# Fostering Computational Thinking skills with Tangible User Interfaces

Tommaso Turchi

*January 18, 2019*
Version: 1.0

Brunel University London

Department of Computer Science
College of Engineering, Design and Physical Sciences

This dissertation is submitted for the degree of Doctor of Philosophy

# Fostering Computational Thinking skills with Tangible User Interfaces

Tommaso Turchi

*External Reviewer* Dr. Andrew Fish
Centre for Secure, Intelligent and Usable Systems
University of Brighton

*Internal Reviewer* Dr. Alan Serrano-Rico
Department of Computer Science
Brunel University London

*Supervisors* Prof. Alessio Malizia and Dr. David Bell

January 18, 2019

**Tommaso Turchi**

*Fostering Computational Thinking skills with Tangible User Interfaces*

This dissertation is submitted for the degree of Doctor of Philosophy, January 18, 2019

Reviewers: Dr. Andrew Fish and Dr. Alan Serrano-Rico

Supervisors: Prof. Alessio Malizia and Dr. David Bell

**Brunel University London**

College of Engineering, Design and Physical Sciences

Department of Computer Science

Kingston Lane, Uxbridge

UB8 3PH and London

Words are, in my not-so-humble opinion,
our most inexhaustible source of magic.
Capable of both inflicting injury, and remedying it.

— Harry Potter and the Deathly Hallows

To Mum and Dad, my greatest source of inspiration.

# Abstract

Given how technology surrounds our whole life, learning to code is becoming more and more crucial for the general public: think for example of the amount of software involved in managing a flight, or when you just turn on the engine of your car. People want to play an increasingly active role in their life and there is already evidence in an overall heightened interest in coding from the many successful public initiatives aiming at introducing coding skills to a wide audience. Nonetheless, coding skills are not just about programming but require an ability of problem-solving, abstraction, pattern recognition to name but a few; in a word, the so-called Computational Thinking (CT) skills, namely a set of thinking skills, habits, and approaches that are integral to solving complex problems using a computer and widely applicable in today's information society.

Due to this sudden global interest in promoting CT skills to many broad and diverse audiences, several tools and methods have been designed with the aim of supporting the introduction of programming concepts in more effective and less daunting ways than the past. A popular theory of learning that can come to the aid on this matter is Piaget's constructivism, which argues that people produce knowledge and form new meanings based upon their experiences in the real world and social interactions. Thus, exploiting human's natural ability for objects manipulation in the physical world and its afforded interactions could be an effective way of supporting users in learning abstract concepts such as the ones underpinning CT.

Tangible User Interfaces are an interaction paradigm that was devised to foster collaborative learning and exploit humans' natural dexterity for physical objects manipulation to provide an easy to use interface that can be used even by inexperienced people. They exploit the physical world to offer a concrete representation of the abstract concepts learners usually struggle with and thus

employing them to teach those concepts underpinning CT might represent an effective and engaging way of supporting the learning of such skills.

This thesis investigates this claim through the development of a software platform combining its digital and physical features to promote CT skills in different domains. The platform design is informed by a review of related work, a workshop with domain experts, and was validated through a series of studies in different application scenarios which reported promising results in terms of CT support.

# Acknowledgement

The research reported in this thesis would not have seen daylight without the assistance, patience, and support of many sometimes unwitting individuals, who I feel the urge to thank one by one here.

First off, my ever tireless supervisor, Prof. Alessio Malizia, who helped me throughout the entire process and pushed me to always nurture my curiosity. Spending time and enjoying an unhealthy amount of coffees with him while listening to his stories was an extremely enriching process that I will always be grateful for.

My second supervisor, Dr. David Bell, who supported me many times throughout the journey, and — together with Prof. Terry Young — helped me feeling welcome in the Department from the first moment I joined.

Prof. Alan Dix, who lead me on the far-distant shores of Tiree and deeper within my research and objectives.

Boyce and my office mates in STJN129, as well as my work collegues in WLFB151, for all the successes, sorrows, and food leftovers we shared over the past years.

Then, on with some more personal notes: to my Mum and Dad, for their continued support at distance and for always being there anytime I needed to get back to recharge my batteries. To Giulia, for standing beside me all along the way and conforting me when I needed the most. To my sister Sara, her husband Samu, and my two precious nieces Matilde and Margherita, for always trying to lit up my mood, no matter what.

And finally, to all my friends at home, with an honourable mention to Alessio, Fabio, Vieri, and Antonio, for making me feel like I never left.

All this won't be forgotten.

# Contents

# 1 Introduction

> *You can't think seriously about thinking without thinking about thinking about something.*
>
> — **Seymour Papert**
> Mindstorms, 1980

This chapter introduces the research carried out in this thesis by outlining the context in relation to Computational Thinking (CT), the addressed Research Questions, and the adopted methodology.

## 1.1 Research Context

Today's society is deeply permeated by technology and it is unmistakably clear how much it affects people's lives. From the vastly complicated range of software involved in managing a flight, to the process of booking that same flight online.

Computer Science (CS) drives jobs growth and innovation throughout economy and society, and computing jobs make up half of all projected new occupations in Science, Technology, Engineering and Mathematics (STEM) fields between 2016 and 2026 according to a recent study by the Bureau of Labor Statistics [@16a]. Right now there are more than 500,000 open computing jobs in the United States alone, and CS is the second highest paid college degree [@18h]. In spite of that, only 8% of STEM graduates study CS [@16b], and only 40% of schools in the U.S. teach it to K-12 students (i.e. from kindergarten to 12th grade) [@16c].

Another research study [MK07] showed that women who study CS in high school are 6 times more likely to major in CS than those who do not, while Black and Hispanic students are 7 or 8 times more likely to do so. This highlights how the diversity problem in tech starts in schools and the primal role of education in supporting the learning across different disciplines and fostering the participation of more and more people to the technological revolution.

To this end, being able to properly support learners in understanding and trusting algorithmic solutions found in computational systems — and thus participating in the design and development of such solutions — can bring

several benefits in everyday life, making them able to succeed in today's complex and technological society [Bun07].

Such support can be achieved by creating the socio-technical conditions for empowering users, as problem owners, to participate in the evolution of such systems [FG06]. In particular, End-User Development (EUD) methods [LPW06] seek to enable end-users (i.e. any computer user) to enjoy some of the computational power that only professional programmers can exploit, and thus can contribute to fostering the technical conditions for such participation. These methods are useful not only in traditional information systems [DHP07] or spreadsheet-based applications [Bur09], but also for tailoring personal devices [DP12; Fra+16] or smart environments [Cab+16; CC16].

### 1.1.1 Computational Thinking

People will always strive to play a more active role in their life, thus programming is becoming an essential skill to master for the general public, resulting in an overall heightened interest in coding. Take for instance the Hour of Code [@18e], a successful global initiative organised by Code.org (a non-profit organisation founded in 2013 and supported, among others, by Mark Zuckerberg and Bill Gates) involving millions of students of different ages starting with 4-year old, aiming at introducing coding skills to a broader and mixed audience.

Programming is no longer just a job skill, but turned into a *literacy*, enabling people to acquire a new way of thinking and looking at the world, fostering the so-called Computational Thinking (CT) skills, i.e. all those thinking abilities reflecting fundamental principles and concepts of CS like problem-solving, abstraction, and pattern recognition to name but a few. These skills empower people to break complex problems down into small chunks and express them to a computer [Vee13]. CT shares many of its concepts, practices, and perspectives with other subject areas taught in schools, such as science, mathematics, arts, and engineering, making a strong case for its promotion in disciplines outside of CS and right from kindergarten [Nam+15] as a new form of literacy [Vee13].

Stephen Wolfram — the founder of Wolfram Research and creator of Mathematica and the Wolfram Alpha answer engine — wrote a blog post [@Wol16] that went viral arguing how CT is going to be a defining skill for our future and how important is to teach it to kids today. He notices that the future of any profession will be full of CT: medicine, law, education, farming, ..., whether it's sensor-based medicine, computational contracts, education analytics or computational agriculture, the future of any field "X" is going to

rely on being able to integrate and exploit CT properly. There either is now a "computational X" or there soon will be.

Indeed, CT is also influencing research in nearly all disciplines and enabling researchers to ask new kinds of questions and to accept new kinds of answers [Bun07]; it can, ultimately, change the way we think about the reality we live in, and its integration in the educational system is of pivotal importance for the future of the generations to come.

## 1.2 Research Questions

Due to the just discussed global interest in promoting CT skills to many broad and diverse audiences, several tools and methods have been designed with the aim of supporting the introduction of programming concepts in more effective and less daunting ways than in the past.

Currently, K-12 teachers around the world running introductory CT sessions are supported by a wide variety of multi-purpose technological tools mostly designed to target their usual scenarios and needs. The majority of them are digital tools using a Visual Programming Language (VPL) (e.g., Scratch [Res+09]) that allows users to program simple tasks by manipulating graphical elements on the screen. Many studies have been carried out investigating these tools in terms of their effects on programming ability or attitude, though not much discussion has arisen about their effects on developing CT in real-world educational scenarios, for instance when learners are working collaboratively in groups.

A popular theory of learning that can come to the aid on finding better ways to support CT skills development is Piaget's constructivism [PI69], which argues that people produce knowledge and form new meanings based upon their experiences and social interactions: exploiting it to teach those concepts underpinning CT might represent an effective and engaging way of supporting the learning of such skills.

Tangible User Interfaces (TUIs) [IU97] are an interaction paradigm that was devised to foster collaborative learning and exploit humans' natural dexterity for physical objects manipulation to provide an easy to use interface that can be used even by inexperienced people. TUIs exploit the physical world to offer a concrete representation of the abstract concepts learners usually struggle with, and thus could be used to foster CT skills [McN04; Hor+09].

Supporting users in cultivating their CT skills and — more generally — going through their routine learning experiences is particularly relevant in Informal Learning (IL) scenarios, namely environments where learning is predominantly unstructured, experiential, and noninstitutional, i.e. outside of the classroom

(e.g., in museums or workplaces). Modern education strives to make learning intrinsically driven, that is by making learners responsible for their own academic explorations, thus fostering appropriation of their own learning; this way their experience becomes more self-directed and personalised, increasing both their motivation and its efficacy. Developing both technological tools and methods to promote CT skills in IL domains puts learners in charge and integrates learning in their daily routines to exploit their motivations and provide a more effective experience. Physical objects manipulation might help to lower the barriers of CT and support users in dealing with such abstract concepts during IL activities.

Moreover, enhancing support for cultivating users' CT skills — and more generally their usual learning experiences — can be optimal when tools and activities are able to keep them in the so-called "Flow state": according to Csikszentmihalyi's theory [NC14], it refers to a state of intense concentration, sustained interest, and enjoyment of the activity's challenge, i.e. when skill and challenge levels of a task are at their highest, allowing users to learn at intense focus. It is hard to obtain such balance, since too much challenge causes anxiety, whereas too little challenge leads to boredom; one of the most common and explored ways of keeping learners in such state is through gameplay, that is by providing them with an engaging challenge and real-time feedback in response to their choices.

Coupling such activities that keep learners in the Flow state with physical interaction might enhance even further learning of CT skills by leveraging on a sustained engagement level, afforded social interactions, and a concrete representation of the abstract concepts underpinning it.

To recap, from this context discerns the main Research Question addressed by this thesis: *"Can the collaborative and cognitive naturalness of physical objects manipulation at the basis of Tangible User Interfaces aid the understanding of core algorithmic principles and thus improve end-users' Computational Thinking skills?"*.

Key Research Questions were formulated in order to support and investigate the main Research Question in detail:

- Do existing VPL-based tools support the collaborative learning of CT skills?
- Can physical objects manipulation help foster Computational Thinking skills in Informal Learning domains?
- Can physical objects manipulation provide a playful and engaging way of learning CT skills through gameplay?

## 1.3 Research Aims and Objectives

The research described in this thesis aims at *investigating the effects of TUIs on the development of CT skills*. In order to investigate this issue and address the aforementioned Research Questions, the following objectives were formulated:

- Identifying features of existing VPL-based tools that are suitable to cultivate CT skills in real-world educational scenarios.
- Designing and developing new tools and methods to support CT skills in different IL domains exploiting physical manipulation.
- Evaluate such tools to investigate which of their features support CT skills.
- Designing a suitable gameplay that can be integrated into different educational domains to support learners in developing CT skills.

## 1.4 Research Methodology

The research carried out in this thesis followed a three-stage process — exploration, development, and validation. Each phase corresponded to a major study carried out and reported in a related chapter.

The exploration phase's main goal was to investigate the current tools used in introductory programming sessions and explore their ability to support CT skills in real-world educational settings. The study quantitatively analysed real artefacts produced by participants to find effects of such tools on the development of CT skills in students working collaboratively and individually with the aim of identifying possible limitations of existing tools.

The development phase focused on a specific educational setting, namely IL scenarios, where people learn in a more self-directed way as they go about their daily activities, driven by their preferences and intentions. Two qualitative studies were carried out to design and validate a TUI-based system with the aim of supporting the learning of CT skills in multiple IL domains.

Finally, the validation phase validated the developed tool in an educational environment with young girls, with the aim of devising a suitable and engaging gameplay to foster CT skills in a wide range of scenarios. The qualitative study analysed multiple group sessions to identify features that exploit collaboration and increase engagement to benefit CT.

## 1.5 Thesis Structure

This thesis consists of six chapters, outlined in the following; in addition to the main literature survey in Chapter 2, a specific review related to each Key Research Question is reported in the chapter addressing it.

*Chapter 1* introduces CT, motivates the research, and outlines its objectives, methodology, and overall structure.

*Chapter 2* surveys the relevant literature related to Computational Thinking and Tangible User Interfaces.

*Chapter 3* presents an overview of current tools used to foster CT in educational contexts and evaluates their efficacy with respect to collaborative learning.

*Chapter 4* carries on with the main thesis investigation over the effects of Tangible User Interfaces on the development of CT skills and focuses on Informal Learning environments, where learning is mostly self-directed and takes place as people go about their daily activities, driven by their preferences and intentions.

*Chapter 5* deals with combining gameplay with TUIs to support the development of CT skills in yet another IL domain.

*Chapter 6* concludes this thesis by summarising the key research questions investigated, its contributions and implications, and presents future research directions.

# Background

# 2

> *If I have seen further, it is by standing on the shoulders of giants.*
>
> — **Isaac Newton**
> in a letter to Robert Hooke, 1675

This chapter presents a survey of the relevant literature pertaining to the work carried out in this thesis. The research surrounding Computational Thinking (CT) is introduced, covering how the concept evolved and differentiated itself from Computational Literacy (CL), the proposed definitions and ways it has been measured and supported so far. Furthermore, an introduction to Tangible User Interfaces (TUIs) is provided, suggesting how they can be used to aid learning and developing skills associated to CT.

## 2.1 Computational Thinking

As discussed in the introduction, since the coming of the Information Age, technology has progressively taken on a more prominent role in our day-to-day life; from the simple task of turning on the engine of your car to the vastly more complicated process behind the management of a flight, it is unmistakably clear how much our entire society depends on software: technology surrounds every aspect of our lives, and relying on it so much can be daunting at times. Most people weren't born in a high-tech world such as the one we live in today, but rather saw it developing overtime during the course of their lives. They got generally familiar with digital systems through their adult life, rather than growing up in the digital age as the so-called Digital Natives.

Being computationally literate and knowing the way into technology — and thus into our society — are becoming much-needed skills to possess for an ever wider and heterogeneous audience. This explosion of interest is witnessed by the many public initiatives that have been quite successful in the past few years in promoting such skills, and even more are coming along following the same path: Code.org's Hour of Code [@18e] is the main example of a successful global initiative involving millions of students of different ages starting with 4-years old and aiming at introducing coding to a wide audience with different backgrounds. Many known figures have stressed the importance of possessing these skills: former President Obama pledged to provide $4 billion in funding for

Computer Science (CS) education in U.S. schools as part of the CS for All initiative announced in 2016.

Many other initiatives and movements [Lee+14; Yad+14; Voo+15] are advocating the need of promoting coding and computational methods right from kindergarten and in disciplines outside CS itself as a new form of literacy.

### 2.1.1 Computational Literacy

One of the pioneers of CT is Seymour Papert, co-director of MIT Artificial Intelligence Laboratory from 1967 to 1981. He first mentioned CT in his seminal 1980 book Mindstorms [Pap80], where he discussed two aspects of computation, namely how it creates new knowledge, and how computers could be used to enhance thinking and change knowledge access patterns.

He connected CT and digital pedagogy to a modern approach to education known as constructivism, a learning theory initiated by Jean Piaget [PI69]. Piaget was a developmental psychologist who often collaborated with Papert back in the 80s; in brief, he stated that learners construct new knowledge in their minds from the interactions of their experiences with previous knowledge. Papert developed his theory of constructionist learning on top of Piaget's constructivism by adding the notion that learning is enhanced when learners are engaged in "constructing a meaningful product".

In the introduction to Mindstorms, Papert refers to the widespread of personal computers, and how people imagined them permeating home life and businesses. But he was thinking beyond those roles, to "how computers may affect the way people think and learn":

> A few talked about the computer as a teaching machine. This book too poses the question of what will be done with personal computers, but in a very different way. I shall be talking about how computers may affect the way people think and learn. I begin to characterize my perspective by noting a distinction between two ways computers might enhance thinking and change patterns of access to knowledge.

In the 1960s Papert, together with Bobrow, Feurzeig, and Solomon, presented LOGO [CGS99], one of the first high-level computer programming languages designed to embody constructionist principles. He continued to design and implement LOGO, creating what he deemed a social-constructionist sandbox. He believed that children are "active builders of their own intellectual structures", namely they could learn, apply, and come to know concepts and things through the process of writing programs: the "child as epistemologist", as he stated.

Accordingly, he claimed that such material and symbolic activities fostered both CL and conceptual skills, as the two are inseparable.

DiSessa [DiS01] followed up on Papert's ideas on CL, and better characterized CL in comparison with the traditional meaning of *literacy*, namely being able to read and write:

> Computers can be the technical foundation of a new and dramatically enhanced literacy, which will act in many ways like current literacy and which will have penetration and depth of influence comparable to what we have already experienced in coming to achieve a mass, text-based literacy.

He argues that literacy is built on three foundational pillars: first, the *material* pillar, involves the external, materially based signs, symbols, depictions, or representations that are technologically based and designed (e.g., alphabet, grammar, and syntax for written language, numbers and symbols for arithmetic). The second pillar is *mental or cognitive* and works in conjunction with the material basis of literacy to represent what we think and do with our minds in presence of materially based representations. The third pillar is *social*, giving literacy a value within the community.

He focused on the material affordances of programming, claiming that it can turn into a literacy if it becomes infrastructural to everyday life; furthermore, he adds that its ease of use can be the key factor in whether it will indeed become infrastructural, using the example of Leibniz's more intuitive notation for calculus in comparison with Newton's.

CT has a lot in common with DiSessa's definition of CL, even though in a recent paper [DiS18] he highlighted some key differences, indeed suggesting that the two movements should come together and join their current insights and future directions.

### 2.1.2 Defining Computational Thinking

After being mentioned a couple of times in Papert's work, CT has been brought into the limelight more recently by Jeannette Wing in her 2006 seminal work [Win06]. She reintroduced it as a mental skill set needed to solve complex problems like a Computer Scientist, but also widely applicable in today's information society.

At first, she did not mean to give a clear definition of CT through a brief, summarizing sentence; on the contrary, she described its characteristics and how they can be exploited in everyday life. She argued that it is mostly about problem-solving, and enumerated a list of features typical of

CT, the most important being *abstraction*, or the ability to think at different levels of abstraction: she stated that CS is not programming, and thus that conceptualizing is key. She also referred to abstraction as a fundamental skill that has to be learnt by anyone to function in modern society, as opposed to a rote skill that is mechanical and repeated over and over. Indeed, in her vision Wing wished that CT was part of the core teaching from the very young age, in the same way as reading, writing, and arithmetic are.

Four years later, Wing herself published another article [Win10] in which she — together with Jan Cuny of the National Science Foundation and Larry Snyder of the University of Washington — provided a more precise and short definition:

> CT is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.

CT is then both a thought process and a skill set. More precisely, it is regarded as the thought process involved in formulating problems and their solutions so that the "solutions are represented in a form that can be effectively carried out by an information-processing agent" [Win10]. Thus, Wing argues that CT is mostly about problem-solving, as well as the capability of using abstraction, problem decomposition, and algorithmic thinking [Win10; Voo+15; Mor12].

It covers far more than programming itself, including a range of mental tools reflecting fundamental principles and concepts of CS, such as abstracting and decomposing a problem, identifying recurring patterns, and being able to generalize solutions. However, as also suggested by Wing's seminal work, CT does not coincide with programming; rather, it "includes a way of thinking about everyday activities and problems" [SSA17]. Such problems are often ill-structured (or wicked), in that they may have neither definitive formulation nor boundaries [Fis17], thus the ability to analyze and solve them is very useful in everyday life. In accordance with this view, Lu and Fletcher [LF09] proposed to teach CT by using languages based on notions that are familiar to people, rather than using programming languages; this way, concepts like abstraction and algorithmic thinking could be more easily brought about.

As highlighted in the recent review by Shute et al. [SSA17], there is little agreement on the exact definition of CT and over the years several different definitions have been proposed.

Most of the literature works attempt to define CT skills as a set of facets comprising abstraction, algorithmic thinking, problem decomposition, and debugging. Shute et al. [SSA17] add iteration and generalization as two more skills that are important in CT.

*Abstraction* is the ability to think at different levels and the capability of modelling the core aspects of problems/systems by discarding irrelevant details [Win06]. Abstraction helps modelling problems and systems by capturing only the essential properties common to a set of objects; it allows hiding non-relevant differences, out of the scope chosen to evaluate the problem context. Building systems in terms of layers of abstraction allows developers to focus on one layer at a time and only on its formal relations between its adjacent layers; when moving to a higher-level, there is no need to worry about the details of the underlying specifications, thus providing us with an easy and effective way of focusing on one issue at a time without forgetting the overall process.

*Problem decomposition* is concerned with breaking a problem down into manageable units [Win06]. It is used when breaking problems, algorithms, artefacts, processes, and systems down to their functionalities, thinking about them in terms of their parts. Their components can then be understood, solved, developed, and evaluated separately, making complex problems easier to solve and large systems easier to design.

*Algorithmic thinking* refers to the ability to create procedures as ordered steps to implement solutions [AD16a].

*Debugging* is particularly emphasized in [Ber+14] as the ability to identify and fixing errors when algorithms do not provide the expected solution.

According to [SSA17], *iteration* is important for repeating design processes to refine solutions and *generalization* is fundamental to transfer CT skills to a wide variety of contexts.

In 2013 Selby and Woollard published an important work [SW13] which surveyed publications related to CT and the definitions extended from Wing's original one. They discussed the need for a precise definition, presenting support from the literature and the academic community. They found that the most common terms used to define CT are "a thought process", "abstraction", and "decomposition". Some other terms relate to different sets of thinking skills like logical, mathematical, and engineering thinking, problem-solving terms, and other CS terms. They propose a definition which states that CT is an activity, often product oriented, associated with, but not limited to, problem-solving. It is a cognitive or thought process that reflects:

- the ability to think in abstractions,
- the ability to think in terms of decomposition,
- the ability to think algorithmically,
- the ability to think in terms of evaluations, and

- the ability to think in generalisations.

Starting from Wing's definition, Repenning and his colleagues from the University of Colorado modelled CT as an iterative process structured in three stages [RBE16] (see Figure 2.1):

1. *problem formulation (abstraction)*, namely a verbal or diagrammatic conceptualization of the problem, in which abstraction plays a fundamental role to conceptualize a problem. Usually, a form of visual thinking can be helpful (e.g., diagrams);

2. *solution expression (automation)*, where the solution is described in a non-ambiguous way so that it can be executed by a computer. The most important tool in this stage is programming;

3. *execution and evaluation (analysis)*, where one may obtain visualizations of the outcome from the other two stages and evaluate them.

On the basis of the evaluation, problem formulation could be refined, and the cycle starts again. The idea they proposed is that the three stages of the CT process should be supported and integrated by means of CT tools, such as any kind of programming — including End-User Programming (EUP) — but also informal drawings, mind maps, and task-specific languages.

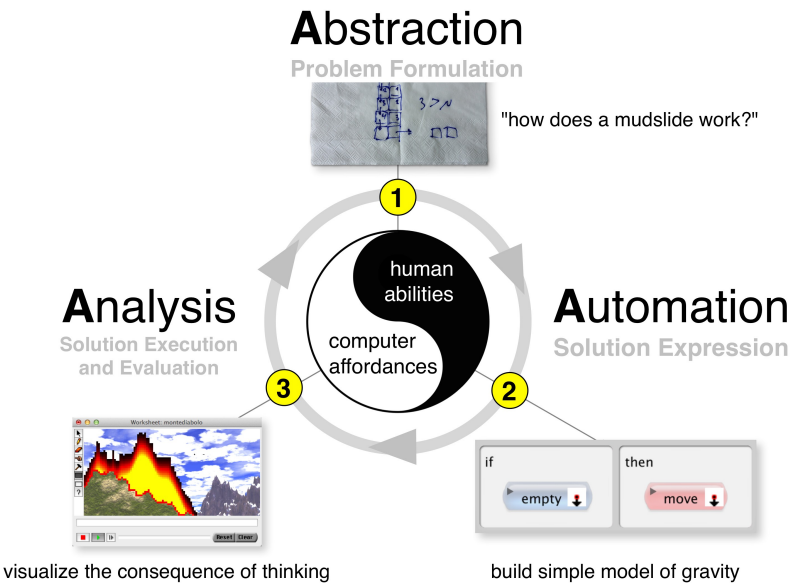



**Fig. 2.1:** The iterative CT process divided into three stages, shown through the example of a mudslide simulation [RBE16].

The whole process is iterative since the third stage often exposes the flaws of the previous two steps and it requires to start again from the

beginning. Describing the process in this way allows to point out the different responsibilities at each stage: execution and evaluation are entirely performed by a computer, while the phase of solution expression lies in the hands of the user. Repenning notes that contrarily to what one might think, the workload of the problem formulation stage can be shared between human and computer, the latter providing useful tools to support the conceptualization process.

