)                           | - The making of code with physical blocks as it is now seems to present a barrier because of blocks such as “C” blocks that have to be better developed for a physical version. |
|                           | (2) John’s Students (12| - I worry about the C blocks and how they will work physically.                                                                | - puzzled by how it worked, and worried a bit about the hardware needed.     |
|                           | year olds)             |                                                                                                                               |                                                                                   |
|                           | (3) CAS teachers       |                                                                                                                               | - Would probably be costly to acquire a final product, and schools might not invest.  
<p>|                           |                        |                                                                                                                               | - Could become a system available on a smart table that could be in places like a library, where people could experiment with it. |</p>
<table>
<thead>
<tr>
<th>HYBRID Experience Prototype</th>
<th>(1) Researcher</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>(2) Ian and Simon</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Andy and Martha</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

-Although the on-screen environment must be adapted to the new interactions, many features can be kept untouched.
- For a more immersive experience, speakers could be strategically placed around the screen, to either give a more directional feedback in programming (interaction with scripts area) or in testing a program (interaction with stage).
- For the Stage, having grid of coordinates as a backdrop can be very good for moving objects to immediately get information of and to inform position to the system. As long as the system can inform the references of the blocks (sprites of backdrop) on the screen to use this information of position in the Scripts, for sensing purposes, having the grid available can be optional.
- ’Tap-Stamp’. Tap stamp is a term that I invented to refer to the act of tapping the Scripts area make a digital ‘copy’ of the blocks placed on it. Once tap stamped, the physical blocks used for that script can be reused, or placed back on their palettes.
- Magnetic snap for blocks: using magnets at the tips of these blocks and of the blocks that they adapt to, ensure that the user has better control of movement in the interaction with blocks, when building scripts.
- From a multiplayer perspective, I foresee the need to rethink the layout of the interactive digital interface to support multiplayer interaction.

- VR and AR could be a helpful technology to integrate feedback into the physical creations on the Stage. One could even use a mobile phone or this.
- This could be open source. Files for blocks of code and/or tracking reference devices could be printed at home or in makerspaces. Even sprites could be printed out in 3D easily.

- There's a lot of technical issues with the magnetic movement of Sprites on the Stage of the prototype, but fundamentally, there is no reason why it shouldn't work. You'd have to have probably a special tablet, or table.
- This seems to head in the direction that schools are heading, of using tablets, not screens, computers, laptops and mice.
- This could be very good for people with visual impairment if there was a tactile version of the screen and they could interact with both. You could put things on the tactile screen, that then repeat on the computer screen, so that you can become familiar with the layout. Then you could progress from that tactile to the screen reader of this hybrid version, and be familiar with where you're going.
- This could be open source 3D printed, then that would make it much more accessible to be used throughout the world.

Table 3 - Part 2 of 2 of results from Experience Prototyping that relate to Implementation. Source: Researcher's own archives
All of participants’ main comments are presented in Tables 1 to 6. From these tables of results we notice that experience prototyping seems to induce more diversified discussions than video sessions (roughly twice as productive). The total amount of comments made by the 4 participants in experience prototyping is roughly the same as the total amount of comments made by the 8 participants in the video sessions.

Participants focused on Role and on Look + Feel, more than on Implementation, and judging by the content of their comments on Implementation, I believe that this has two reasons: technical solutions for computer software and hardware is not a teacher’s primary activity and concern, so because most participants were educators hence this result; and participants did not seem concerned with the final making of such a product, aware that this is not the focus of this study, and keen on engaging in exploration of possible interactions.

I also identified reoccurring opinions spoken of by different participants in different sessions. These are: the progression of learning from physical blocks, to screen-based blocks, to screen-based code, and how this hybrid tool makes up for the transition from the first to the second step which is often difficult to make; the concern with how ‘C’ blocks will fully function in a final product; the possibility of an open source tool; the idea of this being a complimentary tool, not a tool of substitution to what exists; and many of the comments related to the role of this tool, especially those that speak of making sense of abstract concepts through immediate feedback and associations with the real world. Having different participants speak of the same potentials and qualities is a good indicator that this project is relevant for reasons alike.

5. REFLECTIONS

Much like the study of Xie, et al., where they compare three types of interfaces – physical, graphical and tangible (which is, in fact, a hybrid tangible-virtual type in their work) -, my study concludes that TUIs are beneficial for learning through the exploration of the graphical and tangible versions of Scratch, which I refer to as the virtual and the hybrid versions in my work.

