DOCTOR OF PHILOSOPHY

Designing Engaging Learning Experiences in Programming

Martin, Christopher James

Award date:
2017

Awarding institution:
University of Dundee

Designing Engaging Learning Experiences in Programming

Christopher James Martin

PhD. 2017

Contents

List of Figures

List of Tables

Abstract

This thesis describes research into supporting the creation of engaging learning experiences with programming. A review of relevant research that could contribute to the design of engaging learning experiences informed the construction of four pieces of fieldwork. These fieldwork studies were conducted to explore the framing of learning programming in tasks that motivate and are of value to the learner. Findings resulted in the design of a set of eight Learning Dimensions. These Learning Dimensions are proposed to address three key areas: (1) design and delivery of learning task, (2) rhythm or tempo of the learning experience and (3) practicalities. The Learning Dimensions provide educators with insights to support key design decisions for the creation of engaging programming learning experiences. Finally, a web-based tool was designed to make the Learning Dimensions accessible to educators. This tool has been used to evaluate one further workshop.

This thesis consolidates several threads of research into a learner-centred approach to learning to program. The Learning Dimensions identify important areas of decision-making to be considered when designing a learning experience. They support the assertion that factors outwith the content can significantly affect success in programming. The complex interplay between different skills associated with computer programming will remain a challenge to learners. When placed in a rich context that fits the learner well and supports the learning aims, many of these difficulties can be overcome. The Learning Dimensions draw together positive features of a learning experience that are key to ensuring learners have the best possible opportunity to engage with and succeed with computer programming.

Chapter 1: Introduction

Computer programming underpins much of the modern world: communication, transport, financial markets and many more aspects of modern life. From being electro-mechanical calculators, computers have matured to tiny ubiquitous powerful devices (Computer History, 2016). Learning to program has become an important aspect of education. One example of this is the BBC's Make It Digital campaign (Make It Digital, 2015) that seeks to inspire digital making. In the early days of computer programming, languages were esoteric and the ability to program was a niche skill. There was limited prevalence of digital technology: in 2000 in the United States, only 51% of homes had a personal computer in them; this had risen to 78.5% by 2013 (US census). What was present was far from the public consciousness. Arguably, the first PC was the Xerox PARC Alto developed in 1974 but it was not until much later that computers began to find their way into the office and home. This has swung in the opposite direction, with consumer digital technology and services prevalent (Ofcom, 2015), and the demand for people with the skills to create for these platforms is growing (e.g. Geron, 2013).

A substantial literature going back more than 50 years has explored various aspects of learning to computer program. As a result of this growing understanding, there have been major improvements in the educational technology developed to support learners. This thesis seeks to add to that knowledge by proposing a set of Learning Dimensions that crystallise important factors in the design of programming learning experiences. The Learning Dimensions are informed through four complementary empirical studies and are grounded in the literature. Finally, the Learning Dimensions were embedded in a web application that was used to reflect on an additional workshop. The software product of this thesis is that tool, which can assist educators to design engaging learning experiences in programming.

Difficult Route from Novice to Expert

It is generally accepted that it can take ten years to make the transition from novice to expert programmer (Winslow, 1996). It is therefore likely that by the end of a four year undergraduate degree, students are only part of the way to becoming an expert. There is also evidence that after two years of learning to program, most novice programmers are still struggling to be proficient (Kurland et al., 1986). Computer Science Level one (CS1) courses are often subject to unusually high dropout rates (Bennedsen and Caspersen, 2007). There is no shortage of evidence to support the belief that programming is hard. Learning to program is therefore not only a long undertaking but also one that in many cases is not completed.

Competence in programming is competence in a range of tightly interrelated skills (Kurland et al., 1986). As a result, learning to program can be problematic if these different skills are acquired at different rates for different learners. In addition, many other factors will influence the success or failure of learning to program, such as the language, the environment, the concepts being learned and the approach to teaching. This is a mature field of research and there is a good understanding of the challenges the novice faces.