Another popular framework attempting to define CT was proposed by Brennan and Resnick in 2012 [BR12]: their approach is based on a three-dimensional framework composed of computational Concepts, Practices, and Perspectives. Concepts refer to typical programming features and structures, such as sequences, loops, parallelism, events, conditionals, operators, and data. CT Practices are related to how people learn and put these concepts into practice. They identified four practices: (1) being incremental and iterative, (2) testing and debugging, (3) reusing and remixing, (4) and abstracting and modularizing. Finally, computational Perspectives try to describe how the human mind changes the way it views the world when experiencing CT. For instance, computing devices can be considered as something to be consumed, but for computer scientists they are also powerful tools for designing and expressing themselves. Social practices are affected as well because the opportunity of communicating and collaborating with people from all over the world brings huge benefits. It becomes possible to cooperate at a distance, but also to access and contribute to an unbelievable amount of shared knowledge. The last of these perspectives is the so-called questioning, described as not taking anything for granted and using design as a mean to overcome obstacles. It might be the case of an open software with a limited or missing feature; a possible reaction might be to fix it or add it to the program.

This definition evolved after years of work with Scratch [Res+09], a block-based Visual Programming Language (VPL) designed to introduce programming to people — children in particular — as well as allow them to develop always more complex projects. Therefore Brennan and Resnick's definition is directly linked to the tools designed to foster CT and provides a way to analyse them at different but complementary levels.

Another viewpoint of CT which contains a bit of both the definitions above, is the one by Kazimoglu et al. [Kaz+12a]. They identified five core skills as a result of an analysis of existing studies, before developing a computer game, Program Your Robot, which has the aim of teaching CT through programming, though at a higher level than Scratch. The core skills are problem-solving, building algorithms, debugging, simulation, and socializing. It is interesting that this set strives to summarize and blend Wing's and Brennan and Resnick's approaches,

thus it pulls the original CT definition more towards programming concepts, and the latter to a higher level.

Finally, a systematic literature review published in 2016 by Kalelioglu et al. [KGK16] analysed the many existing definitions of CT and argued that, even though many common characteristics can be found across different papers trying to define CT, the research is still at an early stage of maturity, and not many present solid theoretical or conceptual backgrounds. They propose a five stages framework based on the surveyed papers that combines both the scope of CT and problem-solving, but even this definition is not yet finalised and still in the development phase.

However, in the past few years, some critiques around the concept of CT have also been published: in their work [TD16], Tedre and Denning critically analyse the research surrounding CT and put it in perspective with the previous research from the 80s of computing in education. They hold a critical view of the CT buzz, arguing that much research has already been carried out in the past two decades under different definitions, and CT risks of failing in the same ways the previous research wave has. The risks for the CT research community are to fall into the trap of reinventing the wheel and water down 80s initiatives, considering CT as the best way of thinking in all environments, which is an oversimplification.

Lorena A. Barba [@Bar16] wrote a blog post in 2016 that generated a lot of discussion within the CT research community, entitled "Computational Thinking: I do not think it means what you think it means". She rejected the idea that CT means thinking like a Computer Scientist and is not about programming, arguing that current definitions are moving away from the initial ideas of Papert, by putting emphasis on problem-solving rather than projects, understanding rather than doing, content priming over media, and operations over their objects representations. She also argued that abstraction is not unique to CS and many other ideas of CS are not involved by current definitions. She suggested that by applying more closely original Papert's ideas like relevant projects, socializing, and investing on the social and cultural contexts might increase the fun of CS courses and making them less scary.

As remarked by the many studies reported, more research is definitely needed on developing a working definition of CT and the skills it encompasses; for the scope of this thesis though, Brennan and Resnick's proposed framework [BR12] represents the most complete and directly observable definition to be exploited when designing a new CT tool, as reported in the next section.

### 2.1.3 Measuring Computational Thinking

As a consequence of being a recent topic, the assessment of CT skills is still in a very early and experimental stage. There are some intrinsic problems that must be still overcome. First of all, the lack of a unique definition of CT skills implies that the choice of the method for the evaluation strictly depends on which skills are considered as pivotal, thus the assessment framework directly depends on the definition employed. Moreover, it is difficult to identify an evidence-centred way of assessing abilities like problem-solving or abstraction.

A very interesting mapping study has been carried out in 2016 by de Araujo et al. [AAG16] in which they gathered data from 27 studies performed in educational environments published between 2009 and 2016. They questioned the approaches used in promoting CT, which skills were assessed, and what instruments or artefacts were used.

The most widespread approach among these studies is represented by workshops, modules, and regular classes (13 out of 27), followed by exams without any classes (6), and frameworks specifically designed for assessing CT skills (5). Only three studies involved games or online interactive platforms. For what concerns the CT skills evaluated, problem-solving, algorithm construction, and abstraction are the only ones that can be considered recurrent, being involved in respectively 26, 20, and 9 studies; other abilities are not mentioned by more than 4 studies, witnessing the high uncertainty in establishing a common definition for CT.

The most popular tool employed to assess CT skills is multiple-choice questionnaire, found in 11 studies. Other very spread instruments are code evaluation (10 studies), and responses (9), while surveys, interviews, games, videos, lesson plans, and design scenarios are very rarely used (9 times in total).

The most common way of assessing CT skill progression is through the analysis of the produced artefacts; this usually leads to a checklist-based evaluation, consisting of an automated analysis looking for the presence of constructs (e.g., if or while conditions), making it suitable for assessing the so-called CT Concepts, as defined by Brennan and Resnick [BR12]. Clearly, it gets quite difficult to measure how developed Practices and Perspectives are; this is why when evaluating the impact of a new tool over users' CT skills, they also introduced artefact-based interviews and design scenarios in their assessment approach, as summarised in table 2.1. The interviews try to explore how the users developed the discussed artefact, questioning also about their background, how and if they participate in the online community and what is their general opinion about a tool. However, as recognized by the authors, no single approach is sufficient to capture all nuances of CT, an a combination of approaches could be appropriate.

**Tab. 2.1:** Strengths and limitations of different assessment approaches as summarised by Brennan and Resnick with respect to their CT framework [BR12].

| | *Concepts* | *Practices* | *Perspectives* |
|---|---|---|---|
| *Project Analysis* | Commands correspond to conceptual encounters. | — | Possibly adding extra meta-data for offline analysis. |
| *Artefact-Based Interviews* | Nuances of conceptual understanding, but with limited set of projects. | Based on own authentic design experiences, but limited by memory. | Maybe, but hard to ask directly. |
| *Design Scenarios* | Nuances and range of conceptual understanding, but externally selected projects. | Real-time and in novel situation, but externally selected projects. | Maybe, but hard to ask directly. |

If the considered CT skills belong to a higher level than programming, it becomes almost impossible to rely on quantitative data; for this reason, assessors usually recur to interviews and surveys. For instance, in order to evaluate the Program Your Robot game [Kaz+12b], the authors asked students to give feedback about the gameplay and their experience, highlighting when a CT skill had been stimulated in a participant.

Atmatzidou and Demetriadis [AD16a], authors of the educational robotics course mentioned in the previous section, asked participants to answer two intermediate questionnaires, in which they tried to investigate how CT skills were evolving in the activity of programming robots. Students were also prompted to give more personal views about the understanding of CT concepts in a questionnaire after the completion of the training. Besides, a think-aloud protocol was applied when students were asked to solve a programming task, followed by an interview in which opinions on the whole course were collected.

A similar approach was followed by the authors of CTArcade [Lee+14] when trying to compare how the paper and the software version of tic-tac-toe affected algorithmic thinking, pattern decomposition, pattern recognition, and abstraction. Interviews and think-aloud protocol were used, as well as a deep analysis of the users gameplay guided by a codebook for retrieving instances of the considered CT skills.

More generally, as de Araujo et al. [AAG16] pointed out, different interpretations of CT skills and concepts deeply impact their assessment, leading to different approaches, metrics, and dimensions used in experiments.

Therefore, in order to properly address the Research Question posed by this thesis, a combination of different approaches will be used — as suggested by Brennan and Resnick [BR12] — in order to try to capture most of the CT different dimensions summarised in table 2.1.

### 2.1.4 Fostering Computational Thinking

Even though CT was widely discussed for the first time only a few years ago, there have been attempts to foster it starting from the 1960s. Programming has proven to be an excellent way of developing CT skills [Orr09], thus teaching how to program can also be considered as an attempt to embolden the development of CT skills. This is the reason why the very first effort to teach CT can be attributed to Seymour Papert and Wally Feurzeig, who designed and developed the LOGO programming language in 1967 [CGS99]. It was an adaptation of the functional programming language Lisp and the target users were specifically children, with the aim of teaching them the basic concepts of programming.

Afterwards, in the 1980s programming games appeared. The basic idea is to make programming more desirable and enticing adding the possibility of letting your program "compete" against others. RobotWar [@18k], published in 1981, uses a language similar to BASIC, which was praised for being easy to learn. Players develop a program that simulates the behaviour of a robot, which then competes against multiple opponents in an arena. A few years later, Crobots [@18c] was released; it was based on a reduced version of C, and programs were written by the users in order to seek out and destroy other robots.

Game design represents a more recent approach to teaching programming and CT skills. In 1996 AgentSheets [Rep00], designed by Repenning, was released, even though the first prototype dates back to 1989. AgentSheets is still used in multiple contexts, from middle to high schools to academic environments, for various purposes such as introducing to programming, supporting storytelling, and prototyping simple games, just to name a few. The tool takes its name from the fact that the user develops the program on a grid resembling a spreadsheet, whose cells contain agents (see Figure 2.2). These entities, visualized as pictures, can perform multiple actions like reading Web pages and playing sounds and animations. Drag-and-drop interaction is employed to support users without any programming background.

The drag-and-drop paradigm is provided also by Alice [Her10], developed at the Carnegie Mellon University starting from 1997. Alice is actually an object-based programming language that allows creating animations, with an integrated development environment that allows users to forget the language syntax. Therefore, Alice is a valid tool for supporting storytelling while being

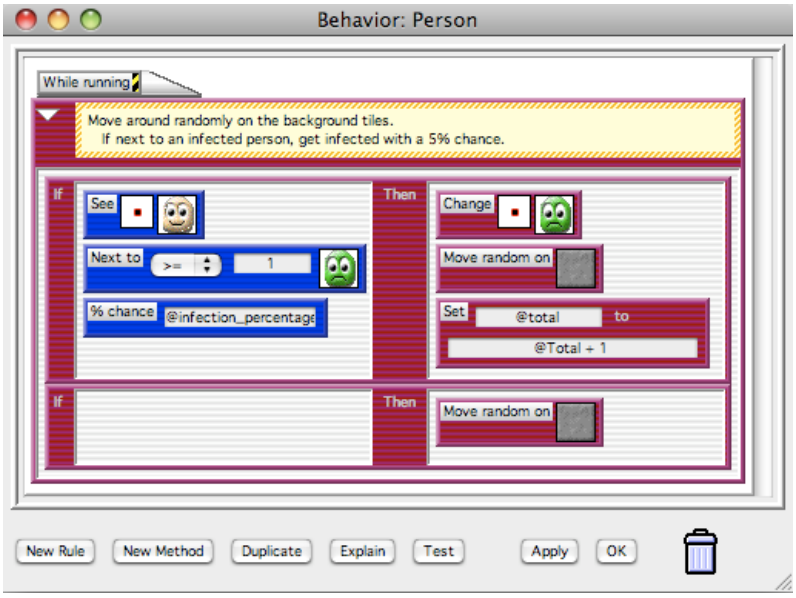



**Fig. 2.2:** Example of the definition of an agent in AgentSheets.

exposed to basic programming concepts without the burden of remembering syntactic constructs.

Perhaps the most influential and versatile tool for learning how to program is Scratch [Res+09], developed at the MIT and publicly released for the first time in 2005. It is a VPL whose interaction is made simple thanks to draggable instructions represented by blocks, fitting one another like puzzle pieces (as shown in Figure 2.3). The process of assembling instructions is guided by the different shapes and colours of the blocks, suggesting which constraints must be satisfied. One of its biggest strengths is the large and heterogeneous community of users that, combined with the possibility of reusing and remixing other users' code, allows to cooperate, share knowledge, and realize complex projects easily. The popularity of Scratch increased in the UK when Code Club [@18b] was founded in 2012, an initiative that aims to develop coding skills in children teaching the Scratch language itself, but also HTML, CSS, and Python.

Program Your Robot [Kaz+12b] is a recent game prototype developed by the research group led by Kazimoglu and colleagues, cited in the previous section. Based on the five core skills they identified as fundamental for CT, they developed a puzzle solving game in which the player has to assist a robot reaching a certain point on a grid. The robot will follow very simple instructions given in the form of an algorithm, while the score depends on conditions, for example, if two functions have been declared and then called by the algorithm. It differs from the software applications mentioned before, since they can be deemed programming languages at all effects, while Program Your Robot is conceived as a serious game. But most of all, tools like Scratch were designed in

order to teach the basics of programming and to show how fun it can be. Instead, Kazimoglu and his colleagues were moved by the goal of producing a game that could explicitly foster CT skills.

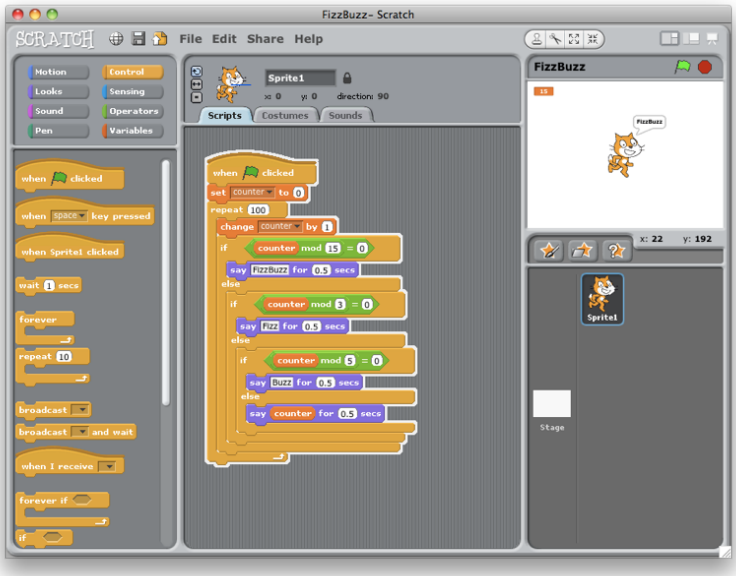



**Fig. 2.3:** An example of a program written with Scratch.

CTAarcade [Lee+14] is another serious game, designed with the target of boosting CT in players by letting them formalize their tacit knowledge and make a step towards abstraction. In CTAarcade users have to implement a set of rules that are observed by a character while playing Tic-Tac-Toe. Making these rules explicit is considered a very important process because they can often be applied in a natural, perhaps unconscious way and normally there is neither occasion nor reason to transform this knowledge into abstract instructions.

Another very interesting approach at fostering CT has been explored by Atmatzidou and Demetriadis in 2016 [AD16b], through seminars and sessions on educational robotics. Students aged 15 and 18 participated in a study in which they were exposed to notions that gradually became more and more complex, from basic programming concepts to managing sensors and variables, trying to put into practice what has been taught during the seminars. It emerged that programming robots actually helped to develop CT skills (abstraction, generalization, algorithm construction, modularization, and decomposition capabilities were assessed), regardless of age.

In the next section, a survey of TUIs literature is presented, together with an overview of their benefits on learning as support for the original claim of using them to foster CT skills.

## 2.2 Tangible User Interfaces

Declining hardware costs have recently enabled many new technologies to be available to a wider audience, together with new and engaging interaction modalities, particularly using gestures or object movements; this revolutionary paradigm goes under the name of the Natural User Interface (NUI), and it allows people to act and communicate with digital systems in ways to which they are naturally predisposed.

The term *natural* has been used in a rather loose fashion, meaning intuitive, easy to use or easy to learn; many studies argue that natural interaction can be designed either by mimicking aspects of the real world [Jac+08] or by drawing on our existing capabilities in the communicative or gesticulative areas [WW11b].

One of the most successful and developed approaches falling into the first category has been introduced by Ishii and Ullmer [IU97] and is known as TUIs (see Figure 2.4 for an example). The aim of TUIs is to give bits a directly accessible and manipulable interface by employing the real world, both as a medium and as a display for manipulation; indeed by connecting data with physical artefacts and surfaces bits can be made tangible.

Using physical tokens as interfaces to computer systems was pioneered by Fitzmaurice and Buxton [FB97] with Graspable User Interfaces: they build on the intuitive knowledge people have of everyday objects and take advantage of their rich physical affordances. They provide a tight match between real and digital objects through their similarity in terms of physical shape or types of manipulations that can be applied to them.

**Fig. 2.4:** Reactable, an electronic musical instrument with a tabletop Tangible User Interface (TUI) [Jor+07].

Many studies in this research area investigate the supposed benefits offered by this interaction paradigm, ranging from intuitiveness [IU97], experiential learning through direct manipulation [MOB09; PRI08], motor memory [Wei+09], accuracy [Mül+14], and collaboration [Sub+07]. Furthermore, the effects of employing a TUI to interact with a digital system are certainly dependent on the tasks and domain, as many comparative studies suggest [Wei+09; Mül+14; Han+09]; for this reason, Kirk et al. [Kir+09] made the case for hybrid surfaces, employing physical elements together with digital ones.

Researchers are also debating how employing TUIs reflects on learning [Hor+09; Mar07; AW13], with specific reference to highly abstract concepts: this stems from Piagetian theories [PI69] supporting the development of thinking — particularly in young children — through manipulation of concrete physical objects. Other studies [WWL14; HCB12] are even linking this effect to the development of CT skills [Win06], namely a new kind of analytical thinking integral to solving complex problems using core computer scientists' tools, such as abstraction and decomposition.

## 2.3 Contributions

The Literature Review described in section 2.2 has been previously published in [MT15; TMD15; TMD17].

## 2.4 Conclusion

In this chapter, a literature review on Computational Thinking research has been presented, starting from how the concept evolved from Computational Literacy, its many proposed definitions and ways it has been evaluated so far. Furthermore, an introduction to TUIs has been presented, suggesting the many ways they can aid learning and developing skills associated with Computational Thinking.

The next chapter starts the investigation on the effects of tangible interaction on the development of CT skills and focuses on existing tools used in education and their effects on promoting them.

# 3 Fostering Computational Thinking Skills with Visual Programming Languages

> *Education means making creators... You have to make inventors, innovators, not conformists.*
>
> **— Jean Piaget**
> Conversations with Jean Piaget, 1980

The previous chapter presented an overview of the existing research related to Computational Thinking (CT) and Tangible User Interfaces (TUIs). This chapter carries on with the primary thesis investigation on the effects of tangible interaction on the development of CT skills and focuses on existing tools used in education and their effects on promoting such skills.

## 3.1 Introduction

As discussed in the introduction, today's society is increasingly surrounded by technology, and it is unmistakably clear how much people rely on software in every part of their lives. That being the case, people will always strive to play a more active role in their life, thus programming is becoming an essential skill to master for the general public, resulting in an overall heightened interest in coding. Take for instance the Hour of Code [@18e], a successful global initiative organised by Code.org (a non-profit organisation founded in 2013 and supported, among others, by Mark Zuckerberg and Bill Gates) involving millions of students of different ages starting with 4-year old, aiming at introducing coding skills to a broader and mixed audience.

Programming is no longer just a job skill but has turned into a literacy, enabling people to acquire a new way of thinking and looking at the world, fostering the so-called CT skills, i.e. abilities like problem-solving, abstraction, and pattern recognition to name but a few. They empower people to break complex problems down into small chunks and express them to a computer [Vee13]. CT shares many of its concepts, practices, and perspectives with other subject areas taught in schools, such as science, mathematics, arts, and engineering, making a strong case for its promotion in disciplines outside

of Computer Science (CS) and right from kindergarten [Nam+15] as a new form of literacy [Vee13].

Due to this sudden global interest in promoting CT skills to many broad and diverse audiences, several tools and methods have been designed with the aim of supporting the introduction of programming concepts in more effective and less daunting ways than the past. A popular theory of learning that can come to the aid on this matter is Piaget's constructivism [PI69], which argues that people produce knowledge and form new meanings based upon their experiences and social interactions. Exploiting them to teach those concepts underpinning CT might represent an effective and engaging way of supporting the learning of such skills.

TUIs [IU97] are an interaction paradigm that was devised to foster collaborative learning and exploit humans' natural dexterity for physical objects manipulation to provide an easy to use interface that can be used even by inexperienced people. They exploit the physical world to offer a concrete representation of the abstract concepts learners usually struggle with and thus could be a valid way of fostering CT skills [McN04; Hor+09].

To recap, from this context discerns the main Research Question addressed by this thesis: *"Do the collaborative and cognitive naturalness of physical objects manipulation at the basis of TUIs aid the understanding of core algorithmic principles and thus improve CT skills?"*.

In the past few years the echo of the discussion around the importance of teaching programming and CT in school [11] resounded in the introduction of coding as part of their national curriculum by many European nations (such as England, Finland, and Estonia) and many other countries (e.g., Russia, South Africa, New Zealand, and Australia [BE14]) either already have or plan to introduce CS as part of their K-12 curriculum [GP13].

Currently, many educational events and lectures are introducing CT to newcomers and learners all around the world. Teachers are supported by a wide variety of multi-purpose technological tools mostly designed to target their scenarios and needs. The majority of them are digital tools using a Visual Programming Language (VPL) (e.g., Scratch) that allows users to program simple tasks by manipulating graphical elements on the screen. Many studies have been carried out investigating these tools in terms of their effects on programming ability or attitude, though not much discussion has arisen about their effects on collaborative learning of CT in real-world educational scenarios.

Tools and methods applied within these educational domains tend to nurture collaboration amongst learners rather than individual activities to facilitate

peer discussion and feedback, exploiting learning through direct experiences as suggested by constructivist theories.

The Research Question derived from this context and addressed in this chapter is then: *"Can existing VPL-based tools support the collaborative learning of CT skills?"*.

This chapter presents an evaluation of an existing VPL-based technological tool — i.e. OzoBlockly [@18j] — and currently used in introductory programming sessions to teach CT skills concerning its support to collaborative learning of CT. The study took place in a real-world environment, analysing an actual introductory programming sessions as they were carried out in a real educational institution by practitioners, without any intervention of researchers. It established a snapshot of today's main tools supporting CT and how new ones might improve upon their shortcomings and exploit their strengths.

## 3.2 Related Works

VPLs are a particular breed of programming languages that allow users to program by manipulating graphical elements on the screen rather than textual as in traditional programming environments [Jos+14]. Recently they have been adopted in many educational scenarios, thanks to their ease of use and efficacy in lowering the entry barriers of professional programming systems.

VPLs fall into two broad categories, namely Block-based, which falls directly from the imperative paradigm on traditional programming languages, or Flow-based, which discerns from traditional functional programming languages [MD17]. Even though there have not been many empirical studies evaluating the benefits of one against the other, in recent years many new Block-based VPL have been developed and successfully employed in introducing programming to children and fostering their CT skills in different scenarios and events.

The main principle guiding the VPLs-based tools' development was the "low floor, high ceiling" approach — i.e. the tool enables any beginner to cross the threshold to create working programs easily (low floor), but they are also powerful enough to satisfy the needs of more advanced users (high ceiling). Other effective tool features for promoting CT skills are represented by (1) providing stepping stones with managed skills and challenges, to get them from the "floor" to the "ceiling" (scaffold); (2) enable transfer between different application contexts; (3) support equity; (4) be systemic and sustainable [RWI10].

VPLs-based tools like Scratch [Res+09], Alice [Her10], Kodu [FFM12], Blockly [TG15], and App Inventor [Gra+12], closely follow these 5 principles

to varying degrees: they are relatively easy to use and allow novices to focus on design and creating while avoiding the issues of the traditional murky and complicated programming syntax.

They engage in CT using a three-stage "Use-Modify-Create" pattern [Lee+11], scaffolding increasingly deep interaction to foster the development of CT skills; users start using existing artefacts and gain confidence through a series of iterative modifications and refinements, as well as fostering appropriation of what started as someone else's and became one's own.

These tools, however, need to support — and, in turn, be supported — by curricular activities such as educational robotics and game design, typically serving as a trigger for the iterative exploration of CT while motivating and engaging school children.

These activities should support what have been proposed to be the four pedagogical phases of learning to think computationally [Nam+15]:

1. unplugged (off-screen) activities, to inspire students and enhance subject knowledge, making abstract concepts both tangible and visible, and improving upon their problem-solving skills;

2. making activities that include playing or making things, encouraging students to cohesively combine multiple ideas [Wil14];

3. tinkering [Ber+14], support learning of CT concepts and exploring in a creative way, providing a rich context for developing and representing understanding through the experience and building process [Pap80];

4. remixing (or "hacking") involves critically looking at an existing code, as well as practising modifying it to suit new purposes; analyzing code, making connections and creating new applications from existing code requires sophisticated reasoning and problem-solving skills.

Lastly, it is worth pointing out that the effectiveness of existing VPL-based tools seems pretty unsettled with respect to the many facets of CT: for instance, a 2008 study [Mal+08] involving 80 urban youth aged 8–18 reported learning of several CT elements through the use of Scratch in an after-school setting; nonetheless, the tool does not provide a mean of encapsulating functionalities into procedures and functions, somehow failing to tap into the abstraction skills. There is undoubtedly a need for new tools that foster CT skills specifically targeted to K-12 education, following the principles just described and guided by the most recent research on how children approach problem-solving [Che+07; PRM01].

## 3.3 Evaluation

This section presents the goals and hypotheses and describes the experiments carried out to evaluate existing technological tools supporting CT, and is organised following the guidelines of the American Psychological Association [Woh+00].

### 3.3.1 Goals

The goal of the study is to compare collaboratively working learners with individual working ones for the purpose of evaluating CT skills development in the context of using a VPL-based technological tool, namely OzoBlockly, in a real-world introductory robot programming session. The purpose is to test whether VPL-based tools are suitable to cultivate CT skills in real-world collaborative learning scenarios.

### 3.3.2 Research Question

Nowadays programming is not just a job skill, but it is rather an ability that enables people to acquire a new way of thinking and to look at the world, fostering mental skills such as problem-solving, abstraction, and pattern recognition to name but a few, the so-called CT skills. They empower people to break complex problems down into small chunks and express them to a computer [Vee13].

Currently, technological tools used to introduce CT to newcomers in educational scenarios are mostly graphical tools using a VPL. Over the past few years, many advantages of such interaction paradigm have been studied and discussed in relation to programming ability and effects on the entry barriers, but its effects on collaborative learning of CT skills have been somewhat disregarded.

The Research Question derived from this context and addressed by this study is then: *"Can existing VPL-based tools support the collaborative learning of CT skills?"*.

### 3.3.3 Experiment Design

A standard between-subjects design with one factor and two treatments were used [Woh+00]. The treatments correspond to (i) programming sessions with learners working individually (control group), and (ii) programming sessions with learners working collaboratively in groups (experiment group).