In both theirs and my work, shortcomings with the user’s virtual interaction with an activity have presented positive outcomes when performed in a hybrid manner. In my study, one of the positive outcomes is collaboration. The use of a mouse means taking turns and when the mouse is substituted by a hands-on interaction, more users can participate and fully engage in the activity, and participation aids learning because students learn from one another by observation and communication. For collaboration, one participant also suggested that this hybrid system could be installed as a smart table (comment by CAS teachers, see Table 3 - Part 1 of 2), so that it could be accessed by multiple access points. However, this specifically could be a matter of software design and not necessarily call for tangible interaction, given
that the virtual environment could be made to fit this purpose on a touch screen smart table, for instance. Still, this would need - and is worth - further investigation. I assume that the three-dimensional proprieties of real objects placed on the Stage area of this table would still allow users to collaborate by means of exchanging physical pieces of a hybrid system in this tangible interaction, and this would probably be hard(er) to perform as a drag and drop movement from one side of the table to the other in order to pass these pieces to someone else. And if with one piece this would be difficult, imagine several.

In addition to collaboration, substituting a mouse interaction with a hands-on interaction has proven to be of preference to children, and one of the reasons for this is that using our hands gives us more control of movements and precision of performance. The more the user can control movements, more he can invent, do, accomplish and feel confident. Precision of performance avoids frustrations of unaccomplished intentions. These are factors that motivate learning and therefore ease the process. This is true also for having a touch screen base to the hybrid system, using its surface instead of just handling objects onto it. Added to this, it is important to keep in tune with how technologies are being used in learning for thoughts of future technologies, as well as present technologies. I remember Martha (participant in the Hybrid Experience Prototyping session) commenting on how small children nowadays present a lack of knowledge on how to use a mouse when placed in front a computer, and how this is related to the fact that the current choice of technology is that which allows us to feel and to use our hands, such as smartphones and touch screens, therefore making computer mice a secondary choice of medium.

Learners learn better by relating concepts to real-life scenarios. John (participant of the Hybrid Video Prototyping session) says that this hybrid version of Scratch mitigates misconceptions of how a program works. This, I believe is linked to the fact that there are less ‘hidden mechanisms’ in a physical object that turns on its axis, than there is in a two-dimensional image (such as Sprites on Scratch) that turns on its axis, for example. If a program tells graphics to turn twice 70°, the image will magically appear in its new position. If the same thing happens with a physical object, it will have to turn twice 70° and then stop at its final position, making the learner think "ah! This is what turn twice 70° really means and how it works". I would say this also gives the learner a kind of control over the act of programming, because concepts deeply understood can only result in more creative freedom and precision of intended outcomes. What this also does is that it makes the important learning of debugging in programming activities easier to follow and to perform, for mistakes are easier to spot. If an object turns twice 70°, and does not end up in the intended position, somewhere in the middle it might have been told to do something else that in the process of turning can be spotted for correction. In a 2D rotation, the process of rotation may happen in such an immediate way that the process it involves cannot be grasped by some learners.
The tangible layer of a hybrid system for learning can also ease interaction with computers when the learner is weary of technology. It is common for people who do not understand how computers work to place a barrier in between themselves and screens. Having a tangible bridge between user and hybrid system eases this difficulty not only as it provides something real to relate to, but most of all because, in the case of a hybrid Scratch, the tangible artefacts help to understand how the computer works.

The fact that puzzle pieces (the maze walls) moved when sprites collided with them can be an example of how real world cause and effect encountered in a tangible experience of programming, can influence the outcome of programs produced. This experience of collision, for instance, could have inspired a student to try to interpret this as a bug in his program, and go back to the scripts to try to write code that mimics that effect on the walls in case of collision.

My findings suggest that the use of tangible objects to interact with an existing software. In this case Scratch. This gives a learner of block based programming the option to interact and to explore Scratch from another perspective. This proposal is not intended for the substitution of the original Scratch, although it presents many reasons in favour of its use, rather it shows another way to interact with the Scratch environment and the lessons it offers on computer programming. Diverging from a creative educational programming tool that enables solely virtual interaction, or solely tangible interaction, this dissertation merges both in order to repurpose interactions according to necessity for learning and accessibility, establishing clearer design correlations than what exists today. It is just as important to physically experience a sequence of instructions, as it is to digitally communicate it to the computer, for in reality programming is an activity external to machines (we plan out a program with pen, paper, objects, people, thoughts) that we translate into a computer language, that is internal to machines. I find it as hypocritical to assume that programming education for novice learners does not necessarily need interaction with screen-based programming, as it is to assume that we can teach programming without linking the activity with the external world with which we interact. None is exclusive, but complimentary.