Much of the research examining support for novice programmers is centred on designing a language and/or interactive programming environment that will support the learners as they face various challenges. For example, Scratch (Maloney et al., 2010) and other visual programming languages use visual building blocks to remove the need to type syntactically correct program expressions. Creating supportive IDEs is an important step in enabling learners, as it is unlikely that commercial development tools will work well for the novice programmer (Kölling, 1999a). The research reported in this thesis takes an alternative and complementary approach to supporting learners via tool-development, by considering the context in which learning takes place and the value of learner-centred motivation.

Main Research Area

This thesis seeks to explore if and how careful design of a learning experience can foster motivation and rich engagement from learners. In contrast to focusing on instructional techniques or specific programming concepts in isolation, this research takes a learner-centred perspective. Rather than stripping out challenges, such as compiler errors in block programming, and supporting them in isolation, this thesis considers programming framed by a task of value to the learner. It is proposed that the broader context in which the learning is situated will have a significant effect on the learner's experience, their ability to engage with and then progress with the challenging array of skills that constitute computer programming.

Thesis Outline

Chapter 2 offers a summary of selected literature relevant to learning to program. The literature summarised relates to three areas: (1) the challenges of learning to program, (2) tools to support learning to program, and (3) motivating learning. Additional literature is brought in throughout the thesis where it is required and most relevant, such as in the study designs and in Chapters 8, 9 and 10 to support parts I-III of Learning Dimensions.

Chapter 3 draws together the background literature and proposes the focused area for the inquiry of this thesis. The contribution identifies the importance of a holistic learner-centred approach to designing engaging learning experiences in programming, in preference to a single aspect of learning to program. Chapter 3 summarises the problem area and sets out the research questions.

Chapters 4-7 describe four empirical studies, each of which was based upon a differently designed workshop. Chapters 8, 9, and 10 introduce the Learning Dimensions and gives detail of each of the eight. Chapter 11 demonstrates how the Learning Dimensions can be used, introducing a web application to reflect upon the design of a learning experience. Finally, Chapter 12 concludes the thesis, summarising the main findings and identifying further work.

Summary

When a learner sets out to gain the skills required to become proficient in computer programming, they are embarking upon a long and challenging journey. A mature set of tools is available that have been well informed by many years of research into supporting novice programmers. Examples are Scratch (Maloney et al., 2010), Greenfoot (Kölling, 2008) and Alice (Cooper, 2010). This thesis adds to the body of knowledge via a set of design insights, made accessible in the form of Learning Dimensions for the design of engaging learning experiences. The Learning Dimensions assist educators in creating learning experiences that will engage and help learners to grow into independent autonomous learners. The focus is on supporting informed decision making by the educator, guided by the Learning Dimensions but not prescribed by them. The complexity of programming is matched in complexity by the challenge of teaching learners as individuals. The Learning Dimensions seek to offer well-grounded advice to educators in how to succeed in this.

The next chapter offers a high-level overview of this research domain.

Chapter 2: Background

Introduction

This chapter presents an overview of research relevant to learning computer programming. It comprises three parts: firstly, there is consideration of the characteristics of novice programmers and the challenges they face when learning to program (part I). Secondly, the types and range of tools available to support those learning to program are examined (part II). Finally, a review is offered of the role of motivation in learning (part III).

I. Novices and the Challenge of Learning to Program

Learning how to program has been a topic of academic research for literally decades. Soloway and Spohrer's (1989) *Studying the Novice Programmer* was a key early reference. Pane and Myers (1996) provide a detailed review and discussion of the literature pertaining to novice programmers. Robins et al. (2003) review and discuss issues relating to development of CS1 courses. All of these authors note that programming is a multi-faceted task with many interrelated skills, and there is recognition that the transition from novice to expert is non-trivial. Winslow (1996) and Dreyfus and Dreyfus (1986) describe the transition from novice to expert. The former comments on the duration, suggesting it may take ten years. The latter identify five categories in the transition from novice to expert: novice, advanced beginner, competence, proficiency and expert. The majority of studies in this field are concerned with the extremes of this spectrum, often focussing on characterising the traits of a novice programmer.