### 3.3.4 Participants

The participants of the experiments were 88 Key Stage 3 (aged between 11 and 14) students coming from 5 classes of different schools in the London Borough of Hillingdon area, all with comparable socio-economic level. The control group was composed by 25 individually working students from 2 of the 5 different schools (15 from one, 10 from another), while the experimental group consisted of 26 groups of students coming from the remaining 3 (15, 30, 18 from each), clustered in 15 groups of 2 students and 11 of 3 because of space and resources constraints.

Some students had some minimal programming experience — mostly with Scratch — but the vast majority had never programmed before. No prerequisite knowledge was required to participate, and none of the participants had prior knowledge of the setup of the experiment.

### 3.3.5 Settings and Experiment Tasks

The context of the experiment was robot programming, an activity often chosen as an engaging introduction to CT in educational events and sessions, as discussed in chapter 2. The VPL-based tool selected for the evaluation was OzoBlockly [@18j], an online programming environment based on Google's Blockly [@18a], allowing users to develop small programs by manipulating graphical elements on the screen. It can be used to control the movements and behaviour of Ozobots, small wheeled robots capable of moving autonomously and emitting lights. The programming environment doesn't require knowledge of any specific programming language and provides several types of instructions such as navigation, light effects, timing, and loops, all grouped by function, as depicted in figure 3.1. Once a user is happy with the outcome, the program can be sent to the robot through an optical sensor and executed by pressing a button on the side of the robot.

The study took place in a laboratory inside the Department of Computer Science within the Brunel University London facilities.

Each group was provided with an Android tablet locked on the OzoBlockly Web page, a fully-charged Ozobot, and a set of small pins.

The experiment took place during an actual CT introductory session, and the tasks consisted of a series of challenges of increasing difficulty to be completed by programming the Ozobot. The first required participants to make the robot do a simple turn and hit a pin at the end of the track. The second and third ones extended the first by requiring always an extra turn to hit the pin, making the track longer and more challenging. Finally, three pins were lined up in front of

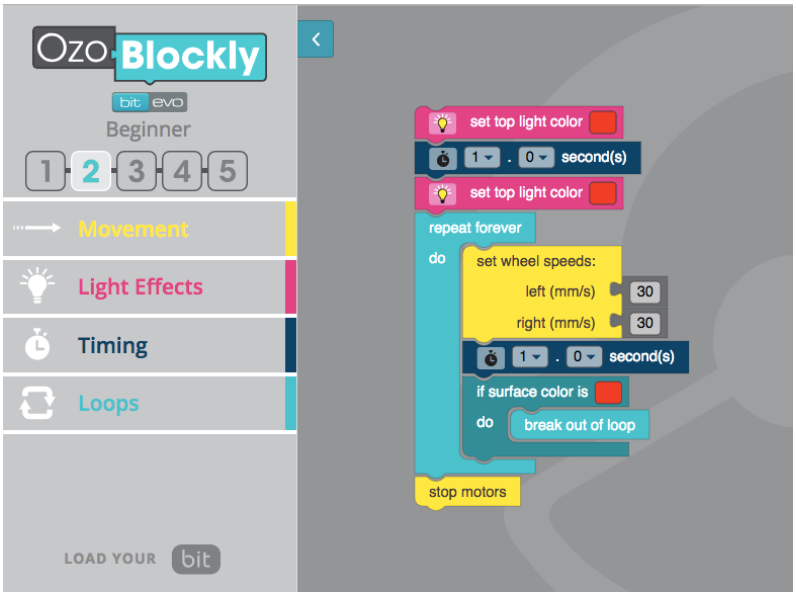



**Fig. 3.1:** The OzoBlockly programming environment.

the robot at a similar distance to one another, requiring participants to dodge each one of them in sequence to complete the race.

Tasks were designed to require multiple trials from learners to be as accurate as possible, fostering the reuse of previous solutions without being too challenging for inexperienced users.

Students were presented with each challenge and asked to try and solve it by themselves, with facilitators assisting students when needed. There was no automatic verification of the results, as the purpose of the educational session was not to evaluate their programming skills, but rather to introduce them to CT skills and engage them in controlling a robot. All the interactions and programs developed and sent to the robot were timed and recorded transparently, without interfering with the users' workflow.

### 3.3.6 Procedure

The experiment was carried out in one-hour long sessions, each involving a cohort coming from a single school, for a total of 5 different sessions.

Each session was chaired by one instructor, and no altering interventions were put in place by researchers, maintaining the actual schedule and tasks that were designed for the sessions. It started with a brief tutorial explaining the mechanics of the Ozobots, the OzoBlockly environment, and the process of sending programs to the robot.

Then participants were asked to try and develop a simple program making the robot move in a straight line, showing them how to do it first as an example.

After that, each challenge was introduced collaboratively and presented on a big screen by the facilitator. Participants were given 10 minutes to complete each task before introducing the next one, allowing them to try different strategies and discuss amongst teammates.

### 3.3.7 Experiment Variables and Formalized Hypotheses

The main independent variable of the experiment is group type and indicates whether a group was composed by more than one participant working together (denoted by "C") or by a single member working alone (denoted by "I").

As discussed in the previous chapter, measuring and defining exactly what CT encompasses is still under lots of discussions within the scientific community, but many proposed definitions mention programs sequentialization and loops as two fundamental and easily measurable concepts underpinning it. In order to keep at the minimum the intervention on a real-world scenario for the experiment, the study focused on actual tasks prompted in an existing educational session, and the measures were designed to be the least intrusive as possible in order to obtain more truthful data. Also, time-based measures were disregarded, since the point of such introductory sessions is not the evaluation of participants' programming capability, but rather to support them in practising their CT skills.

Thus, the main dependent variables selected for the experiment are the average length of sequential instruction issued together to the robot throughout an entire group session (shortened SEQ) — which could be related to the understanding of sequences, a core concept of CT — and the average number of loops used in each program sent to the robot (shortened CYC) — which could relate to the understanding of loops, another core CT concept.

The research question was formalised into the following hypotheses:

*Hypothesis* $H_0$ *(Null hypothesis):* There are no differences in CT skills development between learners working collaboratively and learners working individually when supported by a VPL-based tool.

*Hypothesis* $H_1$ *(Alternate hypothesis):* There is a difference in CT skills development between learners working collaboratively and learners working individually when supported by a VPL-based tool, thus employing such tools in collaborative learning scenarios affects the development of CT skills.

Table 3.1 presents the experiment hypotheses formally in terms of the dependent variables. In the table, "I" refers to learners working individually

(control group) and "G" refers to learners working collaboratively (experiment group).

**Tab. 3.1:** Formalized Hypotheses: SEQ is the average length of sequential instruction issued together to the robot throughout a group session, while CYC is the average number of loops used in each program sent to the robot within the same session; "I" refers to learners working individually (control group) and "G" refers to learners working collaboratively (experiment group).

| Null Hypothesis ($H_0$) | Alternate Hypothesis ($H_1$) |
|---|---|
| $SEQ_I = SEQ_G$ | $SEQ_I \neq SEQ_G$ |
| $CYC_I = CYC_G$ | $CYC_I \neq CYC_G$ |

### 3.3.8 Summary

To recap, table 3.2 presents a summary of the specific CT dimensions as defined in [BR12] examined in the evaluation, along with the related assessment approach, as reported in the previous chapter.

**Tab. 3.2:** Summary of the specific CT dimensions [BR12] considered by the evaluation with the related assessment approach.

| *CT Dimension* | *Description* | *Assessment* |
|---|---|---|
| Concept: sequences | Expressing a particular activity or task as a series of individual steps or instructions that can be executed by the computer. | Project Analysis (SEQ) |
| Concept: loops | Recognizing repetitions in a sequence of instructions and expressing it in a more succinct and abstract way. | Project Analysis (CYC) |

## 3.4 Results

Fifty-one data points were obtained from the experiment, 25 for the control group (I), and 26 for the experimental group (G). The total number of valid programs issued to the robots was 4,653: 2,423 from the control group, and 2,230 from the experimental group.

Figure 3.2 shows the boxplot for the number of programs issued to the robot by each experimental group. The average was 96.92 (standard deviation 33.48) for learners working individually, and 85.77 (standard deviation 38.05) for learners working collaboratively, thus both groups issued a quite similar number of

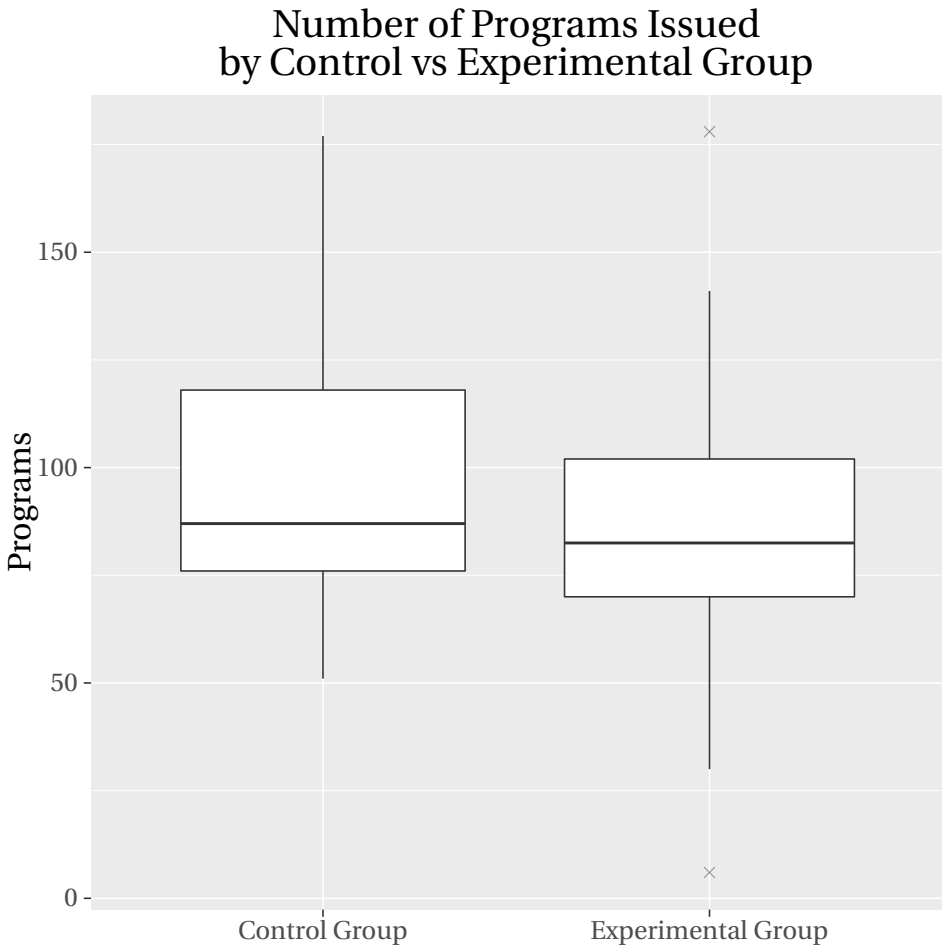



**Fig. 3.2:** Boxplot for the number of valid programs issued by each group.

programs. This was mostly expected, since tasks, setup, and available time was the same for all the participating groups. The slightly smaller average reported by the experimental group might be due to having a single control point (the tablet) and compromising between the perhaps different views of team members.

Moving on to measures related to the experimental hypotheses, the boxplot in figure 3.3 shows the average number of sequential instructions issued to the robot across the different experimental groups (SEQ). The average was 4.79 (standard deviation 1.35) for learners working individually, and 4.21 (standard deviation 1.95) for learners working collaboratively, thus it would seem that working individually prompts the use of slightly more sequential blocks when performing the same tasks.

Finally, the boxplot in figure 3.4 shows the average number of loops issued to the robot across the two experimental groups (CYC). The average was 0.39 (standard deviation 0.48) for learners working individually, and 0.46 (standard deviation 0.5) for learners working collaboratively, thus as with SEQ, the

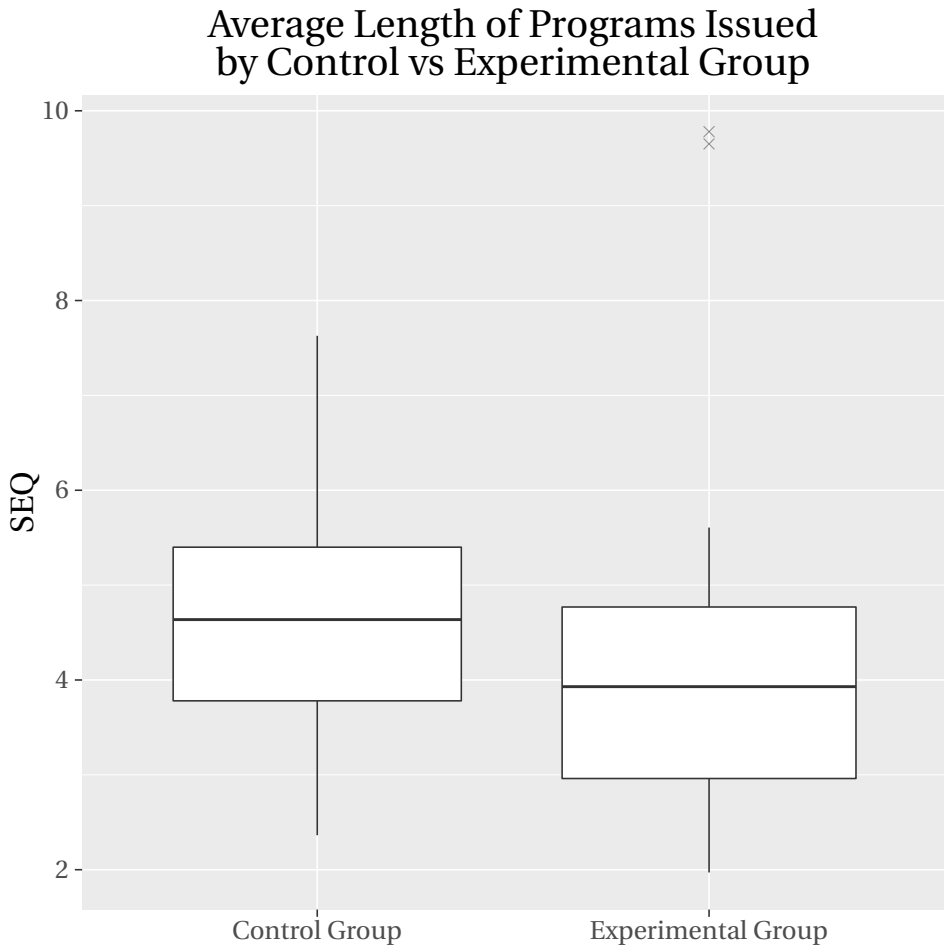



**Fig. 3.3:** Boxplot for the average length of valid programs issued by each group.

difference between the two experimental groups is narrow. In this case, though, it seems that working collaboratively fosters the use of loops more than working individually, favouring CT skills.

Both SEQ and CYC were not normally distributed according to the standard Shapiro-Wilk test ($p_{SEQ} = 0.001$ and $p_{CYC} < 0.001$). To verify the experimental hypothesis, the one-tailed Mann-Whitney U-test was selected, which is a robust, nonparametric test.

The experimental hypothesis ($H_1$) predicts a difference in CT skills development between learners working collaboratively and learners working individually. The tests were not statistically significant though ($p_{SEQ} = 0.0658$ and $p_{CYC} = 0.4776$), hence the null hypothesis cannot be rejected.

## 3.5 Discussion and Post-Hoc Analysis

Even if the results of the experiment did not confirm the experimental hypothesis, there are interesting pointers coming from the collected data that

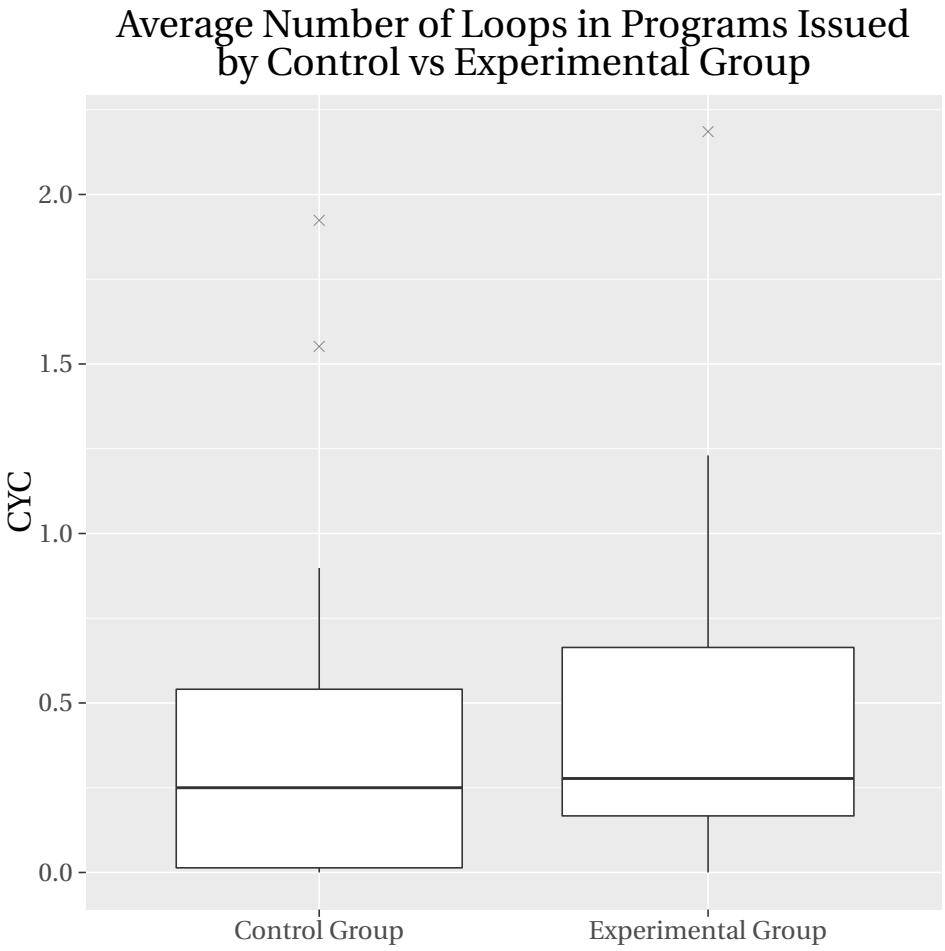



**Fig. 3.4:** Boxplot for the average number of loops issued by each group.

can be helpful to explore more in-depth the main research question of this thesis and offer valid suggestions for the follow-up studies.

Results of the two evaluated measures — SEQ and CYC — point towards opposite effects: even though the difference is narrow, it seems that individual working groups tend to produce longer sequential programs while collaborating groups use more loops. This yet not significant effect would have been expected: by leveraging on collaboration, VPLs-based tools should foster higher-level CT skills such as loops and abstraction, since social interactions are an important factor of constructivist learning theories, as discussed in Chapter 2. Unfortunately, the study failed to measure this sought effect (if present), and the reasons might be various and worth analysing in more detail.

First, VPLs are designed to run on traditional digital systems, which are based on a Graphical User Interface (GUI) interaction paradigm: they are based on artificial control devices such as mouse and keyboard, and they don't support and take advantage of collaborative situations when multiple users are collaborating with the support of the device. A recent evolution in interaction

design is indeed founded on such premise: the more modern Natural User Interfaces (NUIs) are based on more innate human interaction paradigms such as touch, vision and speech, and are known as *natural* because they rely on a user being able to carry out relatively natural motions to control the application or manipulate the on-screen content, fostering and cultivating user collaboration. Designing new and engaging tools founded on the basis of NUIs could represent a shift towards new CT support tools, which might take more advantage of collaborative and experiential learning to better cultivate such skills.

Moreover, VPL-based environments such as Scratch and Blockly have developed highly active online communities over the past few years, providing users with existing solutions to problems, forums to get help and discuss amongst peers, and many other resources. They were designed to take advantage of online collaboration rather than offline, representing a CT support tool more for individual working learners, rather than collaborating ones.

The tasks provided to participants were perhaps too simple to observe a significant effect between the two experimental groups. The selected tasks were indeed real-world introductory tasks, designed to get students interested in robot programming without challenging right away inexperienced ones. Perhaps the sought effect would have been more significant with different and specifically-designed activities, but it would not have been as much grounded in existing real-world practices.

Finally, the study did not take into account more common measures such as time to task, tasks completion rates, and many other subjective measures related to attitude, engagement, and reflection. Even though these are more common measures used in the literature, the study design strived to detect CT developments rather than programming ability. Perhaps further research on non-invasive measures of CT skills could help highlighting this effect without designing ad-hoc evaluation tasks that might not highlight real-world effects taking place.

## 3.6 Threats to Validity

There are several validity threats to the design of this study.

**Internal Validity** The limited task complexity and available time prevented a full evaluation of different CT factors that could have influenced by the different treatments. However, the study has been conducted in a real-world scenario, with the original tasks designed by the instructors to introduce CT to students and in their usual time schedule; even though it presents a limitation regarding

research power, it offered a valid assessment of current practices and real-world effects of the evaluated tools.

**Construct Validity** The CT assessment was reduced to just two factors, understanding of sequences and loops; these are just two concepts related to more higher level skills defined in CT, even though they are not explicitly mentioned by all the proposed definition outline in research. Nevertheless, these concepts were taken into account to avoid instrumenting the experiment and keep it as close as possible to real-world practices; indeed, sequences and loops comprehension can correlate with abstraction and decomposition abilities, which are two of the most mentioned CT skills found in the literature.

**External Validity** The results of the study can be generalized only in the context of the selected scenario, although it was carried out in a real educational setting, during an actual introductory programming session without instrumenting them. In order to generalize the findings to other scenarios though, replication studies employing different VPL-based tools are needed.

## 3.7 Contributions

The Literature Review described in section 3.2 has been previously published in [TM16b; TM16a].

## 3.8 Conclusion

This chapter presented an investigation on existing VPL-based tools used in education and their effects on promoting CT skills in a collaborative learning scenario.

A common VPL-based system — namely OzoBlockly — was tested in a series of real-world introductory programming sessions, where Key Stage 3 learners collaborate to program a small robot and solve some simple tasks. The evaluation compared the effects of using such a system while collaborating with other peers and working individually, in terms of the development of CT skills. The study results failed to demonstrate a clear benefit of learning in collaborating groups over individual working ones in terms of CT skills development. This requires further investigation, however it could point out how existing CT tools favour offline collaboration, but are not specifically designed to support online collaborative learning. Designing tools to support online collaboration to foster CT skills is indeed important and worth investigating whether different an interaction paradigm — such as TUIs — could be used to make this happen.

In the next chapter, the main research question of this thesis about the effects of tangible interaction on the development of CT skills will be investigated in a specific application domain, namely Informal Learning (IL) environments, where learning is self-directed and takes place as people go about their daily activities.

# 4 Fostering Computational Thinking Skills in Informal Learning Domains

> *Children learn best when they are actively engaged in constructing something that has a personal meaning to them — be it a poem, a robot, a sandcastle, or a computer program.*
>
> **— Seymour Papert**

The previous chapter focused on the existing tools used in education and their effects on promoting Computational Thinking (CT) skills. This chapter carries on with the main thesis investigation over the effects of tangible interaction on the development of CT skills and focuses on a specific application domain, namely Informal Learning (IL) environments, where learning is mostly self-directed and takes place as people go about their daily activities, driven by their preferences and intentions.

## 4.1 Introduction

As mentioned in chapter 1, supporting users in cultivating their CT skills and — more generally — going through their routine learning experiences is particularly relevant in IL scenarios, namely environments where learning is predominantly unstructured, experiential, and noninstitutional, i.e. outside of the classroom (e.g., in museums or workplaces). Modern education strives to make learning intrinsically driven, that is by making learners responsible for their own academic explorations, thus fostering appropriation of their own learning; this way their experience becomes more self-directed and personalised, increasing both their motivation and its efficacy. Developing both technological tools and methods to promote CT skills in IL domains puts learners in charge and integrates learning in their daily routines to exploit their motivations and provide a more effective experience. Physical objects manipulation might help to lower the barriers of CT and support users in dealing with such abstract concepts during IL activities.

The Research Question derived from this context and addressed by this chapter is then: *"Can physical objects manipulation help foster CT skills in IL domains?"*.

To properly study and address this question, this chapter introduces TAPAS (TAngible Programmable Augmented Surface), a system combining its digital and physical features to promote CT skills, built on the findings of the previous chapter, the literature, and a user study, that can be repurposed to different IL scenarios.

The contributions reported in this chapter are threefold.

- First, an early design of TAPAS is described, stemming from the results of a workshop carried out with expert designers to collect insightful ideas and design challenges related to the development of a Tangible User Interface (TUI)-based system that can be repurposed to different IL domains.
- Second, the results of a study conducted with the preliminary prototype of TAPAS [MT15; TMD15] are reported, whose findings were used to inform its design to target IL domains.
- Third, some of the main challenges faced by supporting CT skills with objects manipulation in IL domains are reported, based on insights from the studies that have been carried out.

## 4.2 Related Works

Much of the existing body of research within the End-User Development (EUD) area about TUIs has focused on education. Many theories, including the constructivist ones, report the benefits of making interfaces tangible and moving them into the physical world; several studies report on their benefits for children in the classroom since they are built on experience from the real world. Providing real evidence supporting their value for educational use is quite a challenging task, although some recent studies are trying to characterise them more in detail, as this thesis seeks to do with their effects on CT skills.

The existing literature on employing TUIs within an educational domain to foster programming-related skills can be clustered in two main categories according to the paradigm employed: Programming by Demonstration (PbD) or Programming by Instruction (PbI). PbD, also known as Programming by Example, enables users to teach new behaviours to the system by demonstrating actions on concrete examples [Lie00]. PbI, known as Tangible (sometimes Physical) Programming within the TUI domain, takes a traditional approach to programming, that is requiring users to learn and employ a syntactic construct

(e.g., text instructions, visual or natural language) to impart instructions to the system. This makes the PbI paradigm less coupled with the operational domain it is applied to, whilst PbD is directly linked to it by having users manipulate only their artefacts.

Topobo [PRI08] — proposed by Parkes et al. — falls under the first category and comprises a set of components that one can assemble and animate with different manipulations; the system then repeatedly plays those motions back. PbD proved to be an effective and intuitive way of teaching different movements to a system directly on actuated physical objects, therefore it has been specifically named Robot Programming by Demonstration [Bil+08]. The system devised by Lee et al. [Lee+13] uses a different approach: this PbD system allows users to record macros composed by physical and digital actions performed on several objects, such as opening a drawer, turning on the TV, and so on; the system records the actions' sequence and plays them back in the same order once the first action is performed.