This hybrid proposal makes the learning of basic programming accessible to many, by making block based environments and content more accessible to users who are either not confident with computers, or who have special learning needs, without however excluding the digital component of this activity, rather offering a possibility to better relate to it and manipulate it. Added to this notion of widening the outreach of visual block based programming environments, this could also become a part of the open source movement, and have its system components – digital and physical -, available for download and ready for use in software development, laser cutter, 3D printers, and so on. This DIY possibility
would reduce the cost of a final product and minimize its inaccessibility to those in less favoured financial positions, as well as it would support a culture of knowledge sharing and distributed ownership. Such ownership is not just a making of assembling one’s own hybrid programming system, but also a making of using the system in which the experience of setting up a work area with physical palettes of blocks, or a Stage with Sprites that are ordinary objects from one’s surroundings. This hybrid system treats novice learners as individuals, not as components of a universal profile by a rigid learning system. It offers users the tools to make more choices in ‘how to communicate’ with computers.

This design process has shown me that the choice for a meta-learning perspective shared by educators produces many insights as to how design choices affect the process of learning. I did experiment with getting feedback from students (John’s students, as shown on Tables 1-6) in a Hybrid Video Prototyping session, and their ability to judge the relationship between activity and learning seemed, at a first glance, not productive. However, this could only be affirmed if repeated with more end users, and I do firmly believe that for further work end-users (learners) should be included in prototype testing. In this case, I wish to point out as I have previously mentioned, that this would require a fully functioning prototype, and this is also why my choice of process seemed adequate, for it has refined a concept before producing a fully functioning prototype.

6. CONCLUSION

This study provides reasons to believe that we can introduce manipulatives in the interaction with visual block based programming environments to enhance the learning experience at a foundational level and to improve accessibility. It presents the possibility for making computer programming a more relatable subject to the extent of human knowledge and experience through associations of physical nature. This is demonstrated by participant’s experienced opinions and supported by theories in TUIs and Learning, which point in the direction of a hybrid system for learning basic concepts of programming in environments such as Scratch, by all means has a way to enhance the learning experience and outcomes.

For designers of Tangible User Interfaces (TUIs) for learning and the computing education community, this research presents the concept of hybridity for computing education with empirical proof of concept, as presented from prototyping experiments, a topic that is not available in research yet today. I have not encountered this in academic material, but would very much enjoy to see it expand and hope that my contribution will be of use in this direction.

The learning that I also wish to share with the design research community is that of Experience Prototyping – in person and in video sessions - as a potent method and tool for
exploration and evaluation of conceptual ideas. Experience prototyping has enabled me to communicate an intended experience (through in person and video sessions), to provide an experience to others (through in person sessions) and to myself, in a strategic systematic and replicable form. Added to this, the design artefact of experience prototyping - the prototype in itself -, has served as a tool for tinkering, a tangible sketching material that I recommend to fellow designers.

For learners of foundational concepts in programming who are transitioning from Tangible Programming Languages (TPLs) to Block Based Programming Languages like Scratch, this study means that the IxD community is engaged in making their learning processes in computing easier and more enjoyable, and that there is a way to make abstract concepts graspable; and a way to make programming concepts graspable is through interaction with a hybrid system where collaboration and hands-on activity are the number one conditions.

For Scratch and similar learning environments, this study means that it has potential, but that this potential is based on its shortcomings, therefore for as much as Scratch is popular, it is not a perfect, and I even dare say ‘ideal’, educational tool for learners of foundational concepts in computer programming.
REFERENCES


Carlborg, N. & Tyrén, M., 2017. *Introducing micro:bit in Swedish primary schools: An empirical design research on developing teaching material for training computational thinking in Swedish*


De Oliveira, C. C., n.d. Designing educational programming tools for the blind: mitigating the inequality of coding in schools.


Gaver, W., 2012. What should we expect from research through design?. Austin, Texas, ACM, pp. 937-946.

Hankins, J. & et.al., 2015. Touch Ability Games & Puzzles Designed For People With Deaf-Blindness. RESNA ANNUAL CONFERENCE.


http://www.papert.org/articles/SomePoeticAndSocialCriteriaForEducationDesign.html
[Accessed 21 04 2018].