Characteristics of Novice Programmers

Anderson (1985) noted that novices take a 'line by line' approach and tackle problems in a bottom up fashion. Kessler and Anderson (1986) observed that novices lack adequate mental

models: they are limited to surface knowledge. Winslow similarly noted that novices apply general problem-solving strategies rather than context-specific strategies:

> *[An important point] is the large number of studies concluding that novice programmers know the syntax and semantics of individual statements, but they do not know how to combine these features into valid programs. Even when they know how to solve the problems by hand, they have trouble translating the hand solution into an equivalent computer program (1996, p. 17).*

Novices often struggle with the fundamental sequential operation of a program and the cumulative effect of each operation on the notional machine. This may underlie the problems faced when assembling a number of operations. Linn and Dalbey (1989) identified the inability to work at a high level of abstraction as a problem for novices. Novices tend to spend comparatively little time engaged in high-level activities such as planning. They often perform low-level local fixes rather than formulating an understanding of the whole system or related components and performing structural changes. According to du Boulay, in Soloway and Spohrer:

> *What sometimes gets forgotten is that each instruction operates in the environment created by the previous instructions (1989, p. 294).*

These findings, that novices work at a line-by-line level and struggle to consider the context in which their code sits, may be unsurprising. If we consider learning to write natural language, the grammar or syntax in isolation is relatively simple: one can expect a youngster to understand and follow the rules of capitalising and punctuating sentences, and be able reliably to write a good number of words. The challenge is in the composition of words and sentences to construct larger and more complex ideas. The same is true of programming: the difficulty is not in understanding that a `for` loop iterates for a fixed period or that a decision will branch based on a logical comparison. Difficulties arise when navigating from an understanding of simple control structures to complex compositions of control structures. Robins et al. (2003) also identified that much

research into teaching and learning of programming related to the difficulties that novices had moving from knowledge to strategy, and from code comprehension to code generation.

Knowledge rather than Strategy

Davies (1993) proposes a distinction between programming knowledge and programming strategies with an example, as follows: programming knowledge is the ability to describe how an '`if`' statement works in isolation; programming strategy is the ability to apply this knowledge correctly in a program, such as using a loop to initialise an array of `n` elements. Robins et al. (2003) describe the foundation of knowledge as an obvious necessity to programming but note that many studies of programmers are focused on the content and structure of knowledge. Likewise, many introductory textbooks and programming courses are 'knowledge-driven'.

However, Perkins et al. (1989) suggest that knowledge in novice programmers is more complex than just 'knowing'. They describe the presence of 'fragile knowledge', which is categorised as missing (learned but forgotten), inert (learned but not used) or misplaced (learned but used inappropriately). They further observe that there can be different types of novice programmer: stoppers, movers and super movers (Perkins et al., 1989). A stopper is categorised as person who is halted abruptly by an error or difficulty and does not have the inclination to tackle the problem independently. A stopper appears to have abandoned all hope of solving the problem on their own, the emotional response to being confronted with a bug or compiler error being crucial. A novice who becomes very frustrated by unforeseen problems is likely to become a 'stopper'. In contrast, a 'mover' is a learner with enthusiasm who views an error as a challenge rather than an obstacle. The ability to modify and adapt programs effectively in response to errors is likely to reinforce a mover's ability to self-support his or her problem solving and progress. Perkins describes a third category of novice as super movers or "tinkerers who are able to respond to errors but are unable to modify their program effectively and lose track of edits. This could be

regarded as ‘hacking’ and may lead to short-term fixes but will not lead to progress” (Perkins et al., 1989).

Widowski and Eyferth (1986) studied differences in strategies employed by novice and expert programmers when given code comprehension tasks. The study used two programs, one structured normally and another structured abnormally. Novice programmers were shown to approach each program with similar simple strategies, whereas experts were able to tailor existing strategies to novel circumstances. This allowed them to approach the normal and abnormal comprehension tasks differently. This would suggest that a trait of an expert programmer is the ability to apply existing knowledge in a flexible manner.