These systems offer an unparalleled experience in terms of ease of use, but — due to the paradigm they employ — present a quite substantial limitation: users can interact only with the outputs, therefore the instructed behaviours are necessarily composed solely of operations that are directly available, resulting in the inability to represent more complex behaviours; this is the reason why the main problem of PbD systems is the generalizability — i.e. finding the general semantics — of instructed behaviours [Lie00].

Moving to PbI-based systems, Mugellini et al. [Mug+09] proposed the concept of tangible shortcuts: they improved information access and retrieval using physical objects, enabling users to develop new shortcuts through a Visual Language based on a puzzle metaphor. Wang et al. introduced E-Block [Wan+12], a tangible programming tool for young children, enabling them to instruct a robot's movements by assembling different blocks, each assigned to a specific function. Robo-Blocks is a similar system presented by Sipitakiat and Nusen [SN12], which added the ability for users to debug their applications using a display placed on top of each block.

The effects of employing a TUI to interact with a digital system are certainly dependent on the tasks and domain, as many comparative studies suggest [Wei+09; Mül+14; Han+09]; for this reason, Kirk et al. [Kir+09] made the case for hybrid surfaces, employing physical elements together with digital ones.

## 4.3 Tangible Programming in Informal Learning Domains

As discussed in the introduction, IL is a very effective form of learning that fosters appropriation by allowing users to self-pace their experience outside of the traditional classroom. Technological tools and methods can assist learners through their self-directed journey, particularly when the goal is to promote CT skills. Also, by allowing such tools to exploit humans' natural dexterity for objects manipulation as implied by constructivist theories, they might further support learning of the abstract concepts underpinning CT within such domains.

The design of technological tools enhancing CT skills within IL domains is influenced by some *key factors* concerning these scenarios. As reported in [MV99], IL is (F1) integrated with learners' daily routines, (F2) triggered by an internal or external jolt, (F3) not highly conscious, (F4) influenced by chance, (F5) an inductive process of reflection and action, and (F6) linked to the learning of others. Leveraging on these factors when designing a new tool allows for an optimal support of learning activities in such environments and limits its casual nature.

In this chapter, a new tool supporting IL of CT skills is proposed, which

- is a software platform — namely a system designed to run different applications — thus it can be repurposed to different scenarios, integrating into learners' heterogeneous daily routines (F1);
- supports inductive processes (F5) by providing users with an EUD environment where they can iteratively assemble different workflows — namely sequential processes combining different small applications in a step-by-step data-flow fashion, where the output of an application becomes the input of the following one.
- supports collaboration (F6) by implementing a TUI on a tabletop display — namely an interactive display system, naturally fostering users collaboration by providing multiple physical interaction points placed on a horizontal surface for them to play with.

In the next section, the process of designing such a platform is outlined, starting with a focus group with designers to devise a first preliminary design.

### 4.3.1 Preliminary Study

Several design choices have to be made to design a platform with tangible controls that need to be easily integrated into many different IL scenarios. For

this reason, an early focus group with experts was run to collect ideas and suggestions to drive the design choices.

Five experts with backgrounds in different design areas were gathered in a university meeting room for a one-hour session: three experienced interaction designers with some basic programming knowledge — one with a specific background on information visualization and one with quite substantial industry experience — and two product designers without any programming experience at all.

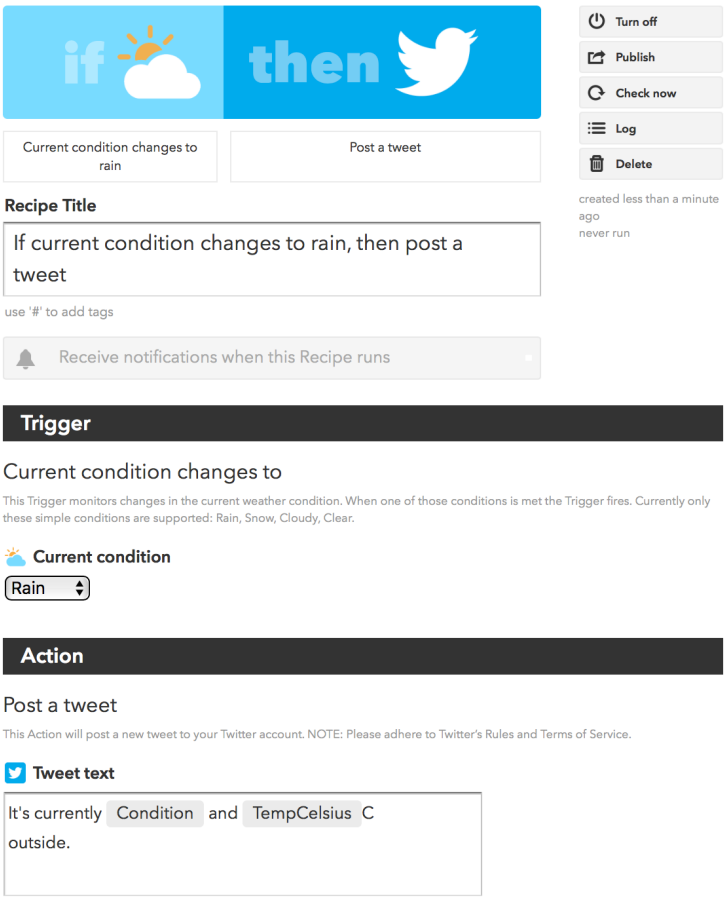
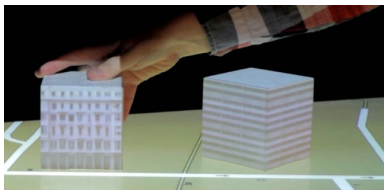



**Fig. 4.1:** An example of a workflow created using IF This Then That (IFTTT): when the condition in the user's location changes to rain (trigger) it will automatically post a tweet (action).

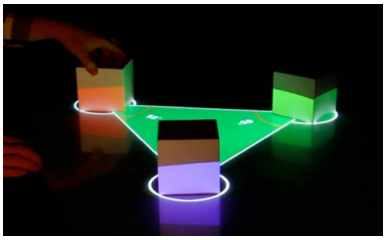
During the first phase — lasting 30 minutes — participants were instructed in the context of this research and the problem it is addressing by the facilitator. They were shown some videos outlining the process of developing workflows on IFTTT (IF This Then That) [@18f], a widely popular EUD Web mashup system [Mal+11]; it allows users to create simple event-based *if-then*-style workflows with different Web services and acts as a hub connecting their events' triggers with actions: one can describe simple rules by selecting the event

that will trigger the workflow (e.g., when the current temperature rises above a certain value or when the user edits a specific file on Dropbox) and an action that should be performed by any other — even the same — supported Web service (e.g., tweet about it or send the file via email), as shown in figure 4.1. This platform is very flexible and can be easily integrated into most people day-to-day activities on the Web to support them. Thus, it was chosen to showcase different types of simple workflows, their inner logic and how the trigger selection provides the subsequent action with anchors dependent on the output's type: for instance, when the event concerns a location the action can access its GPS coordinates, when it involves a text file the action will be able to use its content, and so on.

Then participants were shown a video of an existing TUI system — the Tangible 3D Tabletop [DH14] — which summarized the benefits of this interaction paradigm. In particular, two different ways of employing tangible objects in educational systems were shown [ZAR05], in order to prompt them to produce different ideas: Froebel-inspired Manipulatives (FiMs) are building blocks used to design and represent real-world things, objects, and physical structures, for example, 3D building blocks to represent buildings on a map (figure 4.2a). Montessori-inspired Manipulatives (MiMs) are building blocks focused on modelling more conceptual and abstract structures, for instance, tangibles used to control RGB colour blending (figure 4.2b).



**(a)** 3D building façades on tangibles placed on a map.



**(b)** Three tangibles used as RGB colour blenders. The base colour is projected onto each tangible, and users can change the hue of the triangle drawn within the tangibles by turning them.

**Fig. 4.2:** Two examples of different metaphors involved in the Tangible 3D Tabletop system [DH14].

After the introduction, participants started a 30-minute discussion about ideas and challenges for the design, focusing on an IL scenario involving users with no previous programming experience. The gathered feedback is summarised in the following.

### Study Findings

The features suggested by participants were clustered by subject — concerning either the system's tangible objects or its digital syntax. Here are the main findings from the focus group:

**Tangible features** Participants remarked the fact that the system should react only upon user actions and provide useful feedback through a specific communication channel, in agreement with one of the main principles of Natural User Interfaces (NUIs) [WW11a]. Many suggestions focused on the preferred channel to be used to provide feedback. These included providing tangible objects with a touch-sensitive mechanism in order to activate the feedback only when users physically touch objects on the table, in order to highlight whether selected objects are compatible with each other (fulfilling the workflow constraints). Moreover, the feedback communication channel of choice can be a physical one as well: a magnetic attraction between objects could indicate whether two workflow's components are compatible with each other, while repulsion might represent the opposite. Another participant suggested employing haptic feedback built into the tangibles to communicate compatibility between different ones.

**Digital features** Another set of suggestions were directed towards the digital representation of the platform's syntax. First, the blocks' digital representation should help users understand components' constraints by using, respectively, different and similar colours or shapes for incompatible and compatible components. Also, since a workflow's composition is usually performed one component at a time, i.e. by selecting a function that will follow the latest assembled one, the system shall aid users on the next available components to be chosen by changing the colour or the shape of the currently assembled workflow. Lastly, available components should be displayed all at once, giving users an overall view of the system's capabilities. However, this can also increase mistakes. Since the target group is inexperienced users, the system should assist them in finding the right way of assembling different components, when they cannot figure it out themselves; a useful suggestion on this regard is to provide some sort of "translation tool", which — once a user selects two blocks incompatible with each other — shows them at least one possible way of choosing other components in between to connect the two blocks, assisting users during the composition phase.

The suggestions stemming from the workshop drove a preliminary design of the system, whose details are presented in the next section.

### 4.3.2 Preliminary Design

The system design aims to provide users with a platform that can be employed in a variety of IL domains — e.g., in museums or workplaces — to cultivate CT skills through objects manipulation.

The platform is deployed on tabletop displays, which naturally foster collaboration as required in IL domains, and consists in a Block-based Programming environment [Moh+11] to support inductive processes — widely used in systems like Scratch [Res+09] and Blockly [@18a] — that has proven to have a low learning threshold for non-programmers, as thoroughly discussed in the previous chapter.

It allows users to create, share, modify and reuse simple workflows, namely sequential processes combining different services in a data-flow fashion, where the output of a service becomes the input of the following one, integrating into users' daily routine while supporting IL through inductive processes.

Users impart instructions through a visual syntactic construct in a PbI fashion rather than by demonstrating their intentions to the system: indeed, making a workflow's inner architecture transparent to users can help them to better understand its sequential logic and behaviour, providing further design opportunities to improve their CT skills and easily integrate the system into rather different scenarios.

The system's blocks correspond to workflow components (i.e. functions) that can be assembled together as in many other Block-based Programming environments; each block receives specific formats of data as input and produces different ones as output based on its inner workings and its location within a workflow's logic. Type constraints on different blocks inputs and outputs are afforded using different shapes, as in a puzzle.

A tangible object is associated with the main block — a circle halo with a single hollow to accommodate the next piece to be added to the workflow — which will move alongside the object on the main display's surface; moving the main piece towards another will add the latter's related function to the workflow — only if the two shapes are matching, that is to say, the latest output is compatible with the required input (figure 4.4a). This mechanism aids end-users in understanding the data-flow approach as well as type constraints.

Inputs requested by services might be quite heterogeneous and complex depending on the scenario, requiring at this point a precise and familiar input mechanism that can adapt to different needs. Hence, smartphones have been selected as the tangibles controlling the main blocks in the system: they represent objects whose movements allow users to interact with the system, i.e. they form the physical and digital representation of information in the system, and

are already equipped with all the sensors and feedback mechanisms needed to implement the designers' suggestions obtained from the focus group and push the interaction even further. They can be used to adapt the system to the different users' preferences since they hold much of their personal information. Moreover, they can be used to display a wide range of widgets that can be presented to end-users depending on the specific service being accessed (e.g., a virtual keyboard for text input).

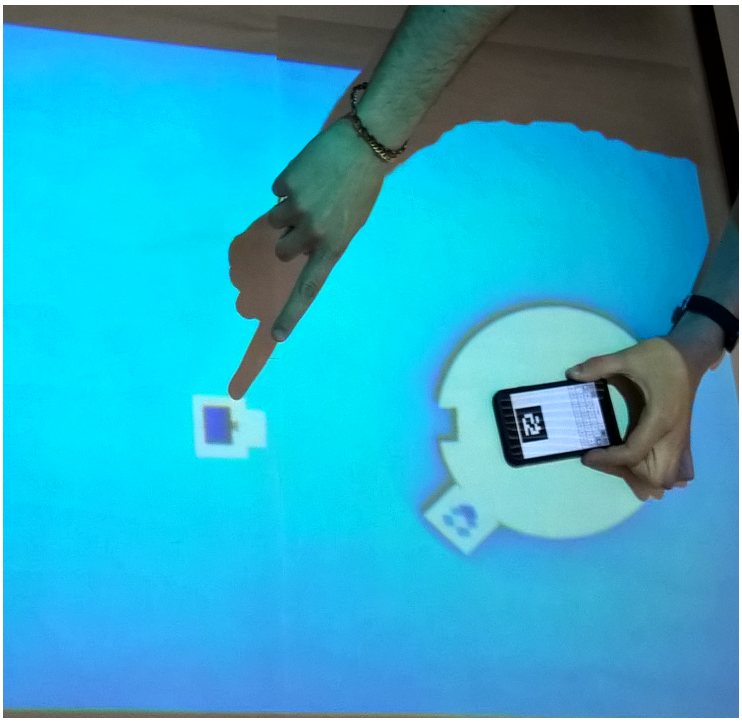


**Fig. 4.3:** An example of a workflow being assembled using the proposed system: a keyboard widget is displayed on the smartphone once a new piece requiring an input is assembled.

Widgets vary depending on the type of input requested: selecting a single option among several will prompt the user with a list box, a single action to be performed will display a button, and a generally unstructured raw text to be inserted will present a keyboard (figure 4.3 and 4.4b). Once a user enters the requested input on a widget, the latter disappears from the smartphone and the projected halo surrounding it opens up a new hollow to allow for the next block to be inserted (figure 4.4c); then using the input, only the hollow that is compatible with it is displayed, preventing invalid compositions. When a workflow is completed, it can be run by pressing a button on the smartphone (figure 4.4d).

The next section reports a study carried out to evaluate the platform and address the Research Question formulated in the introduction of this chapter

(a) The first piece is selected and added to the current workflow by moving the smartphone towards it.

(b) The corresponding widget is displayed on the smartphone waiting for user input.

(c) Once the input is inserted, a piece whose input matches the current workflow's output can be added.

(d) Finally, the workflow is completed and the user can run it from her smartphone.

**Fig. 4.4:** A step-by-step walkthrough of building a workflow.

related to the effects of physical manipulation on the development of CT skills in IL domains.

## 4.4 Evaluation

This section presents the goals, hypotheses, and description of the experiment carried out to test the designed system, following the guidelines of the American Psychological Association [Woh+00].

### 4.4.1 Goals

The goal of the experiment is to evaluate the system design and tangible-based interaction paradigm in a real-world IL scenario, namely a work project meeting. The purpose is to evaluate whether this system might be employed in a given IL scenario and investigate the effects of TUI on the development of CT skills.

### 4.4.2 Research Question

CT is a set of skills that can help people play a more active role in their day to day life, due to the massive impact of software in today's Information Society. Nonetheless, people struggle in dealing with the abstract concepts underlying it, thus it is important to support them in fostering such skills with new and engaging ways without getting in between their actual routine. Constructivist theories suggest that objects manipulation — underpinning TUIs — might be an effective way of aiding end-users in making such highly abstract concepts more tangible and understandable.

Supporting users in developing CT skills is even more important in IL scenarios, where learning is predominantly unstructured, experiential, and noninstitutional, making the experience self-directed and personalised with higher motivations and efficacy.

Promoting CT skills in IL domains through physical objects manipulation might help to lower the barriers of CT by integrating learning in their daily routines and supporting users in dealing with such abstract concepts through their day-to-day experiences.

The proposed system has been developed with the aim of investigating the influence of TUIs on CT skills in multiple IL scenarios. It employs a tangible-based interaction with a tabletop surface — naturally fostering collaboration — and supports inductive processes by means of assembling workflows to solve simple tasks. Thus, it has been designed to easily integrate into users' day-to-day routine to support CT skills.

The main research question addressed by the evaluation is then *"Can physical objects manipulation help foster CT skills in IL domains?"*.

### 4.4.3 Experiment Design

Due to the openness of the range of scenarios analysed — i.e. IL scenarios — an exploratory research design was used, comprising two phases:

1. a combination of oral feedback and observations of end-users engaging with the prototype in a semi in-the-wild scenario, i.e. taking place in a real-world setting and addressing real-world problems, as a way of testing it in a generic IL environment, and

2. semi-structured interviews of domain experts, in order to gather more generic and less domain-specific feedback.

### 4.4.4 Participants

#### First Phase

The study involved end-users selected among Brunel University second-year students in the Department of Computer Science, College of Engineering and Design. As part of their degree they are clustered into groups of 4-6 and tasked with an Android application development project to be undertaken during the course of the year; they are required to meet and work collaboratively every week, normally in the library or in one of the college's meetings rooms, and can use a range of available tools to work together and share information with each other (online dedicated forums or drives, laboratory spaces with coding facilities, etc.). The development is supervised by a teaching staff member, whom they usually meet all together as a group once a week. The objective of these meetings is not to develop the Android application — which is an individual task — but to coordinate and organize a project plan, eventually designing a Gantt diagram to split the workload into individual tasks. This is a time when students self-direct their activities, exchange suggestions, and support each other, being a proper example of a real-world IL scenario.

In particular, three groups of students in their second year participated to the study, made up respectively of four (1 female, 3 males), five (1 female, 4 males), and six (all males) students, reflecting the real project activity requirements and average group size; participants had no prior knowledge of the system, but attended their introductory programming course during their first year, thus they already had some programming and problem-solving experience.

#### Second Phase

Three interaction design experts were involved to get feedback on the featured modality; they were composed of two HCI experts — with a mixture of academic and industry backgrounds — and a product designer, all with more than 20 years of experience in their fields. By involving more experienced participants, the proposed interaction modality has been further evaluated through a less domain-specific point of view, considering a wide range of IL domains for its application.

### 4.4.5 Settings and Procedure

#### First Phase

The study took place within the University facilities, in a room inside the Department of Computer Science designated to students and staff meetings,

where many public interactive displays are already being deployed and used with a researcher present. It was conducted in three different sessions, one for each group of students, lasting half an hour each, and was a preliminary evaluation of the system's feature set and interaction modality.

Due to the inexperience of participants for this phase, eliciting the discussion around the system interaction modality might not have been easily gathered employing only a paper and pencil approach. Thus, students were prompted with it as a "provotype" — i.e. a provocative prototype, namely a prototype that deliberately challenges stakeholders' conceptions by reifying and exposing tensions of existing practice in a context of interest [BD12]; this includes a small set of features highly tailored to the evaluation scenario (i.e. university students collaborating with each other).

The features made available to participants, each rendered with a different block, were: (1) selecting and downloading a file from the user's Dropbox account; (2) displaying a downloaded PDF file or an image on the main tabletop screen; (3) searching for a book in the university library and retrieving its location inside the building depicted in an image; and (4) sending a text document to a specified email address.

For instance, one could pick 1 and 2 (in this order) and the composed application would download a PDF from the user's Dropbox folder and display its content on the big screen (as depicted in figure 4.4); composing 3 and 2 together would result in looking for an available book in the university library and displaying its location on the big screen.

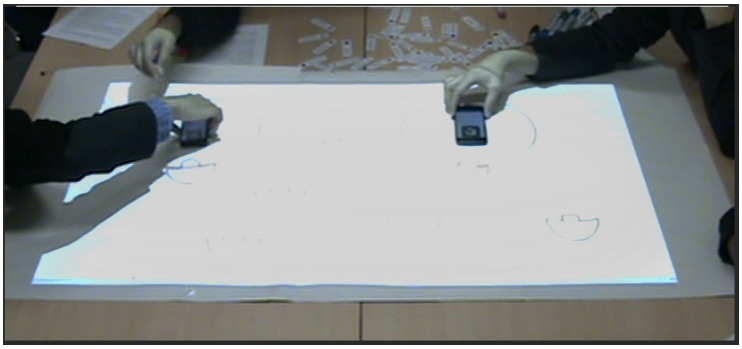


**Fig. 4.5:** One of the participating groups to the study.

Each session lasted 30 minutes and started by briefly introducing the current version of the system to participants, explaining to them how the system works. They were then left to play with it for 15 minutes (figure 4.5), and finally, a semi-structured interview was carried out, mainly focused on the proposed interaction modality.

### Second Phase

Semi-structured interviews were carried out in a controlled environment (figure 4.6), namely during a workshop on the island of Tiree, during the bi-annual Tiree Tech Wave, a gathering of experts in various fields, ranging from interaction designers and artists to computer scientists.

**Fig. 4.6:** The designer study setting.

Individual sessions lasted 45 minutes, starting by introducing the prototype, explaining the rationale behind its design and the targeted scenarios; it was then followed by a brief demonstration of how it works, going through some examples of its usage in real-world IL scenarios. Finally, a semi-structured interview was carried out focusing on the strengths and weaknesses of the prototype in relation to the interaction modality and its applicability in IL domains to foster CT skills, more precisely covering the easiness of the puzzle metaphor, the use of smartphones as tangible objects, future application scenarios, and missing features.

## 4.4.6 Summary

To recap, table 4.1 presents a summary of the specific CT dimensions as defined in [BR12] examined in the evaluation, along with the related assessment approach, as reported in chapter 2.

Moreover, the second phase of this evaluation preliminarily investigated over the interaction modality of TAPAS.

# 4.5 Results

Data collected in both phases were analysed by performing a content analysis on the gathered feedback and observations.

**Tab. 4.1:** Summary of the specific CT dimensions [BR12] considered by the evaluation with the related assessment approach.

| *CT Dimension* | *Description* | *Assessment* |
|---|---|---|
| Concept: sequences | Expressing a particular activity or task as a series of individual steps or instructions that can be executed by the computer. | Project Analysis (First Phase Provotype) |
| Practice: abstracting and modularizing | Building something large by putting together collections of smaller parts. | Artefact-Based Interviews (First Phase Interviews) |

### 4.5.1 First Phase

Overall results point out how the proposed user experience was considered quite satisfactory by participants; as expected, students' feedback mostly focused on missing features and the interaction with the system.

Each group managed to successfully assemble (at least once) two workflows while they were playing with it: the first one started with downloading a PDF file from a Dropbox account and displaying a preview on the main tabletop surface, while the second one started with looking for a specific book in the university library and depicting its location on the main screen.

From the gathered feedback it seems that a TUI is an easy and effective way of interacting with the system throughout the composition of a workflow. Even though all participants are Computer Science (CS) undergraduates, their second-year group project is their first chance of tackling a wider problem-solving scenario, unlike their first year's individual development of smaller applications. This more complex project requires them to learn abstraction and decomposition skills, whilst collaborating with peers. Using the puzzle metaphor and workflows together with tangible interaction helped them build the required CT skills: for instance, collaboratively planning and designing the application's tasks and assigning them to each participant could be a suitable scenario to practice abstraction and composition skills. Moreover, as with API development, the recipe metaphor provides different levels of transparency and abstractions useful to generalize the problem, whilst assembling blocks might help with decomposing a bigger problem into smaller ones [Win11].

Nonetheless, the feedback showed that just a tangible interaction doesn't seem "natural" when it comes to manipulating outputs: every participant trying out the prototype attempted to move images displayed on the screen with their

fingers, suggesting that manipulating items through objects might feel "natural" only when operating in composition/developing mode and there is a perfect mapping between the physical and the digital object, but not when there is actual content the user needs to directly manipulate available on the screen. This might be a result of employing a PbI paradigm to be able to repurpose the system to different domains, which uses a syntactic construct to specify a workflow's instructions as opposed to exploiting only contextual actions on resulting artefacts — i.e. PbD.

From the interaction point of view an interesting remark was made by one of the participants: continuously tracking the smartphone's position on the surface using a fiducial marker requires users not to cover its display with their hands when moving it; however, the hand position on the smartphone might depend on the posture: if a user is standing, he/she might feel more "natural" to hold it from above — thus covering the fiducial marker with the palm — while a seated user might feel more comfortable grabbing it from the side, without covering its display, allowing for its movements to be tracked. Because the majority of existing smartphones are shaped in the same way, it might be worth studying this ergonomic effect in more detail, in order to establish whether users could be provided with a physical enclosure affording the "right" way of holding the smartphone or whether it is a negligible effect when the system runs on horizontal displays placed at a certain distance from the floor.

The same users appear to cope easily with the proposed interaction modality during the workflow editing phase, but a different interaction style has to be devised when it comes to manipulating results.

### 4.5.2 Second Phase

Designers liked the overall idea and the personalization approach for different scenarios, namely using a smartphone as a tangible instead of just a passive object to identify users and link their personal information with the movements they perform on the very same device. In particular, they liked the way blocks use shapes to establish type constraints as it looks like a straightforward way of understanding the composition of workflows to address users' needs.

They recognized the potential of such a system in public spaces, due to its ease of deployment and the cheapness and high availability of the technologies involved: thanks to the simple architecture, it allows deployment in any digitally augmented surface just by installing an RGB camera and running the application on a production server; it can be left in public spaces for a long period of time without the need to perform mundane maintenance operations aimed at adding new features, since users can repurpose it themselves.

Some of their suggestions focused on the way data are presented to users and the use of the dynamic widget to get some input from them: due to the kind of data handled right now — namely lists of files within directories or book titles in a database — it makes sense to prompt users with a choice from a list or offer a keyboard to input raw text. Nevertheless, this will not be the case when dealing with more structured data types, such as points of interest on a map: therefore, they suggested that due to the complexity of workflows that might be put together by end-users, widgets might be designed to be more flexible and personalizable depending on the two-fold level of interaction between the user perspective and data perspective related to the specific data handled by the widget. They emphasized that the two perspectives are interlinked and reinforced mutually. Elements of human-centred information visualization have to be considered in the redesign of the widgets for the next interaction prototype; for instance, by following visual metaphors that incorporate semantic relationships of visual objects both in the physical (tangible) and virtual (digital) world [MS13; Big+14].