Available at: https://blog.playosmo.com/post/94655155207/a-peek-into-osmos-reflective-ai-technology
[Accessed 29 April 2018].

Available at: http://reactable.com/
[Accessed 12 08 2018].


Available at: https://www.schooleducationgateway.eu/en/pub/news_events/computer_programming_and_coding.htm
[Accessed 02 April 2018].


Scratchteam, 2013. Maze Starter, s.l.: s.n.


Available at: https://www.socialsciencespace.com/2018/01/coding-school-research-needed-computing-accessible-children/
[Accessed 02 April 2018].


Available at: https://spielgaben.com/
[Accessed 10 04 2018].

Tangible Play Inc., 2013. Play Osmo. [Online]
Available at: https://www.playosmo.com/en/coding/
[Accessed 28 April 2018].


Figures

Figure 1 - The web based environment of Scratch.
Source:https://portal.sd38.bc.ca/public/8zd3f5g5/Pages/scratch_interface.aspx (accessed on April 18, 2018) ...............................................................................................................................18

Figure 2 - Types of blocks of code, from Scratch. Source: Researcher's own archives.........................19

Figure 3 - GUI (left) and TUI (right). Source: Are tangibles more fun? Comparing children’s enjoyment
and engagement using physical, graphical and tangible user interfaces. (Xie, et al., 2008, p. 196)
...............................................................................................................................26

Figure 4 - The TangiLearn concept and experimentation. Source: retrieved from article Designing
Tangible Multimedia for Preschoolers based on Multimedia Design Theories (Tsong, et al., 2012).
Access on April 28, 2018. .........................................................................................27

Figure 5 - Fiducials on conventional toys, system setup, and single or multiplayer interaction with
NIKVision. Source: retrieved from the developer's official website

Figure 6 - Playing Coding Awbie, Osmo Coding blocks, and multiplayer interaction with Osmo Coding.
29, 2018.....................................................................................................................29

Figure 7 - Setup of shooting the Experience Prototype video. From left to right: laptop with tangible
objects on screen, ready for building Scripts; tripod, camera; 4 LEGO pieces, small magnets,
doughnut shaped metallic 'Ref base'; palettes, sticky rubber for attaching LEGO's to the screen,
pile of LEGO. Source: Researcher's archives. ................................................................36

Figure 8 - Overview of the design process. Source: Researcher's own archives...............................37

Figure 9 - Researcher’s own experimenting with micro:bit. Source: Researcher's own archives. ..........38

Figure 10 - A group of people interact with a Reactable system. Source: still frame retrieved from the
video 'Reactable live performance #1', in

Figure 11- Stage and Scripts areas in Scratch. Source: Researcher's own archives. ............................41

Figure 12 - Recorded notes and sketches from brainstorming for prototyping the initial design
concept. See Appendix A for more notes and sketches. Source: Researcher's own archives. ......42

Figure 13 - Tangible Experience Prototyping. Source: Researcher’s own archives ............................43

Figure 14 - InForm experimentation. Retrieved from the MIT Media Lab official webpage, on

Figure 15 - Magnetic sensor grid, tangible flight simulation, Sensed magnetic field image, and
magnetic unit that goes inside the toys. Retrieved from the article GaussBits: Magnetic Tangible
Bits for Portable and Occlusion-Free Near-Surface Interactions (Liang, et al., 2013, p. 1391).
Access on April 29, 2018. ............................................................................................45

Figure 16- Part 1/2 (a and b) of the sequence of interactions to programme the Maze Game on
Scratch. Source: Researcher’s own archives. ....................................................................46

Figure 17- Part 2/2 (c, d and e) of the sequence of interactions to programme the Maze Game on
Scratch. Source: Researcher's own archives. ....................................................................46

Figure 18- Initial layout of Scratch on A3 paper and remaining Palettes of blocks, per category. Source:
Researcher's own archives. ..............................................................................................47

Figure 19 - ‘Stage’, in a tangible experiment with a dynamic pin-display. Source: Researcher's own
archives ...................................................................................................................................48

Figure 20 - Modified layout of the system, at the end of the prototype experiment. Source:
Researcher's own archives. ...............................................................................................50

Figure 21 - Brainstorming notes of scripts processes and mechanisms for Wizard of Oz Demo
Prototype (Prototype 2). Source: Researcher's own archives. ...............................................52
Figure 22 - Brainstorming notes of stage processes and mechanisms for Wizard of Oz Demo Prototype (Prototype 2). Source: Researcher’s own archives. ................................................................. 52