Furthermore, interviewees pointed out how the continuous back and forth between interacting with the smartphone to input data and with the large display to assemble workflows might be confusing for users: interacting with two different devices, each one with a different interaction style — i.e. tangible on the tabletop, multi-touch on the smartphone — and different underlying metaphors, requires a relatively high cognitive effort in constantly switching paradigm and some users might also miss what is happening on one device while they are too focused on interacting with the other. That is why interviewees suggested keeping the tabletop as the main interaction focus by providing a mixed interaction modality: moving the smartphone will still be used to assemble the puzzle pieces but once one of them requires a certain input, the widget will appear close to it on the tabletop surface and users will interact with it using their fingers.

The final observation concerns the blocks shapes: although it appears to be quite an easy to grasp concept, its efficacy might be improved by offering some additional visual cues; interviewees suggested that in addition to shapes to indicate functions compatible with the currently generated output, it might highlight the available ones and darken the incompatible ones, even when the former is not available due to network outages or other problems, or even associate colours to shapes.

To recap, there are positive elements in the system design for EUD of workflows to be employed in many existing IL scenarios to foster CT skills, such as the Block-based Programming paradigm, the use of the smartphone as being tangible

and personal, and the ease of prototype deployment in-the-wild due to its low-cost and flexible architecture. Nonetheless, there are some challenges to be addressed in the future in terms of interaction design requirements, such as the flexibility of widgets and improving the visual cues to highlight available functionalities, which might hamper the usability and get in the way of the learning experience.

## 4.6 TAngible Programmable Augmented Surface

The results of the evaluation phase highlighted the validity of employing the developed platform in IL scenarios and prompted for its extension to make it easily repurposable to different domains and include some of the gathered feedback.

In the following, the TAPAS architecture is presented, developed to investigate the influence of TUIs on CT skills in different IL scenarios.

### 4.6.1 Architecture

TAPAS comprises a horizontal tabletop display and an RGB camera capturing the movements of the users' smartphones on the main display's surface using fiducial markers [Bon+13] (i.e. images used as a point of reference when placed in the camera's field of view), as summarized in figure 4.7; it supports — and later extends — the Tangible User Interface Object (TUIO) protocol [Kal+05], already adopted by many research communities within the TUI area as a general and versatile communication interface between tangible tabletop controller interfaces and underlying application layers, which has been designed specifically for interactive multi-touch tabletop surfaces.

When a user logs into the provided Web application running on a smartphone using her credentials, this will display a fiducial uniquely assigned to that account. The system can then track the position of the fiducial across the tabletop surface, knowing to whom it belongs.

Tracking objects using fiducials allows supporting physical object other than smartphones, each providing its own set of sensors and feedback mechanisms if any. The TUIO protocol, however, is quite generic and limited to tracking positions of generic objects in a 2D space, without providing a way for objects to expose their supported I/O interfaces. For this reason, an extension of the TUIO protocol has been proposed [MTO17] and is reported in the following section.

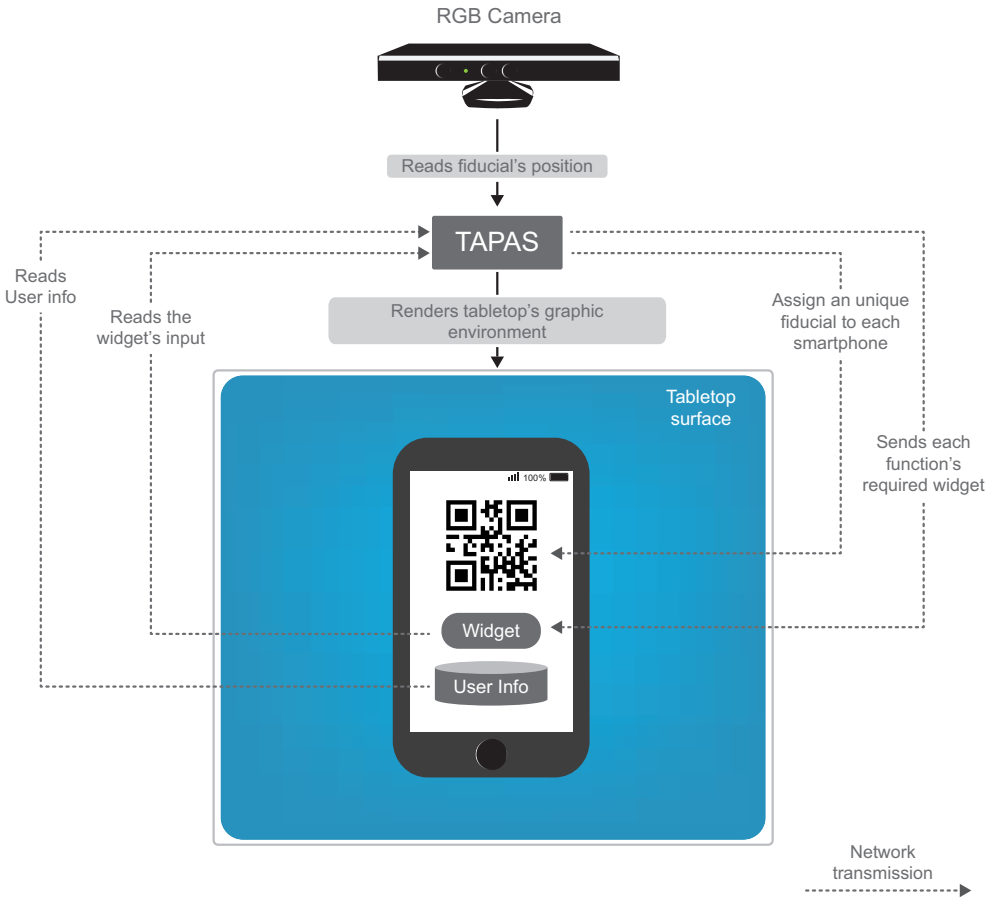



**Fig. 4.7:** The architecture of TAPAS: using a fiducial marker — assigned by the application itself — and an RGB camera, TAPAS can track a smartphone's movements on a tabletop surface; through the smartphone, TAPAS is able to link each and every smartphone's movements to its users and display a corresponding dynamic widget.

### Tangible User Interface Repurposable Object (TUIReO)

As stated before, TAPAS supports the TUIO protocol and extends it to provide developers with a framework — called TUIReO — that allows them to easily experiment with tracking generic programmable objects on a big display. Applications developed with this framework should foster new interactive experiences, featuring EUD with ubiquitous tangibles with advanced feedback and input mechanisms.

The TUIO protocol, as Kaltenbrunner et al. [Kal+05] stated, "is an attempt to provide a general and versatile communication interface between tangible tabletop controller interfaces and underlying application layers. It was designed to meet the needs of tabletop interactive multi-touch surfaces, where the user is able to manipulate a set of objects and draw gestures onto the table surface with the fingertips."

The TUIReO framework (figure 4.8) was designed to provide further interaction capabilities for multi-device environments on top of the TUIO protocol. This allowed experimenting further with TAPAS and block-oriented programmable objects, as reported in the next chapter.

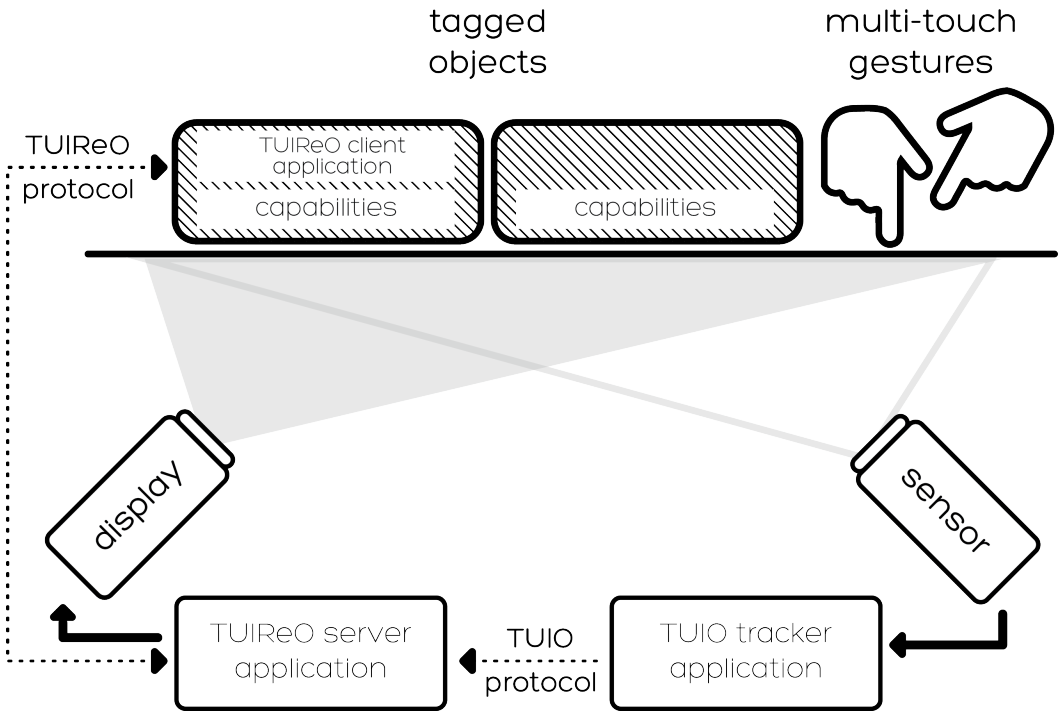



**Fig. 4.8:** TUIReO Framework Architecture

TUIReO is built on top of TUIO to provide an abstraction layer over the capabilities of the tagged smart objects that are already handled by TUIO. It aims to encapsulate the capabilities of a smart object — namely the properties that the physical object offers to the environment and that can be controlled and detected remotely — with its virtual TUIO representation. This enables developers to fully exploit the object features, such as the inputs and outputs channels that it might provide, either physical or digital (in the form of a display or a physical button).

Programmable objects support the following capabilities:

**Interaction capabilities** i.e. buttons, multi-touch events, mid-air gestures.

**Display capabilities** i.e. LEDs, screens.

**Retrieval capabilities** i.e. storage, user's details (e.g., Facebook account).

**Affordance capabilities** i.e. shape, haptic.

The TUIReO environment comprises of (1) a sensor used by the TUIO component to track tagged objects and multi-touch gestures happening over

the tabletop surface, whose positions are transmitted to (2) a display running the TUIReO server application. The server application implements a TUIO client and stores each objects movements and smart capabilities together, controlling the display output of the tabletop and the inputs and outputs of every object through the TUIReO protocol.

TUIReO can be considered a framework for tracking programmable objects on a tabletop and managing their physical and digital properties. Its design stems from lessons learned from the development of the initial prototype reported in section 4.3.2; it has been employed in TAPAS to implement a tangible Block-based programming environment, while a future evaluation needs to be carried out from a development point of view.

### 4.6.2 Implementation

TAPAS has been implemented in the form of two applications running side by side, in a client-server fashion. The first, named Simple Pluggable Range Imaging Tracking Server (SPRITS), implements the TUIReO server application, tracking objects and gestures over a horizontal display. It is developed in C++ and sports an abstraction layer over different depth-cameras (supporting OpenNI-based [@18i] sensors at first) and communication protocols (TUIO being the default one). It can be calibrated to fit any surface size (at a maximum of 2 meters distance) that the camera is pointed at.

SPRITS tracks fiducials and touches issues to the display and streams them on the network through the TUIReO protocol. This feed is picked up by the main TAPAS application: it is developed in JavaScript and runs on the Web. It can run on any PC with a JavaScript-enabled browser, and can be customised using a mix of CSS (to personalise the look and feel of the displayed objects) and JavaScript (to customize the behaviour of each block).

## 4.7 Discussion and Post-Hoc Analysis

The results of the study prompt to address the Research Question set out to be investigated in the introduction.

While this indeed was just a preliminary study on a specific application domain, its findings can certainly be used to highlight some of the advantages and issues with TAPAS. There are initial hints of it being able to support peer-to-peer collaboration (i.e. where all participants have the same role within the group), however it also features chaired modalities by leveraging on the use of smartphones (as suggested in [Cli13]). Moreover, TAPAS supports users in individual activities as well, enabling them to use their preferred tools while carefully considering the resulting privacy issues; indeed, the choice of

employing smartphones as tangible probes in TAPAS was influenced by privacy concerns, allowing it to draw upon user data while keeping the user in control of what she wants to share and with whom. For this reason, further developments are currently in the works on TAPAS' Web app in order to develop a more sophisticated interface that enables users to effectively tweak their privacy settings and control which data TAPAS can have access to.

The proposed interaction modality has been positively received by both end-users and domain experts, paving the way for its application in a wide range of domains where users need to perform simple tasks, either individually or collaboratively, while fostering their CT skills. While this last point will be discussed more in details in the next chapter, TAPAS was successfully to a given IL scenario by developing a set of specific functions, which can be easily assembled by users into different workflows and interact with it.

Two relevant challenges of fostering CT skills with Tangible Programming in IL scenarios can be identified from the findings, one stemming from the results of TAPAS' evaluation with end-users and another from the interviews with domain experts.

Firstly, the user experience seems to differ when the tangible interaction is used for composing services with blocks (positive experience) from when users interact and collaborate on the results of the workflow execution through their smartphones (less positive experience). This could be due to the different set of constructs involved within each stage:

1. Building a workflow requires the user to deal with abstract concepts — like functions and constraints — that are not naturally coupled with any existing physical counterpart; providing users with an intuitive visual metaphor and enabling them to interact with the system in a natural way (through a tangible) might be an effective strategy to help them build the right mental model, together with exposing the right transparency level of the workflows' inner logic in order to improve abstraction and decomposition skills, indeed helping to develop their CT abilities.

2. In an NUI based environment, direct manipulation of contents is more intuitive than using intermediate control mechanisms; hence, when it comes to manipulating results produced by their workflows, users require the interface to be completely transparent, without any syntactical — least of all tangible — artefact to operate on an environment's constructs.

This contrast is also evident from the literature (see section 4.2) highlighting the many differences between the PbD and PbI paradigms: due to its very nature, in a PbD system the composition and execution environments are perfectly

overlapped, i.e. the same artefacts the users operate on to program the system is used also to interact with its results; in Robot Programming by Demonstration, for instance, users teach movements to a robot by simply simulating them directly onto its body. This is radically different from a PbI approach, where the two environments — composition and execution — are generally detached from one another, each one using different metaphors and concepts, e.g., the visual editor of the now-defunct Yahoo! Pipes [Pru07] is used to compose a pipe (data-flow) that generates a specific execution environment made of Graphical User Interface (GUI) elements as designed by the user. While this distinction might be overlooked from an interaction perspective when a system only relies on a GUI, it becomes more relevant when it is about TUIs. Even though PbI seemed the right paradigm to choose in the analysed scenario due to its generalizability and the benefits brought to CT skills, choosing the right paradigm according to the naturalness of interaction is arguably scenario-dependent, as is often the case with Domain Specific Visual Languages.

Secondly, from the study with designers, an interesting challenge has emerged which is related to the use of Visual Languages with TUIs. In particular, the majority of examples found in the literature (section 4.2) — including TAPAS — use Visual Languages when employing a PbI paradigm.

Visual Languages have been widely used within the field of EUD in order to ease the development process for end-users; the interaction paradigm used for Visual Languages is GUI-based, whilst due to the selected scenario a more natural way of allowing EUD would be to support tangible interaction. Studying whether there is an appropriate EUD paradigm for TUI environments requires understanding whether any of the available paradigms, e.g., PbI and PbD, are suitable for Tangible Programming or if, on the contrary, new paradigms need to be introduced. There is some evidence, as in Robot Programming by Demonstration for instance, that PbD is suitable for that specific scenario using Tangible Programming but, as often happens in the EUD community, the solution might be domain dependent.

Finally, as for the preliminary studies just described, TAPAS can support future evaluations over the effects of a tangible PbI paradigm in relation with CT in different IL domains, as further reported in the next chapter.

## 4.8 Threats to Validity

Here are the validity threats to the design of this study.

**Internal Validity** The limited number of components developed and deployed to the tested system could have influenced its usage, thus the findings cannot

be properly generalized for many other contexts. Yet, since TAPAS was employed as a provotype — that is to challenge users by exposing tensions and thus to support design explorations [BD12] — observations related to the interactions users and designers carried out can give a good insight into its real usage. The experimenter effect is concerned with any biasing effects in a study that is due to the actions of the researcher. The researcher attempted to carry out the study as objectively and as accurately as possible without interfering, acting as an observer limited to recording feedback. The subject effect could have determined changes in the participants' behaviour due to being in the study and under observation; in this case, the study was carried out within a traditional university environment with the actual group members participating to the activity.

**External Validity** The results of the study can be generalized only in the context of the scenario where TAPAS was deployed, although it represents quite a common setting. In order to generalize the findings to other scenarios, replication studies are needed.

## 4.9 Contributions

Parts of the work and results described in this chapter have been previously published in the following:

- The Literature Review described in section 4.2 and the First Phase of the Evaluation in section 4.4 have been published in [MT15].
- The Evaluation described in section 4.4 has been published in [TMD15].
- The Preliminary Study described in section 4.3.1 and the Evaluation in section 4.4 have been published in [TMD17].
- The TUIReO Framework described in section 4.6.1 has been published in [MTO17].
- The TAPAS Architecture described in section 4.6 has been published in [DMT18].
- The Preliminary Design of TAPAS described in section 4.3.2 and the Second Phase of the Evaluation in section 4.4 have been published in [Dix+16].

## 4.10 Conclusion

This chapter introduced TAPAS, an application running on a tabletop system, which allows users to develop simple workflows using their smartphones by

combining a tangible and visual interaction. Its architecture and design rationale stem from a two-phases evaluation carried out with the aim of studying the effects of physical manipulation in IL domains on the development of CT skills.

From the first phase of the study (involving second-year undergraduates working in groups) it seems that TAPAS provides a positive user experience and could be used in IL scenarios where learning is self-directed and driven by people's preferences and intentions during their daily activities; a potential side effect caused by employing it to support learning might be a development of CT skills, thanks to its design rationale, which will be discussed and further evaluated in the next chapter.

However, from the findings, it also appears that coupling tangible interaction with a PbI paradigm causes an incompatibility of interaction styles between the composition and the execution environments, where the use of a different tangible-based syntactic construct in the former causes the need for a different interaction style to be used in the latter.

The second phase of the study was focused on its interaction modality and involved a group of interaction design experts; the results show that the system design presents positive elements to support collaboration in IL domains, recognizing the potential of the exploited puzzle metaphor in allowing end-users to develop simple workflows. They also suggested extending the platform in order to cope with more complex data to be manipulated by users. However, it was also pointed out that employing a Visual Language in a TUI system doesn't always provide users with a natural interaction experience, thus further investigations are needed to determine the role of the scenario in the choice of the right paradigm (i.e. PbI vs PbD).

In the next chapter, the main Research Question of this thesis will be laid down in combination with gameplay activities and TAPAS will be employed to test the related hypothesis.

# 5 Fostering Computational Thinking Skills through Gameplay

> *You'll never see a video game being advertised as being easy. Children who do not like school will tell you its not because its too hard. It's because it's boring.*
>
> **— Seymour Papert**
> The Children's Machine, 1993

The previous chapter focused on using a Tangible User Interface (TUI) to foster Computational Thinking (CT) skills in a specific application domain, namely Informal Learning (IL) environments, where learning is mostly self-directed and takes place as people go about their daily activities, driven by their preferences and intentions. This chapter deals with exploiting a TUI to support the development of CT skills in IL domains in combination with gameplay activities.

## 5.1 Introduction

As briefly mentioned in the first chapter, enhancing support for cultivating users' CT skills — and more generally their usual learning experiences — can be optimal when tools and activities are able to keep them in the so-called "Flow state": according to Csikszentmihalyi's theory [NC14], it refers to a state of intense concentration, sustained interest, and enjoyment of the challenge of an activity, when skill level and challenge level of a task are at their highest, allowing users to learn at intense focus. It is hard to obtain such balance, since too much challenge causes anxiety, whereas too little challenge leads to boredom; one of the most common and explored ways of keeping learners in such state is through gameplay, that is by providing them with an engaging challenge and real-time feedback in response to their choices.

Coupling such activities able to keep learners in the Flow state with physical interaction might enhance even further learning of CT skills by leveraging on a sustained engagement level, afforded social interactions, and a concrete representation of the abstract concepts underpinning it.

The Research Question derived from this context and addressed by this chapter is then: *"Can physical objects manipulation provide a playful and engaging way of learning CT skills through gameplay?"*.

The hypothesis is that through the design of an appropriate gameplay involving TUIs and an engaging medium like Virtual Reality (VR), a high efficacy in supporting CT skills can be achieved. Furthermore, using a digital shared surface with tangibles might foster collaborative learning [Sub+07] and physical interaction to further improve CT skills.

The hypothesis was tested by involving 18 UK secondary school Key Stage 4 (15 years old) female students in collaborative gaming sessions with a prototype of a game called TAPASPlay. The young girls had no previous programming experience and only a small group had some previous experience programming with Scratch.

The chapter concludes with a set of observations and recommendations for designing payful and engaging systems to teach CT skills to a wide audience.

## 5.2 Related Works

Digital games proved attractive and engaging for all groups of people and therefore, Game-Based Learning (GBL) has been proposed as one pedagogical framework for developing CT skills [Wei+16]. In order to help to acquire CT skills two main approaches have been introduced in GBL: learning through designing games and learning through gameplay.

### 5.2.1 Learning through Design

Learning through designing games has been studied for many years. In 1996 AgentSheets, designed by Alexander Repenning, was released, even though the first prototype dates back to 1989 [Rep00]. AgentSheets is still used in multiple contexts, from middle to high schools to academic environments, for various purposes such as introducing to programming, supporting storytelling and prototyping simple games. The tool takes its name from the fact that the user develops the program on a grid resembling a spreadsheet, whose cells contain agents. These entities, visualized as pictures, can perform multiple actions like reading Web pages or playing sounds and animations. A graphical interface allows creating "if-then" rules that represent agents' behaviour. Drag-and-drop interaction is supported, which revealed to be an interaction style suitable to people without any programming background. It has been demonstrated that AgentSheets favours CT skills like problem-solving, abstraction, and pattern recognition [Koh+10]. Monteiro et al. [Mon+17] have recently studied how

AgentSheets may improve algorithm design skills, thanks to the logic underlying the creation of rules, as well as teach the automation concept.

Alice is another Visual Programming Environment (VPE), developed at the Carnegie Mellon University starting from 1997 [Her10]. It focuses on creating 3D programming projects and is actually an object-based programming language that allows creating animations and interactive games by defining object behaviour through a drag-and-drop interaction of specific blocks. Alice is a valid tool for supporting storytelling while being exposed to basic programming concepts, both of imperative and object-oriented programming paradigms, without the burden of remembering syntactic constructs. Alice has been successfully employed to assess CT skills in primary and middle school [WWL14; Wer+12].

Similarly, Kodu is an integrated VPE which allows creating games structured as 3D worlds comprising different types of objects able to react to some events [FFM12]. Kodu is aimed at fostering familiarity with basic programming notions, in an intuitive and playful way. It thus supports notions of the imperative programming paradigms, such as sequentiality, conditional instructions, variables and assignments; it also encompasses some concepts of object-oriented programming, such as that of classes and objects, and information hiding. Different studies have been performed to demonstrate the capability of Kodu to improve CT skills such as problem-solving, abstraction, problem decomposition and pattern recognition [FC11; Tou+13].

Perhaps the most influential and versatile tool for learning how to program by designing games is Scratch [Res+09], developed at the MIT and publicly released for the first time in 2005. It is a Visual Programming Language (VPL) whose interaction is made simple thanks to draggable instructions represented by blocks, fitting one another like puzzle pieces. The process of assembling instructions is guided by the different shapes and colours of blocks, suggesting which constraints must be satisfied. One of its biggest strengths is the large and heterogeneous community of users that, combined with the possibility of reusing and remixing other users' code, permits to cooperate, share knowledge and realize complex projects more easily. Scratch is widely considered a successful tool to teach programming to K-12 students and foster CT skills [BR12; Cet16; GPC15].

Robot programming has regarded itself as game-design learning; for instance, in [AD16a] Lego Mindstorms [@18g] is used to improve CT skills of high school students. In this approach, the learner first identifies the goal for the robot and then defines an algorithm, that is, a set of steps to carry out, to accomplish the

goal; then the robot can follow the instructions and act accordingly; the learner may debug the program by observing and tuning robot's behaviour.

Tangible Programming is yet another way to create interactive games by mixing physical objects with a more traditional instruction-based approach. For example, in Tern [HJ07] the goal of introducing children to computer programming is pursued by using small wooden cubes with instructions on their faces. The sequence of instructions results in a series of movements performed by a small robot. In T-Maze [WZW11] the programming phase is conducted in a very similar fashion, with a camera dedicated to reading the programming sequence in real-time. Children from 5 to 9 can create their own maze maps and complete escaping tasks, thus solving simple programming tasks.

### 5.2.2 Learning through Gameplay

Learning through gameplay is another approach to GBL for improving CT skills. It might represent a valid alternative to learning through design, since, as highlighted in [Lee+14], building a game from scratch could be too challenging for novice programmers and thus frustrating for the majority of players.

Among them, [Kaz+12a] proposes Program Your Robot, a game prototype developed to support children in practising the five skills that the authors identified as fundamental for CT: problem-solving, algorithm design, debugging, simulation and socializing. It is a puzzle-solving game in which the player has to assist a simulated robot to reach a certain point on a grid. Players thus design a solution algorithm that the robot will follow, by using symbolic representations of "action commands" (to move the robot in an environment) and "programming commands" (basic constructs such as sequence, selection, iteration, and function). These commands are dragged from their toolbars to specific areas of the environment. Players need to move the robot, activate lights and collect items by proceeding towards different levels of the game. Rewards are obtained in the form of new collectible items, slots or enemies to avoid as the player advances through the game. Program Your Robot is conceived as a serious game and thus differs from the software applications for game design mentioned before, which can be deemed programming languages to all effects. Indeed, tools like AgentSheets or Scratch were designed in order to teach the basics of programming and to show how fun it can be. Instead, Kazimoglu et al. [Kaz+12a] were moved by the goal of creating a game that could foster CT skills. However, also in Program Your Robot the player is exposed to basic programming constructs and thus the gameplay keeps on being strictly related to a programming activity.

CTArcade [Lee+14] is another serious game where players have to design a set of rules that are executed by a character while playing Tic-Tac-Toe. Making these rules explicit is considered an important process, because people often apply them in a natural, perhaps unconscious way and normally there is neither the chance nor the reason to transform this knowledge into abstract instructions. Lee et al. [Lee+14] report a study, implementing a "think aloud" protocol, where 18 children have been observed playing on CTArcade and on paper; the study shows that children articulate more CT skills (problem decomposition, pattern recognition, pattern generalization and algorithmic thinking) using CTArcade compared to playing on paper. However, the analysis was carried out on Tic-Tac-Toe only, a widely popular and common game; therefore, CT was difficult to externalize and observe.

Liu et al. [LCH11] investigate the use of TrainB&P to assist students in developing computational problem-solving abilities. With this simulation game, the students can construct a railway system and design the transportation behaviours of trains on a railway by using several building blocks such as straight, curved, and branch tracks. In particular, the system allows students to program the transportation behaviours and simulate them in a 3D environment. The results of the study, carried out with the participation of 117 students, demonstrate how the gameplay enhances students' motivation and brings them in a flow state during the learning experience.

All the above systems use traditional interaction styles based on keyboard and mouse; on the contrary, even though TAPASPlay shares with them the objective of fostering CT skills through gameplay, it leverages on an interaction style based on tangible objects and VR. Tangible interaction and VR have been chosen to try and increase the playfulness of the system and create an engaging and collaborative learning environment. Furthermore, as investigated in the previous chapter, physical object manipulation might help users deal with abstract concepts, as well as cultivate skills such as abstraction and problem decomposition [WWL14]. In line with [Kaf16], TAPASPlay also aims to foster collaborative learning, that is, it regards CT as a creative and social practice (the "Connecting" Perspective defined in Brennan and Resnick's Framework [BR12]). Lastly, TAPASPlay fits within the realm of Constructionist Video Games [Wei+16], namely computational environments in which players create personally meaningful artefacts to overcome artificial conflicts or obstacles resulting in quantifiable outcomes. In the following, the design and implementation of TAPASPlay is described in detail.

## 5.3 TAPASPlay

TAPASPlay is a turn-taking game intended for end-users with little or no experience in programming, designed to foster their CT abilities by leveraging on physical interaction and keeping them engaged in a Flow state.

### 5.3.1 Design

TAPASPlay has been developed on top of the TAPAS (TAngible Programmable Augmented Surface) system discussed in the previous chapter; it is a block-based programming environment that allows end-users to build simple workflows by assembling different services with the aim of fostering CT skills in IL scenarios. The interaction with TAPAS is carried out using smartphones as tangible objects and digital blocks projected over a tabletop surface.

As in TAPAS, interacting with TAPASPlay requires a tabletop surface, an RGB camera and a smartphone. Smartphone movements on the display or surface are tracked by the RGB camera, which locates the position of a fiducial marker shown on the phone screen and uses it as reference point. TAPASPlay has been implemented as a Web application that is projected on the tabletop surface and is able to interact with players' smartphones. A smartphone application provides players with additional feedback and tools for completing the game. Finally, VR technology is used to visualize the outcome of the game.

As mentioned earlier, TAPASPlay can be regarded as a constructionist video game aimed at providing users with an educational and entertaining experience. It aims at teaching CT skills through gameplay while fostering socialization and thus collaborative learning.

To accomplish these goals, the game has been designed on the basis of the following requirements:

1. The interaction with the game should be based on a puzzle metaphor that proved to be an intuitive approach to find a solution to a given problem (algorithmic thinking) [TMD17]. This means that TAPASPlay has to communicate the existence of constraints and to support the gameplay through puzzle pieces and their shapes, aiding users whilst giving constraints in their selection process.

2. Puzzle pieces should be physically manipulated, in order to favor the appropriation of abstract concepts through tangible interaction [WWL14].

3. The game must be played in player versus player modality, since competition is one of the most important elements of serious games to increase motivation and learning [COO15]. Moreover, each character

might be programmed by a group of players, thus favoring socialization and collaboration within the group.

4. The game must feature a storytelling suitable to a VR representation, which can be visualized by wearing affordable goggles (e.g., Google Cardboard [@18d]).

These requirements led to conceive gameplay around alchemy, that is the (fictitious) process of transmuting metals: players compete to be the best alchemist by forging three swords and three shields, made of three different metals. In order to build each sword, players have a limited amount of energy points they can spend on transmutations, which in turn make a sword earn force points. Transmutations (also called transformations in the following) can be combined together in different ways, allowing users to experiment and practice problem abstraction and decomposition by following the puzzle constraints. The objective is to maximise force points while carefully managing energy points on each sword. Trying to earn force points while finding a tradeoff with energy points is an NP-hard problem that can be solved with different strategies (e.g., greedy algorithm, backtracking), requiring algorithmic thinking in finding even a sub-optimal solution.

In particular, the game is structured in three phases:

1. defining the offensive strategies, by means of forging swords;
2. defining the defensive strategies, by means of forging shields;
3. visualizing the representation of a battle in a VR headset.

### 5.3.2 Forging Swords

The first phase is aimed at fostering different CT skills, such as problem decomposition, algorithmic thinking, abstraction, and iteration. During the first phase, each player creates three offensive strategies by composing three different swords. In order to accomplish that, players have to attach transformations, represented as pieces of a puzzle, to a halo surrounding the user's smartphone on the main display. Each strategy is a sequence of transformations taken from a randomly generated set shown at the beginning of the game on the main display (figure 5.1).

Each half of the tabletop screen is available for a player to forge the swords. The halo, with its three hilts, follows the movement of the dragged smartphone and, when a collision with a puzzle piece is detected, such a piece is attached to the vertically oriented hilt given that the move is allowed by the game rules.

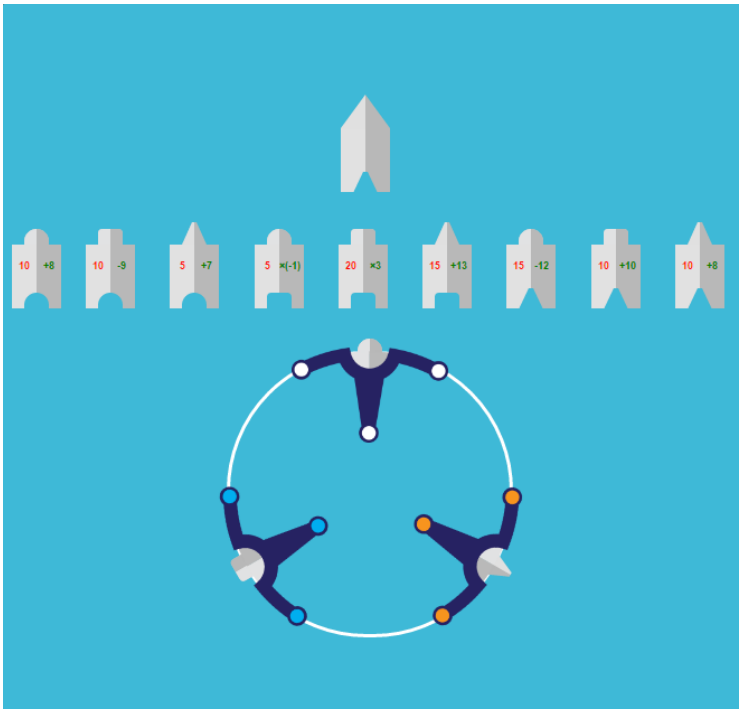



**Fig. 5.1:** Initial state of the game: the set of transformations is displayed, as well as the main halo with three hilts and the final piece.

The three swords are defined one at a time, so that each can have a different set of puzzle pieces available for the players, avoiding repetitions and increasing in difficulty. For instance, in figure 5.2, players are creating their first swords.

**Fig. 5.2:** Forging swords through tangible and puzzle-like interaction.

A hilt attached to the main halo surrounding the players smartphone represents the starting point of the sword (figure 5.3a), while the final piece has a shape that resembles the tip (figure 5.3b).

**(a)** An example of initial state of the sword.

**(b)** An example of final piece.

**Fig. 5.3:** Examples of initial and final pieces of a sword.

Every puzzle piece has an input and an output shape. There are three shapes in total, round, square and triangular, which in turn correspond to three types of metal, namely bronze, iron, and steel. So, if a puzzle piece has a round input shape and a triangular output shape as in figure 5.4a, it is equivalent to a transformation that turns bronze into steel. Each sword is made of a different type of metal, determined by the shape of the final puzzle piece. For example, in figure 5.4b the shape of the final piece is triangular and thus a steel sword has been forged.

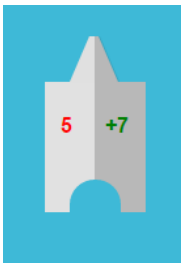
The aim of this first phase is to maximize the force points of each sword, which can be earned by attaching transformations to the sequence. However, every transformation consumes a number of energy points. More precisely, a transformation is a tuple of four values: (1) an input shape, (2) an output shape, (3) a number of energy points, displayed on the transformation (left half in figure 5.4a), and (4) the force points gained, displayed on the transformation as well (right half in figure 5.4a).

In order to apply a transformation, two conditions need to be fulfilled: (1) the input shape of the transformation is the same as the output shape of the last transformation attached to the sword (or, if the transformation applied is the first one, the input shape has to be the same as the output shape of the initial state of the sword); and (2) the alchemist must have a number of energy points greater or equal than the one shown on the transformation.

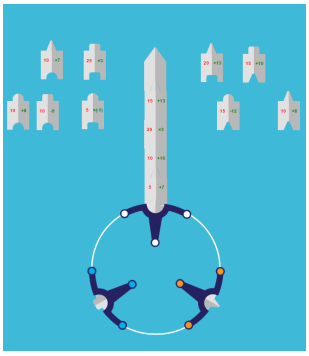
Once a transformation is applied (supported by a "magnetic effect" on the puzzle piece provided by the system), the energy points of the alchemist are decreased by the energy points of the transformation, while the force points of the strategy can be increased, decreased or multiplied, depending on the operation suggested by the transformation.

The initial state of each sword consists of an output shape attached to a hilt on the halo, a number of force points, and a number of energy points. The final

state (figure 5.4b) is reached when the player is satisfied with its sequence of transformations and decides to — and can — attach the final piece to the sword.




(a) An example transformation.



(b) A sword example.

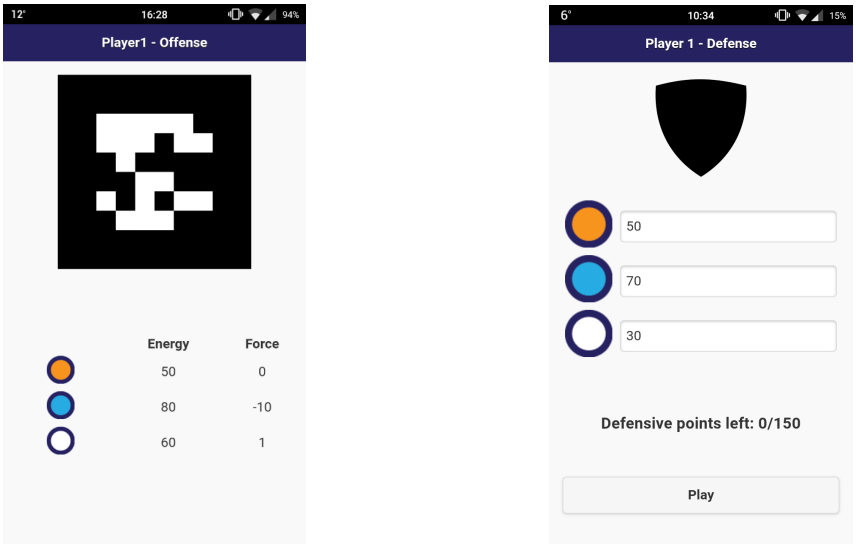
**Fig. 5.4:** Forging a sword by composing transformations.

Players can see a feedback of their operation on their smartphone since force and energy points presented on their screen are updated according to the values displayed on the transformation. See for example figure 5.5a, where the correspondence between swords and values displayed on the smartphone is given by the cue balls matching the gems of the hilts showed on the halo.

Maximizing force points requires to decompose the problem of forging a sword in smaller transformation problems (problem decomposition) and then solve sub-problems by selecting transformations through a greedy technique (i.e. selecting among the available pieces the one that gives more force points) or backtracking (i.e. going back to a previous decision point when reaching an invalid solution), thus fostering algorithmic thinking. During this activity, a player could mentally combine two transformations and regard them as a new piece with its own input and output shape, which can be used to forge the sword; therefore, abstraction comes into play during solution creation. Finally, the definition of each offensive strategy prompts the player to iterate the steps of evaluation and selection of transformations until she is satisfied with the solution and moves over to the next one. The overall activity is then repeated three times, one for each sword, always with a game scenario (i.e. the available puzzle pieces) of increased difficulty.

### 5.3.3 Forging Shields

The second phase is functional to the playability of the game and not strictly related to fostering CT skills, even though it requires some analytical abilities. In this phase, the players must define their defensive strategies, which consist of allocating a number of defence points into three shields, each one corresponding to a different metal. The choice should be based on a couple of




**(a)** The energy and force points of the swords.

**(b)** The defence points in the shields.

**Fig. 5.5:** The smartphone application

considerations: how the player guesses the opponent distributed force points on the different swords and which transformations have been chosen to build her own swords. For instance, if a player was not able to optimize the strategy for the steel sword, then she might consider allocating most defence points into the steel shield, in order to counterpoise her weak offensive strategy. To allocate defence points into the shields, each player interacts with a simple interface displayed on the smartphone (figure 5.5b).

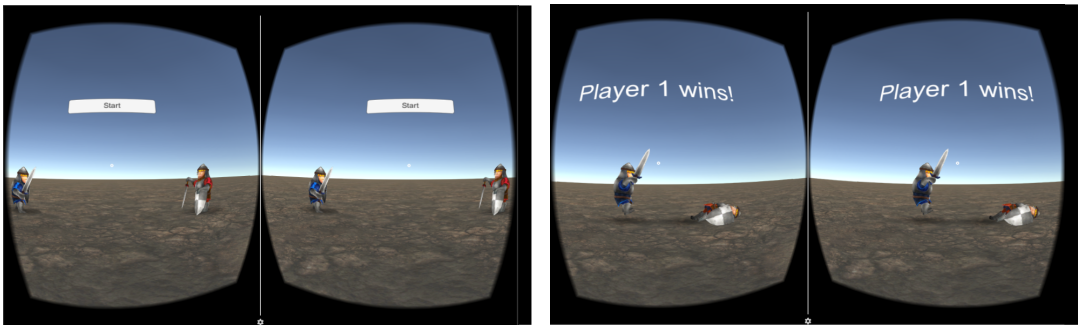
### 5.3.4 Enjoying the battle in VR

The third phase of the game was designed to foster debugging capabilities, one of the main CT skills highlighted in the literature. In the current version of TAPASPlay, however, this feature is limited to the visualization of the battle in VR.

More precisely, when both the previous phases of the game are completed, a simple Android application showing a VR video is made available from the server. Both players must wear VR goggles to enjoy the content of the video. The server provides each player with a different video on the basis of the scores it has received from the Web application. For instance, if player 1, who used the halo with blue hilts, reached the highest score, the video shows a knight wearing a blue armour defeating the opponent dressed in red; otherwise, a video with reversed roles is played. The VR video shows two knights armed with sword and shield. In the beginning, a button with the "Start" label is visualized and a pointer placed at the centre of the user's sight suggests that gazing at it will allow playing the animation (figure 5.6a). After having pressed the button, the two

knights approach the centre of the scene and, when they are close enough, they start duelling. They exchange a few hits for a little while, then the knight on the left takes a few steps back, runs toward the opponent and launches the decisive blow. The wounded knight falls on the ground and, while the winner cheers, a text appears on the background, confirming which player won (figure 5.6b).

Supporting this VR visualization to inspect players' solutions provides an engaging way to visualise the outcome of the strategies and supports the learning of debugging capabilities within the gameplay by providing a way of tracking them.




**(a)** Beginning of a duel. **(b)** End of a duel: player 1 has won.

**Fig. 5.6:** Visualizing the battle in VR.

## 5.4 Evaluation

This section presents the goals, hypotheses, and description of the experiment carried out to test TAPASPlay and address the Research Question stated in the introduction of this chapter, following the guidelines of the American Psychological Association [Woh+00].

### 5.4.1 Goals

The goal of the experiment is to evaluate whether TAPASPlay can be employed to develop CT skills while providing a fun and enjoyable gameplay. The purpose is to evaluate whether physical manipulation might foster CT skills through gameplay in IL domains.

### 5.4.2 Research Questions

User participation in system development can be effectively achieved by creating the conditions for their empowerment by supporting them in appropriating those CT skills [Win06] necessary for understanding and contributing to the system evolution.

Gameplay offers an opportunity to teach high-level CT concepts indirectly in an engaging way to an ever wider audience, keeping them in a continuous Flow state.

In order to effectively teach CT skills, gameplay should foster playful engagement and collaborative learning. TUIs exploit humans' innate dexterity for objects' manipulation to aid understanding of abstract concepts — such as the ones involved by coding and CT [TMD17]. Coupling them with gameplay also improves the playfulness [Pri+03], relieving users from the mental burden carried by more artificial interaction paradigms.

One of the other benefits of TUIs is their natural predisposition towards collaboration, which is beneficial not only for learning in general but especially for fostering CT skills [WWL14]. Moreover, it is also a distinctive trait of some of the most engaging games.

The Research Question derived from this context and addressed by this chapter is then: *"Can physical objects manipulation provide a playful and engaging way of learning CT skills through gameplay?"*.

### 5.4.3 Experiment Design

An exploratory research design was used, comprising oral feedback, observations, and a post-test survey.

### 5.4.4 Participants

The participants of the experiment were 18 UK secondary school female students of Key Stage 4 (15 years old) coming from different schools in the London area. None of them had a solid programming background, but a small subset (3 of them) had a little experience in block-based programming with Scratch. No prerequisite knowledge was required to perform the tasks, and none of the participants had prior knowledge of neither TAPAS nor TAPASPlay. A brief introduction to the system and the game was provided to the experiment group.

### 5.4.5 Settings and Experiment Tasks

The study was conducted within the Brunel University London facilities, in a laboratory inside the Department of Computer Science, as depicted in figure 5.7.

Participants were presented with a prototype of TAPASPlay and tasked them with playing a single-turn game, i.e. forging one sword each; the VR visualisation and the defense strategy definition were purposively removed in order to focus the evaluation just on the proposed interaction and the effects of a TUI-based gameplay on CT skills in IL domains.

The developed game scenario (i.e. the available puzzle pieces at the beginning of a game, as depicted in table 5.1) was meant to provide players with many

**Fig. 5.7:** The study setting inside a university laboratory.

strategies and different difficulty levels. This way participants could implement different strategies based on their skill level and progression throughout the game.

Indeed, the puzzle shapes provide constraints and introduce conflict within the game: a player needs to maximise the force points of her sword while using the available puzzle pieces in an appropriate order. Moreover, each puzzle piece has a cost, and the sum of the puzzle pieces' cost that makes up a sword mustn't go over 100. The initial and final shapes were both triangular.

**Tab. 5.1:** The TAPASPlay scenario tested with participants.

| | Input Shape | Output Shape | Energy Cost | Force Points |
|---|---|---|---|---|
| 1: | ∪ | ⊓ | 10 | +8 |
| 2: | ∪ | ∧ | 10 | −9 |
| 3: | ∨ | ∩ | 5 | ×(−1) |
| 4: | ∨ | ∧ | 20 | ×3 |
| 5: | ∨ | ⊓ | 15 | +13 |
| 6: | ⊔ | ⊓ | 5 | +8 |
| 7: | ⊔ | ∩ | 15 | −12 |
| 8: | ⊔ | ∧ | 10 | +10 |
| 9: | ⊔ | ⊓ | 10 | +8 |

### 5.4.6 Procedure

Participants were randomly clustered in 8 groups, 6 groups of 2 people each and 2 groups of 3, and the study was conducted in four different sessions, one for each individual game match played: one between the two teams of 3 people, and the rest between the other teams, paired in a randomised fashion.

All interactions with the tabletop surface and oral feedback provided during the game were recorded. At the end, a random participant from each group was asked to fill in a short questionnaire about her experience. Responses were given in terms of (Q1) enjoyment, (Q2) collaboration, and (Q3) interactivity, using a Likert scale from 1 (strongly disagree) to 5 (strongly agree) — like well-known similar questionnaires (e.g., SUS [Bro96]) — with a neutral midpoint (neither agree nor disagree) in order to avoid directing the choices towards just negative or positive sentiments. The questionnaire presented the following statements:

Q1 I enjoyed playing TAPASPlay.

Q2 I think TAPASPlay can be a fun game to play with friends.

Q3 I enjoyed playing TAPASPlay on a tabletop by moving a smartphone.

A 6 minutes average duration of each match was devised from an early internal playtesting phase of the game scenario. Each experimental session was then planned to last 15 minutes in total: 3 minutes for a brief explanation of how the game works and its rules, 2 minutes of practice, 8 minutes for the actual match, and 2 minutes to give feedback and fill in the questionnaire.

### 5.4.7 Summary

To recap, table 5.2 presents a summary of the specific CT dimensions as defined in [BR12] examined in the evaluation, along with the related assessment approach, as reported in chapter 2.

Moreover, this evaluation preliminarily investigated over the enjoyability (questionnaire's Q1), collaboration support (questionnaire's Q2), and interactivity (questionnaire's Q3) of the learning experience provided by TAPASPlay.

## 5.5 Results

The collected data were analysed by (1) performing a content analysis on the feedback, (2) summarising the recorded game strategies employed by participants, and (3) analysing the questionnaire results. The findings are reported below.

**Tab. 5.2:** Summary of the specific CT dimensions [BR12] considered by the evaluation with the related assessment approach.

| *CT Dimension* | *Description* | *Assessment* |
|---|---|---|
| Concept: sequences | Expressing a particular activity or task as a series of individual steps or instructions that can be executed by the computer. | Project Analysis |
| Practice: being incremental and iterative | Changing the plan in response to approaching a solution in small steps. | Artefact-Based Interviews |
| Practice: abstracting and modularizing | Building something large by putting together collections of smaller parts. | Design Scenarios |

### 5.5.1 Feedback

The overall response was positive, but participants needed some practice at first to get going assembling swords. Indeed, all the groups managed to play and successfully assembly a sword in the given time.

All groups were involved in playing the game, and all groups members were trying to work out the right sequence of pieces to assemble the strongest sword possible. All participants looked engaged in the discussion with their peers, offering support and ideas to solve the problem: none of the participants was left isolated from their groups, regardless of size. TAPASPlay fosters collaboration and stimulates discussion by having users around a table interacting with objects laying on it.

A pointer received during the experiment was related to the proposed interaction modality. The TUI seemed easily grasped and manoeuvred by participants, but the individual control point (i.e. the smartphone) and lack of support for mixed interaction (e.g., multi-touch) were pointed out by someone as the main hiccup to a better gameplay experience. Yet this promoted an off-the-screen collaboration where group members interact with each other to reach a decision, while in the end, one member took control of the smartphone on the tabletop surface. It fostered group discussion and kept all team members involved in the decision process, balancing the need of each member to experiment with her own ideas and contribute to the overall discussion.

Another point that was made from some participants during the study was related to the fall-back mechanism of TAPASPlay. A simple undo action, triggered with a button on the smartphone interface, detaches the latest

piece that was attached to the current halo and puts it back to its original position on the board, redistributing the energy points consumed. Three groups used the undo action, while the others preferred discussing the strategies amongst themselves and act once they figured out the whole process, without experimenting it first. Designing an improved fall-back mechanism that properly represents the system status and capabilities across different heterogeneous devices with a TUI is still an open question for Cross-Device interaction research [Hou+17].

### 5.5.2 Strategies

A total of 10 complete strategies were produced by all the groups, as summarised in table 5.3: on the left the strategies are reported as ordered sequences of the puzzle pieces in table 5.1 with their corresponding numbers, as they were assembled during the game; the number of groups that issued a strategy is reported on the right when greater than 1. Six groups completed a single strategy each and decided to end the game there, while the other two groups — not playing in the same session — kept experimenting further after completing one sword, and issued two complete strategies each: the first successfully completed strategies (b) and (d), while the second (e) and (f).

**Tab. 5.3:** The strategies completed by participating groups. On the left, numbers correspond to the puzzle pieces labelled in table 5.1, while on the right the number of times (when greater than 1) the corresponding strategy was issued during the study is reported.

| | | |
|---|---|---|
| (a) | $5 \rightarrow 8$ | |
| (b) | $5 \rightarrow 6 \rightarrow 8$ | (×2) |
| (c) | $5 \rightarrow 9 \rightarrow 8$ | (×3) |
| (d) | $5 \rightarrow 6 \rightarrow 9 \rightarrow 8$ | (×2) |
| (e) | $5 \rightarrow 9 \rightarrow 6 \rightarrow 8$ | |
| (f) | $5 \rightarrow 9 \rightarrow 6 \rightarrow 8 \rightarrow 4$ | |

The average number of pieces used to complete a sword was 3.4, with a standard deviation of 0.8. The majority of the strategies issued were naïve, in that they were built through a greedy algorithm using just a small number of pieces and without multiple trials (strategies (a), (b), and (c) in table 5.3), while the other strategies were a bit more complex and sometimes required more effort — i.e. backtracking, deferring completing a strategy directly and using more pieces to gain more points — to be discovered.

### 5.5.3 Survey

Lastly, the questionnaire was filled by a randomly chosen participant from each group, whose results are reported in table 5.4.

**Tab. 5.4:** The survey results.

| | Strongly Disagree | Disagree | Neither Agree nor Disagree | Agree | Strongly Agree |
|---|---|---|---|---|---|
| Q1 | 0 | 0 | 3 | 3 | 2 |
| Q2 | 0 | 1 | 2 | 4 | 1 |
| Q3 | 1 | 0 | 0 | 4 | 3 |

All three statements proposed (section 5.4.6) were rated positively by the majority of respondents. The game interactivity (Q3) was the most positively perceived with an average score of 4, 5 out of 8 participants judged the enjoyment (Q1) more than neutral, with an average score of 3.875. The collaboration aspect of TAPASPlay (Q2) seemed to have been appreciated by most of the participants, with three exceptions, with an average score of 3.625.

## 5.6 Discussion and Post-Hoc Analysis

From the results of the study, the Research Question set out to be investigated in the introduction can be addressed.

First, the experience provided by TAPASPlay was received positively from participants: the feedback recorded during the game reports a positive reception from users, which is also confirmed by the survey results (Q1). Devising an engaging gameplay is fundamental in order to foster CT skills, having to remove all the extra mental burden that comes from unnecessary game mechanics. TUIs provide a natural way of interacting with the game, without any artificial means of control, making the game easy to play and fun. TAPASPlay was also positively received in terms of interactivity, as evidenced by the survey results (Q3).