Figure 23 - Sketch of script for Wizard of Oz Demo Prototype (Prototype 2). Source: Researcher’s own archives. .................................................................................................................. 53

Figure 24 - Original Scratch screen layout with intended modifications for Prototype 2 - Wizard of Oz Demo. Source: Researcher’s own archives. ................................................................. 55

Figure 25 - Modified version of Scratch’s screen layout, for sequence of frames to be used in Prototype 2 - Wizard of Oz Demo. Source: Researcher’s own archives. ................................................................. 56

Figure 26 - Stage 1: creating Backdrop. Screenshots taken from the Hybrid Experience Prototype video. Source: Researcher’s own archives. ................................................................. 56

Figure 27 - Stage 2: creating Sprites. Screenshots taken from the Hybrid Experience Prototype video. Source: Researcher’s own archives. ................................................................. 57

Figure 28 - Stage 3: activating Scripts area. Screenshots taken from the Hybrid Experience Prototype video. Source: Researcher’s own archives. ................................................................. 58

Figure 29 - Stage 4: building Scripts and recording them digitally. Screenshots taken from the Hybrid Experience Prototype video. Source: Researcher’s own archives. ................................................................. 58

Figure 30 - Stage 5: Activating the program and playing the Maze Game. Screenshots taken from the Hybrid Experience Prototype video. Source: Researcher’s own archives. ................................................................. 59

Figure 31 - Stage with grid of coordinates as a backdrop, and Sprites HIPPO and CAT (bottom right of Figure); Sprites area (bottom left of Figure); and Tabs areas, such as Scripts (top of Figure). Source: Researcher’s own archives. ................................................................. 61

Figure 32 - Physical palettes of Blocks of Code. Source: Researcher’s own archives. .................................................................................................................. 62

Figure 33 - Participants of the Hybrid Experience Prototype sessions, Ian and Simon (left), and Andy and Martha (right). Source: Researcher’s own archives. ................................................................. 63

Figure 34 - Hybrid Experience Prototype session with Ian and Simon. Shots taken from Parts 2 and 3 of the structure of the session. Source: Researchers own archives. ................................................................. 64

Figure 35 - Hybrid Experience Prototype session with Andy and Martha. Shots taken from Parts 2 and 3 of the structure of the session. Source: Researchers own archives. ................................................................. 65
APPENDICES

APPENDIX A: Recorded notes and sketches from brainstorming for prototyping the initial design concept, corresponding to subchapters 4.2 and 4.3.
Block Based Coding Environment

Non-Visual Block Based Coding

Do an "Algorithm" of interactions

Sequence

Isolate prototype

Show how + why I decided to go for

My reactive pool, based on reactive tables.

(reactive tables + reactive blocks)

(less blocks + less clutter + easier to read)

Same issues as getting lost in drag-n-drop.

Boolean

REPORTER

C Blocks

Cap

Simon Says

Analogies

- Doll/Person
- "Tree"
- "Recipe" (like a cooking recipe)
- Game

In Real Life

- Interactive tactile maps
- Tactile Graphics

Why block based, not text based?

How does it benefit over text based?

What are the advantages?

How does manipulation contribute to the learning of

Computer based visual/interpretation
Good \rightarrow Tangible Multimedia

What
Goal \rightarrow Tangible Scratch

How
I can achieve it \rightarrow Translating it to
(Transferring)
Tangible Multimedia

What I need to do to achieve this \rightarrow Explore tangible possibilities for Scratch

Tangible Multimedia in Scratch:

Stage
Scenarios

1. Draw a figure
2. Beach/skate figure
3. Figure moves around stage
4. Figure moves out of stage
5. Place an existing character on stage

Try 3 sets of this and photograph steps of action

Take notes of insights:

What works and what does not work

Create
Test (the best idea(s)) with others!
**APPENDIX B**: Screen shots taken the game on the original Scratch to use as a guide of placement of sprites and maze walls on the stage, and of how the scripts can be built accomplish the game's goals.

Full Script for HIPPO.
Full Script for CAT.
APPENDIX C: Observation of Maze Game construction (section 4.3.1). Source: Researcher’s own archives.
APPENDIX C: Overview of the full sequence of screenshots taken from the hybrid experience prototype Wizard of Oz video). Source: Researcher’s own archives.