What is even more remarkable is that such results were achieved within a group of young girls, even though the game wasn't designed with this specific user group in mind: gender imbalance and under-representation have always been major issues affecting the Silicon Valley and the whole tech community in general, making it necessary to come up with new strategies to correct this phenomenon [Tza+17; Bec+06; Huf02; CJ98]. Engaging young girls in Science, Technology, Engineering and Mathematics (STEM) activities means empowering them with the right tools to actively participate and take control of the issues coming up in the future, allowing them to take on a more central role in the science and technology sector.

The survey results report how TAPASPlay provides an engaging gameplay (Q1) using a TUI that makes it a highly interactive experience (survey's Q3).

Collaboration is yet another aspect worth discussing in more detail: TAPASPlay was designed from the ground up on top of TAPAS to support collaboration in order to foster CT skills. Indeed, Kazimoglu et al. [Kaz+12a] report that socialization is another CT skill fostered by learning through gameplay. The feedback obtained from participants and the results of the survey (Q2) confirm that TAPASPlay was well received as a collaborative game, stimulating discussions amongst teammates and fostering a stimulating learning environment. The gaming experience led users to socialize by continuously sharing thoughts about their approaches during the game, thus stimulating cooperative strategy development useful in co-design processes.

The lack of group members isolation is yet another benefit of the proposed gameplay observed during the study: softening the "lone wolf" effect — described as the preference to work alone and dislike of group processes — can positively affect team performance and improve learning [BDG05]. Indeed, one can seldom observe an even participation in learning groups, especially big ones [MT15], thus smoothing group participation level is a favourable consequence of balancing interaction style, groups activity and size. However, this effect will have to be validated further in future studies with larger groups size.

Moreover, strategies issued by participating groups can be analysed to discuss how TAPASPlay fosters CT skills: interestingly, the strategies adopted by the groups were quite different from each other, making use of a different amount of puzzle pieces and of different algorithmic strategies (improving from a greedy strategy to backtracking). This, depending on the developed scenario, can provide the right conditions for supporting CT skills at different levels, allowing players to assembly different strategies and reach for the hardest ones to build as their skills progress (i.e. low floor, high ceiling [RBE16]).

Another result worth pointing out is what happened to the two groups that issued more than one strategy (section 5.5.2). The first one assembled strategy (e) in table 5.3, then (f); this progression is evidence of a divide-et-impera approach, i.e. the result of decomposing a strategy into subproblems and recursively solve them: once the problem has been solved with the first strategy, the group recognised that the solution could be extended by adding an extra piece, gaining more points.

Next, the second one assembled strategy (b) in table 5.3, then (d); perhaps even more deeply than before, this progression is evidence of abstracting the building of a sword and recognising that another piece can be added without

changing the input and output of the whole strategy, thus completing it and gaining more points.

The rationale behind the design of TAPASPlay also provides other pointers towards fostering CT skills, stemming from its design. TAPASPlay detaches composition from execution [TMD17] by offering two different interaction styles and tools: puzzle-based interaction on a tabletop display and a smartphone are used for composing the strategy (problem-solving); whilst VR to support and make more exciting checking solution execution. This mechanism fosters the design-debug-run stages, three key aspects of CT [Kaz+12a], or in other terms, the process of problem formulation-solution expression-execution and evaluation [RBE16].

Moreover, while automation is supported by VR, analysis, abstraction and problem decomposition are types of reasoning that players are supposed to apply when trying to maximize the force points, under the constraints represented by shapes and limited energy points. As a matter of fact, the choice of displaying all transformations together at the beginning of a game makes it deliberately complex for the player to formulate a straightforward solution. On the other hand, if the player is "lazy" and does not want to apply a methodic decomposition process, but merely tries to satisfy the constraints (i.e. greedy strategy), a solution would be reached, but chances are that it won't be a good one in terms of force points. Therefore, the player will try to "fix it" by analyzing it and identifying the weakest subsequence of transformations. Hence, the solution would be reformulated by replacing the poor part with a different sequence of pieces (i.e. backtracking). This process might be repeated several times, inducing the player to iteratively apply the model of CT process proposed in [RBE16].

All these skills are indeed crucial for the end-users to play an active role in the algorithmic solution proposed and discussed with technologists, therefore enhancing the formers' active participation to system development and evolution, aiding them in understanding and selecting the right solution while helping them modelling the problem.

## 5.7 Threats to Validity

There are several validity threats to the design of this study.

**Internal Validity** The limited number of participants allowed to properly reason about different effects found during the study, but a more extended experiment testing all the game phases with more users needs to be designed in order to properly validate the effects over isolation of team members, which

cannot be definitive yet. The experimenter effect is concerned with any biasing effects in a study that is due to the actions of the researcher. The researcher attempted to carry out the study as objectively and as accurately as possible, acting as an observer limited to recording feedback, but the survery responses might be affected. The subject effect could have determined changes in the participants' behaviour due to being in the study and under observation; in this case, the study was carried out within a traditional learning environment during a series of workshops with similar game activities.

**External Validity** The sample was a group of only female students coming from different schools of the London area, which was a proper starting point to validate engagement in an under-represented group within the tech industry, but in extending this work TAPASPlay should be tested with a more diverse and international user group to investigate different effects. The lack of a mixed modality which fosters on-screen collaboration and support for an advanced fall-back mechanism limited in-game experimentation and prevented certain uses which might have affected the observed results.

**Construct Validity** Due to experiment time limitations, the post-test questionnaire was filled by a random member of each group and was limited to three questions designed to measure different aspects of the experience. This, together with the limited number of respondents, might have affected the results, even though the survey results weren't used alone, but rather cross-referenced them with the in-game oral feedback from participants.

## 5.8 Contributions

Parts of the work and results described in this chapter have been previously published in the following:

- The Design of TAPASPlay described in section 5.3 has been published in [Mal+17b; Mal+17a; Fog+17].
- The Evaluation of TAPASPlay reported in section 5.4 and its Design in section 5.3 have been published in [TFM19].

## 5.9 Conclusion

The growing interest in CT is witnessed by very recent literature [Yad+17], which describes how CT is becoming more and more important in student and teacher education. In this chapter, CT skills are shown to be fundamental to allow end-users to collaborate to system design and evolution at use time. For this reason, contrarily to other block-based approaches, in TAPASPlay blocks

do not represent programming statements (like for example, the "if-then" block in Scratch) but remain at a higher level of abstraction, to promote problem decomposition abilities rather than programming ones.

Like TAPAS, TAPASPlay considers TUIs and physical object manipulation fundamental tools to make user activities more engaging. Indeed, it has been demonstrated that tangible programming has the potential to help children cultivate skills such as abstraction and problem decomposition [WWL14].

In this chapter the design rationale behind TAPASPlay was presented, a turn-taking serious game using gameplay to foster CT skills by making learners experience engaging and social. In particular, it contributes to the research trend that explores learning through gameplay [Kaz+12a] — instead of learning through designing systems — in fostering CT skills. The prototype was employed in a study with a group of secondary school girls that investigated the effects of physical objects manipulation on learning CT skills through gameplay. The results showed some evidence that TAPASPlay offers an engaging and playful environment to develop CT skills.

TAPASPlay is, however, a first attempt at fostering CT skills of end-users through gameplay. Further experiments testing all three game phases with different user groups and game scenarios will be carried out to demonstrate the validity and robustness of the idea. Furthermore, several extensions of TAPASPlay have been already planned, in order to tailor the system to end-users' characteristics and introduce different levels of complexity in the game. At the moment, only a VR simulation of the battle is available as an outcome of the game; however, the system could be extended adding a more interactive functionality that better resembles the debugging activity, in which players can compare step-by-step how they built their swords and eventually see what was the optimal solution.

The next chapter concludes the thesis by summarising the results in light of the original Research Question to be addressed, recaps the contributions and implications, and discusses possible future research directions.

# Conclusion

# 6

> *Science is a way of thinking much more than it is a body of knowledge.*
>
> **— Carl Sagan**
> Science Friday interview, 1996

This chapter concludes the thesis by discussing the results of all the studies carried out in relation to each investigated Research Question and in light of the main one set out to be addressed in the introduction. The research contributions and implications are laid out, highlighting also its current limitations. Finally, possible future research directions are summarised.

## 6.1 Summary

Thanks to the amount of software driving our whole lives, coding and Computational Literacy (CL) have become essential skills for an ever wider audience. Nonetheless, the highly abstract concepts involved by them still constitute a huge barrier to a widespread appropriation of technology for the general public. Yet, these skills are not just related to programming itself, but they contribute to developing the so-called Computational Thinking, namely all those thinking abilities reflecting core concepts and practices of Computer Science (CS). This set of skills can enable people to actively participate and address today's challenges with the help of modern technological tools, solving complex problems and express their solutions using a computer.

This thesis set out to investigate the effects of physical manipulation on the development of Computational Thinking (CT) skills: according to the constructivist theories of Jean Piaget [PI69], exploiting human's innate dexterity for objects manipulation in the physical world and its afforded social interactions could be an effective way of aiding users in practising abstract concepts as CT. Physical manipulation sits at the core of Tangible User Interfaces (TUIs), a digital interaction paradigm designed with the aim of providing users with an easy to use interface that can benefit inexperienced people. Such a paradigm, often used to support the interaction of young children with technology in the classroom, could be employed to promote CT skills by providing users with a physical representation of the concepts involved [McN04; Hor+09], acting as a scaffold between the real world and digital [WWL14].

Chapter 3 presented an overview of existing tools based on Visual Programming Languages (VPLs) currently used in programming sessions around the world to introduce CT to the most diverse audiences. An investigation was carried out to verify to what extent — if at all — such tools support the collaborative learning of CT skills, specifically of two computational Concepts [BR12], sequences and loops. The reported quantitative study compared collaboratively working learners with individual working ones for the purpose of evaluating CT skills development in the context of using a VPL-based technological tool in a real-world introductory programming session. Unfortunately, the results of the study didn't show any significance, but the lessons learned in designing and carrying out the study were used in the following experiments to design better tools that support CT skills in collaborative environments.

Chapter 4 addressed the main thesis investigation over the effects of physical manipulation on the development of CT skills in a specific set of educational domains, namely Informal Learning (IL) ones, where learning is mostly self-directed and takes place as people go about their daily activities, driven by their preferences and intentions. A preliminary design of TAPAS (TAngible Programmable Augmented Surface) was presented, a software platform combining its digital and physical features to promote CT skills in different IL domains. TAPAS' design stems from a workshop with expert designers used to collect insightful ideas and design challenges related to its development. TAPAS was then used to investigate the effects of physical manipulation on the development of such skills through a two-phase qualitative study carried out both with undergraduates working in groups and expert designers. The results showed that TAPAS provides a positive user experience and could be used effectively in IL scenarios; a potential side effect caused by employing it to support learning might be a development of those CT skills associated with the computational Concept of sequences, and the practice of abstracting and modularizing [BR12] thanks to its design rationale, but more studies are needed in order to investigate this effect further.

Finally, Chapter 5 took an extra step towards addressing the main thesis Research Question and presented an investigation on the effects of physical manipulation on the development of CT skills through gameplay activities, a common scenario often used in introductory programming courses. An extension of TAPAS, called TAPASPlay, was presented to address it, which consists of a turn-taking serious game using gameplay to foster CT skills by making learners experience engaging and social. The developed prototype was employed in a study with a group of secondary school girls, whose results

showed some evidence that TAPASPlay might offer an engaging and playful environment to develop CT skills, specifically in relation to the computational Concept of sequences, and the Practices of being incremental and iterative, and abstracting and modularizing [BR12].

## 6.2 Research Contributions

The main research contributions of this work concern the use of TAPAS as a repurposable tool to study the effects of physical manipulation on the development of CT. It was designed following the results of a workshop with experts that shaped it to be repurposed to different IL scenarios. TAPAS was then evaluated in a sample scenario and showed promising results in terms of provided user experience; it was then repurposed (as TAPASPlay) to a different IL scenario through gameplay, showing promising results in relation to CT skills development and user engagement.

The feasibility of repurposing TAPAS to different IL domains have been shown, allowing it to be used in many other scenarios to support learning sequences and abstracting and modularizing. TAPASPlay demonstrated that supporting CT with gameplay and physical manipulation can prompt for a sustained user engagement while offering new ways of assisting skill progression amongst learners, with respect to computational practices like abstracting and modularizing and being incremental and iterative.

Moreover, the results of the study reported in Chapter 3 highlighted the need for developing new CT tools supporting learning sequences and loops that better leverage on collaboration amongst peers to enhance learning.

Finally, some of the main challenges faced by supporting CT skills with physical objects manipulation in IL domains were highlighted, based on insights from a two-phase study carried out with end-users and interaction designers.

To recap, the main Research Question addressed by this thesis was: *"Can the collaborative and cognitive naturalness of physical objects manipulation at the basis of Tangible User Interfaces aid the understanding of core algorithmic principles and thus improve end-users' Computational Thinking skills?"*.

The following Key Research Questions were formulated and addressed throughout the thesis in order to support and investigate the main Research Question in detail:

- *"Do existing VPL-based tools support the collaborative learning of CT skills?"* From the results of the study carried out in Chapter 3, a definite answer cannot be yet provided, but further studies are needed to show that existing tools are leveraging on collaborative learning.

- *"Can physical objects manipulation help foster Computational Thinking skills in Informal Learning domains?"* The results of the two-phases study carried out in Chapter 4 are promising, suggesting that TUIs can provide support for developing CT skills in such domains.
- *"Can physical objects manipulation provide a playful and engaging way of learning CT skills through gameplay?"* The results of the study reported in Chapter 5 suggest that combining a TUI with gameplay can develop CT skills and support the collaborative learning.

Ultimately, further studies are needed to fully address the main Research Question, but from the preliminary studies carried out and reported in this thesis, one can argue that physical manipulation provides support for developing CT skills and might represent the natural evolution of existing tools currently used in educational environments. Future studies should cover more dimensions of CT as defined by Brennan and Resnick [BR12], with particular reference to CT Perspectives, which are usually hard to capture. Different Design Scenarios should be developed, with a more extensive set of instructions that can be issues by users in order to provide more breadth to the available measures.

## 6.3 Research Limitations

The research reported in this thesis has some limitations that highlight the need for further future research.

**Scenarios** The two main studies directly investigating the main thesis Research Question in Chapters 4 and 5 were carried out in Brunel Facilities with students from either the University itself or the surrounding High Schools. Replication studies are needed in order to generalise their findings to students from other areas and — since supporting CT in IL domains relates to a very heterogeneous audience — to other age groups. Familiarity with technology should be another confounding variable worth considering in these cases.

**Perspectives** The TAPAS platform was tested only from a user perspective, analysing its effects on supporting CT skills. On the other hand, its architecture, as discussed in Section 4.6.1, is meant to allow its repurposing to different scenarios; indeed, TAPAS was repurposed to a different IL domain in combination with gameplay activities and rebranded as TAPASPlay in Chapter 5. This process was carried out by its original author, thus the ease of such activity needs to be properly evaluated by analysing it from a developer perspective. For this reason, TAPAS' source code is going to be published with an open source license, allowing other developers to repurpose it do different domains.

**Sample Sizes** Most of the studies carried out and reported in this thesis involved small groups of participants, typically less than 20. Such small groups of students were appropriate for those preliminary qualitative studies, but bigger and longitudinal ones are needed to reveal the real effects of using the proposed systems and investigate initial claims.

**Significance** The study in Chapter 3 was carried out to evaluate CT skills development in the context of using a VPL-based tool for collaboratively working learners; even though the sample size was quite substantial (88 students), the results failed to show statistical significance, which prompts for further investigations over this matter.

**Assessment** As pointed out a number of times throughout the thesis, the research community still lacks an accepted definition of CT and, thus, a unified way of measuring it. In carrying out the studies reported, different (mostly qualitative) measures have been employed to attempt to capture effects correlated with the development of such skills, but if new methods and tools need to be designed to better support learners, researchers must keep on investigating this matter and devise an appropriate framework that can be used, highlighting the pros and cons of existing ones.

## 6.4 Fostering Computational Thinking Skills

In this thesis, a range of tools and methods supporting CT skills have been proposed. Many useful pointers have been raised throughout this work, which are collected and summarised in the following.

Collaboration — as suggested by Piaget's constructivist theory of learning [PI69] — provides a promising way of fostering CT skills in different scenarios, but seems slightly overlooked by current research in this field.

TUIs present an engaging way of fostering CT skills by supporting users in practising abstract concepts by leveraging on physical objects manipulation and encourage collaboration amongst users, which in turns support their learning activities.

Gameplay could be used to engage young girls in Science, Technology, Engineering and Mathematics (STEM) activities, empowering them with the right tools to actively participate and take control of the issues coming up in the future, whilst allowing them to take on a more central role in the science and technology sector.

## 6.5 Future Work

**Support more tangibles** By taking full advantage of the Tangible User Interface Repurposable Objects (TUIReOs) protocol discussed in Section 4.6.1, TAPAS deployments can support a wide range of tangible objects, which might expose their specific function and provide digital features based on their shape, exploiting their physical affordance.

**Reduce setup requirements** The setup required to run TAPAS is quite complex and requires specific hardware that needs to be mounted in dedicated spaces. Further work should optimize its digital footprint and requirements, in order to ease the needed setup and enable its ubiquitous deployments.

# Bibliography


[11] *Report of a Workshop on the Pedagogical Aspects of Computational Thinking*. Washington, D.C., USA: National Academies Press, Aug. 2011 (cit. on p. 24).

[AAG16] Ana Liz Souto O. de Araujo, Wilkerson L. Andrade, and Dalton D. Serey Guerrero. "A systematic mapping study on assessing computational thinking abilities". In: *2016 IEEE frontiers in education conference (FIE)*. IEEE. 2016 (cit. on pp. 15, 16).

[AD16a] Soumela Atmatzidou and Stavros Demetriadis. "Advancing Students' Computational Thinking Skills Through Educational Robotics". In: *Robotics and Autonomous Systems* 75.PB (Jan. 2016) (cit. on pp. 11, 16, 67).

[AD16b] Soumela Atmatzidou and Stavros Demetriadis. "Advancing students computational thinking skills through educational robotics: A study on age and gender relevant differences". In: *Robotics and Autonomous Systems* 75 (2016) (cit. on p. 19).

[AW13] Alissa Antle and Alyssa F. Wise. "Getting down to details: Using theories of cognition and learning to inform tangible user interface design". In: *Interacting with Computers* 25.1 (Jan. 2013) (cit. on p. 21).

[BD12] Laurens Boer and Jared Donovan. "Provotypes for participatory innovation". In: *the Designing Interactive Systems Conference*. New York, NY, USA: ACM, 2012 (cit. on pp. 51, 62).

[BDG05] Terri Feldman Barr, Andrea L. Dixon, and Jule B. Gassenheimer. "Exploring the "Lone Wolf" Phenomenon in Student Teams". In: *Journal of Marketing Education* 27.1 (2005) (cit. on p. 83).

[BE14] Anja Balanskat and Katja Engelhart. *Computing our future: Computer programming and coding-Priorities, school curricula and initiatives across Europe*. 2014 (cit. on p. 24).

[Bec+06] Laura Beckwith, Margaret Burnett, Valentina Grigoreanu, and Susan Wiedenbeck. "Gender HCI: What about the software?" In: *Computer* 39.11 (2006) (cit. on p. 82).

[Ber+14] Marina Umaschi Bers, Louise Flannery, Elizabeth R. Kazakoff, and Amanda Sullivan. “Computational thinking and tinkering: Exploration of an early childhood robotics curriculum”. In: *Computers & Education ()* 72 (Mar. 2014) (cit. on pp. 11, 26).

[Big+14] Alex Bigelow, Steven Drucker, Danyel Fisher, and Miriah Meyer. “Reflections on how designers design with data”. In: *the 2014 International Working Conference*. New York, NY, USA: ACM, 2014 (cit. on p. 55).

[Bil+08] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. “Robot Programming by Demonstration”. In: *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer, 2008 (cit. on p. 41).

[Bon+13] Quentin Bonnard, Séverin Lemaignan, Guillaume Zufferey, et al. *Chilitags 2: Robust fiducial markers for augmented reality and robotics*. 2013 (cit. on p. 56).

[BR12] Karen Brennan and Mitchel Resnick. “New frameworks for studying and assessing the development of computational thinking”. In: *In AERA 2012*. 2012 (cit. on pp. 13–17, 31, 52, 53, 67, 69, 79, 80, 88–90).

[Bro96] John Brooke. “SUS-A quick and dirty usability scale”. In: *Usability evaluation in industry* 189.194 (1996) (cit. on p. 79).

[Bun07] Alan L. Bundy. *Computational thinking is pervasive*. 2007 (cit. on pp. 2, 3).

[Bur09] Margaret Burnett. “What Is End-User Software Engineering and Why Does It Matter?” In: *Virtual and Mixed Reality*. Berlin, Heidelberg: Springer, 2009 (cit. on p. 2).

[Cab+16] Federico Cabitza, Daniela Fogli, Rosa Lanzilotti, and Antonio Piccinno. “Rule-based tools for the configuration of ambient intelligence systems: a comparative user study”. In: *Multimedia Tools and Applications* 76.4 (Apr. 2016) (cit. on p. 2).

[CC16] Joelle Coutaz and James L. Crowley. “A First-Person Experience with End-User Development for Smart Homes”. In: *Pervasive Computing, IEEE* 15.2 (2016) (cit. on p. 2).

[Cet16] Ibrahim Cetin. “Preservice Teachers Introduction to Computing: Exploring Utilization of Scratch”. In: *Journal of Educational Computing Research* 54.7 (2016) (cit. on p. 67).

[CGS99] Anit Chakraborty, Randy Graebner, and Tom Stocky. “LOGO: A project history”. In: *Unpublished paper for MITs* 6 (1999) (cit. on pp. 8, 17).

[Che+07] Tzu-Yi Chen, Gary Lewandowski, Robert McCartney, Kate Sanders, and Beth Simon. “Commonsense computing”. In: *ACM SIGCSE Bulletin* 39.1 (Mar. 2007) (cit. on p. 26).

[CJ98] Justine Cassell and Henry Jenkins, eds. *From Barbie to Mortal Kombat: Gender and Computer Games*. Cambridge, MA, USA: MIT Press, 1998 (cit. on p. 82).

[Cli13] Sarah Clinch. "Smartphones and pervasive public displays". In: *IEEE Pervasive Computing* 12.1 (Feb. 2013) (cit. on p. 59).

[COO15] Nergiz Ercil Cagiltay, Erol Ozcelik, and Nese Sahin Ozcelik. "The effect of competition on learning in games". In: *Computers & Education* 87 (2015) (cit. on p. 70).

[DH14] Peter Dalsgaard and Kim Halskov. "Tangible 3D tabletops". In: *interactions* 21.5 (Jan. 2014) (cit. on p. 44).

[DHP07] Christian Dorner, Jan Hess, and Volkmar Pipek. "Improving Information Systems by End User Development: A Case Study". In: *ECIS 2007 Proceedings* (2007) (cit. on p. 2).

[DiS01] Andrea A. DiSessa. *Changing minds: Computers, learning, and literacy*. MIT Press, 2001 (cit. on p. 9).

[DiS18] Andrea A. DiSessa. "Computational Literacy and The Big Picture Concerning Computers in Mathematics Education". In: *Mathematical Thinking and Learning* 20.1 (2018) (cit. on p. 9).

[Dix+16] Alan Dix, Alessio Malizia, Tommaso Turchi, et al. "Rich digital collaborations in a small rural community". In: *Collaboration Meets Interactive Spaces*. Springer, 2016 (cit. on pp. 62, 115).

[DMT18] Giuseppe Desolda, Alessio Malizia, and Tommaso Turchi. "A tangible-programming technology supporting end-user development of smart-environments". In: *Proceedings of the 2018 International Conference on Advanced Visual Interfaces*. ACM. 2018 (cit. on pp. 62, 116).

[DP12] Jose Danado and Fabio Paternò. "Puzzle: A Visual-Based Environment for End User Development in Touch-Based Mobile Phones". In: *Virtual and Mixed Reality*. Berlin, Heidelberg: Springer, 2012 (cit. on p. 2).

[FB97] George W. Fitzmaurice and William Buxton. *Graspable user interfaces*. University of Toronto, 1997 (cit. on p. 20).

[FC11] Allan Fowler and Brian Cusack. "Kodu Game Lab: Improving the Motivation for Learning Programming Concepts". In: *Proceedings of the 6th International Conference on Foundations of Digital Games*. FDG '11. Bordeaux, France: ACM, 2011 (cit. on p. 67).

[FFM12] Allan Fowler, Teale Fristce, and Matthew MacLauren. "Kodu Game Lab: a programming environment". In: *The Computer Games Journal* 1.1 (May 2012) (cit. on pp. 25, 67).

[FG06] Gerhard Fischer and Elisa Giaccardi. “Meta-design: A Framework for the Future of End-User Development”. In: *End User Development*. Dordrecht: Springer, Jan. 2006 (cit. on p. 2).

[Fis17] Gerhard Fischer. “Exploring Design Trade-offs for Quality of Life in Human-centered Design”. In: *Interactions* 25.1 (Dec. 2017) (cit. on p. 10).

[Fog+17] Daniela Fogli, Federico Danesi, Alessio Malizia, Tommaso Turchi, and David Bell. “Sustaining Cultures of Participation by Fostering Computational Thinking Skills through Game-Play”. In: *on CEUR in Proceedings of GHItaly* (2017) (cit. on pp. 85, 116).

[Fra+16] Rita Francese, Michele Risi, Genoveffa Tortora, and Maurizio Tucci. “Visual Mobile Computing for Mobile End-Users”. In: *IEEE Transactions on Mobile Computing* 15.4 (2016) (cit. on p. 2).

[GP13] Shuchi Grover and Roy Pea. “Computational Thinking in K-12: A Review of the State of the Field”. In: *Educational Researcher* 42.1 (Feb. 2013) (cit. on p. 24).

[GPC15] Shuchi Grover, Roy Pea, and Stephen Cooper. “Designing for deeper learning in a blended computer science course for middle school students”. In: *Computer Science Education* 25.2 (2015) (cit. on p. 67).

[Gra+12] Jeff Gray, Hal Abelson, David Wolber, and Michelle Friend. “Teaching CS principles with app inventor”. In: *Proceedings of the 50th Annual Southeast Regional Conference*. ACM. 2012 (cit. on p. 25).

[Han+09] Mark Hancock, Otmar Hilliges, Christopher Collins, Dominikus Baur, and Sheelagh Carpendale. “Exploring tangible and direct touch interfaces for manipulating 2D and 3D information on a digital table”. In: *the ACM International Conference*. New York, NY, USA: ACM, 2009 (cit. on pp. 21, 41).

[HCB12] Michael Horn, Jordan Crouser, and Marina Umaschi Bers. “Tangible interaction and learning: The case for a hybrid approach”. In: *Personal and Ubiquitous Computing* 16.4 (Apr. 2012) (cit. on p. 21).

[Her10] Charles W. Herbert. *An Introduction to Programming Using Alice 2.2*. 2nd. Boston, MA, USA: Course Technology Press, 2010 (cit. on pp. 17, 25, 67).

[HJ07] Michael S. Horn and Robert J. K. Jacob. “Tangible Programming in the Classroom with Tern”. In: *CHI ’07 Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’07. San Jose, CA, USA: ACM, 2007 (cit. on p. 68).

[Hor+09] Michael Horn, Erin Treacy Solovey, Jordan Crouser, and Robert Jacob. “Comparing the use of tangible and graphical programming languages for informal science education”. In: *the SIGCHI Conference*. New York, NY, USA: ACM, 2009 (cit. on pp. 3, 21, 24, 87).

[Hou+17] Steven Houben, Nicolai Marquardt, Jo Vermeulen, et al. “Opportunities and Challenges for Cross-device Interactions in the Wild”. In: *interactions* 24.5 (Aug. 2017) (cit. on p. 81).

[Huf02] Chuck Huff. “Gender, Software Design, and Occupational Equity”. In: *Inroads – SIGCSE Bulletin* 34.2 (June 2002) (cit. on p. 82).

[IU97] Hiroshi Ishii and Brygg Ullmer. “Tangible bits”. In: *the SIGCHI Conference*. New York, NY, USA: ACM, 1997 (cit. on pp. 3, 20, 21, 24).

[Jac+08] Robert Jacob, Audrey Girouard, Leanne Hirshfield, et al. “Reality-based interaction”. In: *Proceeding of the twenty-sixth annual CHI conference*. New York, NY, USA: ACM, 2008 (cit. on p. 20).

[Jor+07] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. “The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces”. In: *Proceedings of the 1st international conference on Tangible and embedded interaction*. ACM. 2007 (cit. on p. 20).

[Jos+14] Beate Jost, Markus Ketterl, Reinhard Budde, and Thorsten Leimbach. “Graphical programming environments for educational robots: Open roberta-yet another one?” In: *Multimedia (ISM), 2014 IEEE International Symposium on*. IEEE. 2014 (cit. on p. 25).

[Kaf16] Yasmin B. Kafai. “From Computational Thinking to Computational Participation in K–12 Education”. In: *Communications of the ACM* 59.8 (July 2016) (cit. on p. 69).

[Kal+05] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza. “TUIO: A protocol for table-top tangible user interfaces”. In: *Proceedings of the The 6th International Workshop on Gesture in Human-Computer Interaction and Simulation*. 2005 (cit. on pp. 56, 57).

[Kaz+12a] Cagin Kazimoglu, Mary Kiernan, Liz Bacon, and Lachlan MacKinnon. “Learning Programming at the Computational Thinking Level via Digital Game-Play”. In: *Procedia Computer Science* 9 (Jan. 2012) (cit. on pp. 13, 68, 83, 84, 86).

[Kaz+12b] Cagin Kazimoglu, Mary Kiernan, Liz Bacon, and Lachlan Mackinnon. “A serious game for developing computational thinking and learning introductory computer programming”. In: *Procedia-Social and Behavioural Sciences* 47 (2012) (cit. on pp. 16, 18).

[KGK16] Filiz Kalelioglu, Yasemin Gülbahar, and Volkan Kukul. “A framework for computational thinking based on a systematic research review”. In: *Baltic Journal of Modern Computing* 4.3 (2016) (cit. on p. 14).

[Kir+09] David Kirk, Abigail Sellen, Stuart Taylor, Nicolas Villar, and Shahram Izadi. “Putting the physical into the digital: Issues in designing hybrid interactive surfaces”. In: *People and Computers XXIII Celebrating People and Technology - Proceedings of HCI 2009*. Microsoft Research Cambridge, Cambridge, United Kingdom. Dec. 2009 (cit. on pp. 21, 41).

[Koh+10] Kyu Han Koh, Ashok Basawapatna, Vicki Bennett, and Alexander Repenning. “Towards the Automatic Recognition of Computational Thinking for Adaptive Visual Language Learning”. In: *Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing*. VLHCC ’10. Washington, D.C., USA: IEEE, 2010 (cit. on p. 66).

[LCH11] Chen-Chung Liu, Yuan-Bang Cheng, and Chia-Wen Huang. “The Effect of Simulation Games on the Learning of Computational Problem Solving”. In: *Computers & Education* 57.3 (Nov. 2011) (cit. on p. 69).

[Lee+11] Irene Lee, Fred Martin, Jill Denner, et al. “Computational thinking for youth in practice”. In: *Acm Inroads* (2011) (cit. on p. 26).

[Lee+13] Jisoo Lee, Luis Garduño, Erin Walker, and Winslow Burleson. “A tangible programming tool for creation of context-aware applications”. In: *the 2013 ACM international joint conference*. New York, NY, USA: ACM, 2013 (cit. on p. 41).

[Lee+14] Tak Yeon Lee, Matthew Louis Mauriello, June Ahn, and Benjamin B. Bederson. “CTArcade: Computational thinking with games in school age children”. In: *International Journal of Child-Computer Interaction* 2.1 (Jan. 2014) (cit. on pp. 8, 16, 19, 68, 69).

[LF09] James J. Lu and George H. L. Fletcher. “Thinking About Computational Thinking”. In: *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* 41.1 (Mar. 2009) (cit. on p. 10).

[Lie00] Henry Lieberman. “Programming by example (introduction)”. In: *Communications of the ACM* 43.3 (Mar. 2000) (cit. on pp. 40, 41).

[LPW06] Henry Lieberman, Fabio Paternò, and Volker Wulf. *End User Development (Human-Computer Interaction Series)*. Berlin, Heidelberg: Springer, 2006 (cit. on p. 2).

[Mal+08] John Maloney, Kylie Peppier, Yasmin B. Kafai, Mitchel Resnick, and Natalie Rusk. “Programming by choice: Urban youth learning programming with scratch”. In: *SIGCSE’08 - Proceedings of the 39th ACM Technical Symposium on Computer Science Education*. MIT Media Laboratory. Dec. 2008 (cit. on p. 26).

[Mal+11] Alessio Malizia, Andrea Bellucci, Paloma Díaz, Ignacio Aedo, and Stefano Levialdi. “EStorys: A visual storyboard system supporting back-channel communication for emergencies”. In: *Journal of Visual Languages & Computing* 22.2 (Apr. 2011) (cit. on p. 43).

[Mal+17a] Alessio Malizia, Daniela Fogli, Federico Danesi, Tommaso Turchi, and David Bell. “TAPASPlay: A game-based learning approach to foster computational thinking skills”. In: *Visual Languages and Human-Centric Computing (VL/HCC), 2017 IEEE Symposium on*. IEEE. 2017 (cit. on pp. 85, 115).

[Mal+17b] Alessio Malizia, Tommaso Turchi, Federico Danesi, Daniela Fogli, and David Bell. “TAPASPlay: a Tangible Game-Based Learning Approach to Foster Computational Thinking Skills”. In: *IS-EUD 2017* (2017) (cit. on pp. 85, 116).

[Mar07] Paul Marshall. “Do tangible interfaces enhance learning?” In: *the 1st international conference*. New York, NY, USA: ACM, 2007 (cit. on p. 21).

[McN04] Timothy S. McNerney. “From turtles to Tangible Programming Bricks: Explorations in physical language design”. In: *Personal and Ubiquitous Computing* 8.5 (Jan. 2004) (cit. on pp. 3, 24, 87).

[MD17] Dave Mason and Kruti Dave. “Block-based versus flow-based programming for naive programmers”. In: *Blocks and Beyond Workshop (B&B), 2017 IEEE*. IEEE. 2017 (cit. on p. 25).

[MK07] Rick Morgan and John Klaric. “AP Students in College: An Analysis of Five-Year Academic Careers. Research Report No. 2007-4.” In: *College Board* (2007) (cit. on p. 1).

[MOB09] Andrew Manches, Claire O’Malley, and Steve Benford. “Physical manipulation: evaluating the potential for tangible designs”. In: *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*. New York, NY, USA: ACM, 2009 (cit. on p. 21).

[Moh+11] Siti Nor Hafizah Mohamad, Ahmed Patel, Rodziah Latih, et al. “Block-based programming approach: Challenges and benefits”. In: *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics, ICEEI 2011*. Universiti Kebangsaan Malaysia, Bangi, Malaysia. IEEE, Oct. 2011 (cit. on p. 46).

[Mon+17] Ingrid Teixeira Monteiro, Luciana Cardoso deCastro Salgado, Marcelle Pereira Mota, Andréia Libório Sampaio, and Clarisse Sieckenius de Souza. “Signifying software engineering to computational thinking learners with AgentSheets and PoliFacets”. In: *Journal of Visual Languages and Computing* 40 (2017). Semiotics, Human-Computer Interaction and End-User Development (cit. on p. 66).

[Mor12] Julián Moreno. “Digital Competition Game to Improve Programming Skills”. In: *Journal of Educational Technology & Society* 15.3 (2012) (cit. on p. 10).

[MS13] Aditi Majumder and Behzad Sajadi. “Large area displays: The changing face of visualization”. In: *Computer* 46.5 (May 2013) (cit. on p. 55).

[MT15] Alessio Malizia and Tommaso Turchi. “Pervasive displays in the wild: Employing end user programming in adaption and re-purposing”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2015 (cit. on pp. 21, 40, 62, 83, 115).

[MTO17] Alessio Malizia, Tommaso Turchi, and Kai A. Olsen. “Block-oriented programming with tangibles: An engaging way to learn computational thinking skills”. In: *Blocks and Beyond Workshop (B&B), 2017 IEEE*. IEEE. Oct. 2017 (cit. on pp. 56, 62, 116).

[Mug+09] Elena Mugellini, Denis Lalanne, Bruno Dumas, et al. “MEMODULES as Tangible Shortcuts to Multimedia Information”. In: *Virtual and Mixed Reality*. Berlin, Heidelberg: Springer, 2009 (cit. on p. 41).

[Mül+14] Jens Müller, Tobias Schwarz, Simon Butscher, and Harald Reiterer. “Back to tangibility”. In: *the 2014 International Working Conference*. New York, NY, USA: ACM, 2014 (cit. on pp. 21, 41).

[MV99] Victoria J. Marsick and Marie Volpe. “The Nature and Need for Informal Learning”. In: *Advances in Developing Human Resources* 1.3 (1999) (cit. on p. 42).

[Nam+15] Immaculate K. Namukasa, Donna Kotsopoulos, Lisa Floyd, et al. *From computational thinking to computational participation: Towards Achieving Excellence through Coding in elementary schools*. Tech. rep. University of Western Ontario, 2015 (cit. on pp. 2, 24, 26).

[NC14] Jeanne Nakamura and Mihaly Csikszentmihalyi. “The concept of flow”. In: *Flow and the foundations of positive psychology*. Springer, 2014 (cit. on pp. 4, 65).

[Orr09] Genevieve Orr. “Computational thinking through programming and algorithmic art”. In: *SIGGRAPH Talks 2009* (2009) (cit. on p. 17).

[Pap80] Seymour Papert. *Mindstorms: children, computers, and powerful ideas*. New York, NY, USA: Basic Books, Inc., Jan. 1980 (cit. on pp. 8, 26).

[PI69] Jean Piaget and Barbel Inhelder. *The psychology of the child*. 1969 (cit. on pp. 3, 8, 21, 24, 87, 91).

[Pri+03] Sara Price, Yvonne Rogers, Mike Scaife, Danae Stanton, and Helen Neale. “Using tangibles to promote novel forms of playful learning”. In: *Interacting with Computers* 15.2 (2003). Interaction Design and Children (cit. on p. 77).

[PRI08] Amanda J. Parkes, Hayes Solos Raffle, and Hiroshi Ishii. “Topobo in the wild”. In: *Proceeding of the twenty-sixth annual CHI conference*. New York, NY, USA: ACM, 2008 (cit. on pp. 21, 41).

[PRM01] John F. Pane, Chotirat Ann Ratanamahatana, and Brad A. Myers. "Studying the language and structure in non-programmers' solutions to programming problems". In: *International Journal of Human-Computer Studies* 54.2 (Feb. 2001) (cit. on p. 26).

[Pru07] Mark Pruett. *Yahoo! Pipes*. First. O'Reilly, 2007 (cit. on p. 61).

[RBE16] Alexander Repenning, Ashok Basawapatna, and Nora Escherle. "Computational thinking tools". In: *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2016 (cit. on pp. 12, 83, 84).

[Rep00] Alexander Repenning. "AgentSheets: An interactive simulation environment with end-user programmable agents". In: *Interactions* (2000) (cit. on pp. 17, 66).

[Res+09] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, et al. "Scratch: Programming for all". In: *Communications of the ACM* 52.11 (Nov. 2009) (cit. on pp. 3, 13, 18, 25, 46, 67).

[RWI10] Alexander Repenning, David Webb, and Andri Ioannidou. "Scalable game design and the development of a checklist for getting computational thinking into public schools". In: *the 41st ACM technical symposium*. New York, NY, USA: ACM, 2010 (cit. on p. 25).

[SN12] Arnan Sipitakiat and Nusarin Nusen. "Robo-Blocks". In: *the 11th In-ter-na-tion-al Con-fer-ence*. New York, NY, USA: ACM, 2012 (cit. on p. 41).

[SSA17] Valerie J. Shute, Chen Sun, and Jodi Asbell-Clarke. "Demystifying computational thinking". In: *Educational Research Review* 22 (2017) (cit. on pp. 10, 11).

[Sub+07] Sriram Subramanian, David Pinelle, Jan Korst, and Vincent Buil. "Tabletop Collaboration through Tangible Interactions". In: *16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2007)*. IEEE, 2007 (cit. on pp. 21, 66).

[SW13] Cynthia Selby and John Woollard. *Computational thinking: the developing definition*. Jan. 2013 (cit. on p. 11).

[TD16] Matti Tedre and Peter J. Denning. "The long quest for computational thinking". In: *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. ACM. 2016 (cit. on p. 14).

[TFM19] Tommaso Turchi, Daniela Fogli, and Alessio Malizia. "Fostering computational thinking through collaborative game-based learning". In: *Multimedia Tools and Applications* 78.10 (2019), pp. 13649–13673 (cit. on p. 85).

[TG15] Jake Trower and Jeff Gray. “Blockly language creation and applications: Visual programming for media computation and bluetooth robotics control”. In: *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM. 2015 (cit. on p. 25).

[TM16a] Tommaso Turchi and Alessio Malizia. “A Human-Centred Tangible approach to learning Computational Thinking”. In: *ICST Transactions on Ambient Systems* 3.9 (2016) (cit. on pp. 36, 116).

[TM16b] Tommaso Turchi and Alessio Malizia. “Fostering computational thinking skills with a tangible blocks programming environ-ment”. In: *Visual Languages and Human-Centric Computing (VL/HCC), 2016 IEEE Symposium on*. IEEE. IEEE, 2016 (cit. on pp. 36, 115).

[TMD15] Tommaso Turchi, Alessio Malizia, and Alan Dix. “Fostering the adoption of Pervasive Displays in public spaces using Tangible End-User Programming”. In: *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2015 (cit. on pp. 21, 40, 62, 115).

[TMD17] Tommaso Turchi, Alessio Malizia, and Alan Dix. “TAPAS: A tangible End-User Development tool supporting the repurposing of Pervasive Displays”. In: *Journal of Visual Languages & Computing* 39 (Apr. 2017) (cit. on pp. 21, 62, 70, 77, 84, 115).

[Tou+13] David S. Touretzky, Daniela Marghitu, Stephanie Ludi, Debra Bernstein, and Lijun Ni. “Accelerating K-12 Computational Thinking Using Scaffolding, Staging, and Abstraction”. In: *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. SIGCSE ’13. Denver, Colorado, USA: ACM, 2013 (cit. on p. 67).

[Tza+17] Katerina Tzafilkou, Nicolaos Protogeros, Charalampos Karagiannidis, and Adamantios Koumpis. “Gender-based behavioral analysis for end-user development and the RULESattributes”. In: *Education and Information Technologies* 22.4 (2017) (cit. on p. 82).

[Vee13] Annette Vee. “Understanding Computer Programming as a Literacy”. In: *Literacy in Composition Studies* (2013) (cit. on pp. 2, 23, 24, 27).

[Voo+15] Joke Voogt, Petra Fisser, Jon Good, Punya Mishra, and Aman Yadav. “Computational thinking in compulsory education: Towards an agenda for research and practice”. In: *Education and Information Technologies* 20.4 (Dec. 2015) (cit. on pp. 8, 10).

[Wan+12] Danli Wang, Yang Zhang, Tianyuan Gu, Liang He, and Hongan Wang. “E-Block”. In: *Adjunct proceedings of the 25th annual ACM symposium*. New York, NY, USA: ACM, 2012 (cit. on p. 41).

[Wei+09] Malte Weiss, Julie Wagner, Yvonne Jansen, et al. “SLAP widgets”. In: *the SIGCHI Conference*. New York, NY, USA: ACM, 2009 (cit. on pp. 21, 41).

[Wei+16] David Weintrop, Nathan R. Holbert, Michael Horn, and Uri Wilensky. "Computational Thinking in Constructionist Video Games". In: *IJGBL* 6.1 (2016) (cit. on pp. 66, 69).

[Wer+12] Linda Werner, Jill Denner, Shannon Campe, and Damon Chizuru Kawamoto. "The Fairy Performance Assessment: Measuring Computational Thinking in Middle School". In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. SIGCSE '12. Raleigh, North Carolina, USA: ACM, 2012 (cit. on p. 67).

[Wil14] Michelle H. Wilkerson-Jerde. "Construction, categorization, and consensus: student generated computational artifacts as a context for disciplinary reflection". In: *Educational Technology Research and Development* (2014) (cit. on p. 26).

[Win06] Jeannette M. Wing. "Computational thinking". In: *Communications of the ACM* 49.3 (2006) (cit. on pp. 9, 11, 21, 76).

[Win10] Jeannette M. Wing. *Computational thinking: What and Why?* 2010 (cit. on p. 10).

[Win11] Jeannette M. Wing. "Computational thinking". In: *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2011 (cit. on p. 53).

[Woh+00] Claes Wohlin, Per Runeson, Martin Höst, et al. *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000 (cit. on pp. 27, 48, 76).

[WW11a] Daniel Wigdor and Dennis Wixon. "No Touch Left Behind". In: *The Spatial NUI*. Elsevier, 2011 (cit. on p. 45).

[WW11b] Daniel Wigdor and Dennis Wixon. "The Spatial NUI". In: *Brave NUI World*. Elsevier, 2011 (cit. on p. 20).

[WWL14] Wang, Danli, Wang, Tingting, and Liu, Zhen. "A tangible programming tool for children to cultivate computational thinking". In: *The Scientific World Journal* 2014.3 (Jan. 2014) (cit. on pp. 21, 67, 69, 70, 77, 86, 87).

[WZW11] Danli Wang, Cheng Zhang, and Hongan Wang. "T-Maze: A Tangible Programming Tool for Children". In: *Proceedings of the 10th International Conference on Interaction Design and Children*. IDC '11. Ann Arbor, Michigan: ACM, 2011 (cit. on p. 68).

[Yad+14] Aman Yadav, Chris Mayfield, Ninger Zhou, Susanne Hambrusch, and John T. Korb. "Computational Thinking in Elementary and Secondary Teacher Education". In: *Transactions on Computing Education* 14.1 (Mar. 2014) (cit. on p. 8).

[Yad+17] Aman Yadav, Sarah Gretter, Jon Good, and Tamika McLean. “Computational Thinking in Teacher Education”. In: *Emerging Research, Practice, and Policy on Computational Thinking*. Cham: Springer, Apr. 2017 (cit. on p. 85).

[ZAR05] Oren Zuckerman, Saeed Arida, and Mitchel Resnick. “Extending Tangible Interfaces for Education: Digital Montessori-inspired Manipulatives”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’05. Portland, Oregon, USA: ACM, 2005 (cit. on p. 44).

# Webpages

[@16a] *Bureau of Labor Statistics Employment Projections*. 2016. URL: http://www.bls.gov/emp/tables.htm (visited on Sept. 27, 2018) (cit. on p. 1).

[@16b] *National Center for Education Statistics IPEDS Completions Survey*. 2016. URL: https://ncsesdata.nsf.gov/webcaspar/index.jsp?subHeader=WebCASPARHome (visited on Sept. 27, 2018) (cit. on p. 1).

[@16c] *Trends in the State of Computer Science in U.S. K-12 Schools*. 2016. URL: http://csedu.gallup.com/home.aspx (visited on Sept. 27, 2018) (cit. on p. 1).

[@18a] *Blockly*. 2018. URL: https://developers.google.com/blockly (visited on Sept. 27, 2018) (cit. on pp. 28, 46).

[@18b] *Code Club*. 2018. URL: https://codeclub.org/en/ (visited on Sept. 27, 2018) (cit. on p. 18).

[@18c] *CRobots*. 2018. URL: http://crobots.deepthought.it/ (visited on Sept. 27, 2018) (cit. on p. 17).

[@18d] *Google Cardboard*. 2018. URL: https://vr.google.com/cardboard/ (visited on Sept. 27, 2018) (cit. on p. 71).

[@18e] *Hour of Code*. 2018. URL: https://hourofcode.com (visited on Sept. 27, 2018) (cit. on pp. 2, 7, 23).

[@18f] *If-This-Then-That*. 2018. URL: http://ifttt.com (visited on Sept. 27, 2018) (cit. on p. 43).

[@18g] *Lego Mindstorms*. 2018. URL: https://www.lego.com/en-us/mindstorms (visited on Sept. 27, 2018) (cit. on p. 67).

[@18h] *National Association of Colleges and Employers*. 2018. URL: http://www.naceweb.org/job-market/compensation/the-top-paid-majors-for-the-class-of-2018/ (visited on Sept. 27, 2018) (cit. on p. 1).

[@18i] *Open Natural Interaction*. 2018. URL: https://github.com/OpenNI/OpenNI (visited on Sept. 27, 2018) (cit. on p. 59).

[@18j] *OzoBlockly*. 2018. URL: https://ozoblockly.com/ (visited on Sept. 27, 2018) (cit. on pp. 25, 28).

[@18k] *RobotWar*. 2018. URL: https://corewar.co.uk/robotwar/robotwar.txt (visited on Sept. 27, 2018) (cit. on p. 17).

[@Bar16] Lorena A. Barba. *Computational Thinking: I do not think it means what you think it means*. 2016. URL: http://lorenabarba.com/blog/computational-thinking-i-do-not-think-it-means-what-you-think-it-means/ (visited on Sept. 27, 2018) (cit. on p. 14).

[@Wol16] Stephen Wolfram. *How to Teach Computational Thinking*. 2016. URL: http://blog.stephenwolfram.com/2016/09/how-to-teach-computational-thinking/ (visited on Sept. 27, 2018) (cit. on p. 2).

# List of Acronyms

**CL** Computational Literacy

**CS** Computer Science

**CT** Computational Thinking

**EUD** End-User Development

**EUP** End-User Programming

**FiM** Froebel-inspired Manipulative

**GBL** Game-Based Learning

**GUI** Graphical User Interface

**IFTTT** IF This Then That

**IL** Informal Learning

**MiM** Montessori-inspired Manipulative

**NUI** Natural User Interface

**PbD** Programming by Demonstration

**PbI** Programming by Instruction

**SPRITS** Simple Pluggable Range Imaging Tracking Server

**STEM** Science, Technology, Engineering and Mathematics

**TAPAS** TAngible Programmable Augmented Surface

**TUIO** Tangible User Interface Object

**TUIReO** Tangible User Interface Repurposable Object

**TUI** Tangible User Interface

**VPE** Visual Programming Environment

**VPL** Visual Programming Language

**VR** Virtual Reality

# List of Figures

# List of Tables

# Colophon

This thesis was typeset with LaTeX $2_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

# Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

Parts of the work described in this thesis have been previously published in:

- Alessio Malizia and Tommaso Turchi. "Pervasive displays in the wild: Employing end user programming in adaption and re-purposing". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2015
- Tommaso Turchi, Alessio Malizia, and Alan Dix. "Fostering the adoption of Pervasive Displays in public spaces using Tangible End-User Programming". In: *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2015
- Tommaso Turchi and Alessio Malizia. "Fostering computational thinking skills with a tangible blocks programming environ-ment". In: *Visual Languages and Human-Centric Computing (VL/HCC), 2016 IEEE Symposium on*. IEEE. IEEE, 2016
- Tommaso Turchi, Alessio Malizia, and Alan Dix. "TAPAS: A tangible End-User Development tool supporting the repurposing of Pervasive Displays". In: *Journal of Visual Languages & Computing* 39 (Apr. 2017)
- Alan Dix et al. "Rich digital collaborations in a small rural community". In: *Collaboration Meets Interactive Spaces*. Springer, 2016
- Alessio Malizia, Daniela Fogli, Federico Danesi, Tommaso Turchi, and David Bell. "TAPASPlay: A game-based learning approach to foster computational thinking skills". In: *Visual Languages and Human-Centric Computing (VL/HCC), 2017 IEEE Symposium on*. IEEE. 2017

- Daniela Fogli, Federico Danesi, Alessio Malizia, Tommaso Turchi, and David Bell. "Sustaining Cultures of Participation by Fostering Computational Thinking Skills through Game-Play". In: *on CEUR in Proceedings of GHItaly* (2017)

- Tommaso Turchi and Alessio Malizia. "A Human-Centred Tangible approach to learning Computational Thinking". In: *ICST Transactions on Ambient Systems* 3.9 (2016)

- Alessio Malizia, Tommaso Turchi, and Kai A. Olsen. "Block-oriented programming with tangibles: An engaging way to learn computational thinking skills". In: *Blocks and Beyond Workshop (B&B), 2017 IEEE*. IEEE. Oct. 2017

- Alessio Malizia, Tommaso Turchi, Federico Danesi, Daniela Fogli, and David Bell. "TAPASPlay: a Tangible Game-Based Learning Approach to Foster Computational Thinking Skills". In: *IS-EUD 2017* (2017)

- Giuseppe Desolda, Alessio Malizia, and Tommaso Turchi. "A tangible-programming technology supporting end-user development of smart-environments". In: *Proceedings of the 2018 International Conference on Advanced Visual Interfaces*. ACM. 2018

*London, January 18, 2019*

Tommaso Turchi