Comprehension rather than Generation

Comprehension and generation are two distinct and key aspects to programming. Brooks (1977, 1983) developed a model of programming comprehension based on the mappings between problem domain and solution domain (via intermediary domains). Brooks suggested that this is a top down process requiring knowledge of the structure of the solution and programming domain. He describes this as a hypothesis-driven operation at a conceptually high level. This type of ‘big picture’ analysis is a skill associated with expert programmers, as opposed to the fine-grained ‘line by line’ perspective often taken by the novice. The expert programmer hypothesises about a potential mapping between the solution domain and the programming domain, which is proved or disproved by finding a corresponding code fragment that implies the presence of the high-level operation, which is defined by Pane and Myers (1996) as a beacon.

Rist (1995) suggested a comprehensive model of program creation based on knowledge being represented as nodes in memory (working, episodic, semantic, and external). A node encapsulates an action or operation, varying in size from a line of code to a complete routine. Program creation comprises responding to a search cue and assembling retrieved nodes into a plan. It was noted that experts would tend to hold a variety of useful plans and are able to solve problems linearly

(initialisation, process, and output). Novice programmers would often be faced with the challenge of creating plans, which was observed to result in code generation being ordered around the central task rather than the linear approach observed in experts.

In their review of programming studies, Robins et al. (2003) observed a greater number of studies of program comprehension than studies of program generation. This may result from a greater ease of controlling the study of comprehension compared to controlling the study of program generation. It may also reflect the reality that the majority of expert programming tasks comprise component assembly, maintenance and debugging of existing systems rather than creation of new systems from first principles. Code comprehension and generation are two tightly coupled tasks. Indeed, to be a proficient programmer, expertise is required in both. However, Gilmore (1990) shows there is in fact little correlation between the ability to write code and the ability to read code, and describes these as distinct activities that each need to be taught.

Characterisations of novice programmers are often associated with research to appraise the difficulties of programming, which has been the focus of further activity over a number of years. Examples are considered in the next section.

Difficulties of Programming

du Boulay's 1986 framework locates five specific overlapping areas of difficulty experienced by novice programmers (Table 2.1). du Boulay's identification of structures as an area of difficulty matches Winslow's statement identifying 'chunks' as units:

> *They [novices] have difficulty working with algorithms that deal with a collection of operations to produce a 'chunk' of program which needs to be thought of as a single unit. In fact the transitions from intermediate solution to a syntactically correct program is also problematic (1996, p. 514).*

Table 2.1: du Boulay's Areas of Difficulty (summarised from du Boulay, 1986)

AREA of DIFFICULTY	DESCRIPTION
Orientation	describes the difficulty of understanding what programming is. This is increasingly difficult as the term 'programming' is borrowed to mean different things to different people. Examples are programming your washing machine, web programming with HTML, CSS and ever-increasing macro languages in general-purpose packages.
Notional machine	refers to the difficulty in understanding the relationship between the program and its transient states in execution. This is the cumulative effect of each line of code as it executes in sequence.
Syntax and Semantics	are the knowledge of the formal programming language required to produce syntactically correct programs that are intelligible to the proposed platform and the semantic understanding to ensure they produce the desired output and execute as planned.
Structures	are the finely grained chunks of code that achieve small goals, for example, using a loop to iterate though an array.
Pragmatics	describes the mastery of the supporting tools used to write, compile, debug and execute a program.

A further area for confusion identified by du Boulay is the misapplication of analogies resulting from design assumptions in language design. These may be assumptions the learner makes when they first encounter keywords and syntax which are also English words. Computer programs operate with a degree of precision and accuracy that is far less ambiguous than humans' communication, allowing even carefully chosen words to be misinterpreted. Examples of this given by du Boulay (1986) include people using *then* or *and* in the sense of 'what next'.

Even the simplest programs are likely to involve assignment operations to store resultant values from various operations. Assignment operators appear to be mathematical and this may result in some assumptions. The order of execution and its effect on the notional machine are key to understanding the results of an arithmetic operation. For assignment to make sense, the concept of a variable has to be understood, and analogy is often used here. According to du Boulay, in Soloway and Spohrer:

> *The most common analogy for a variable is to some kind of box or drawer with a label on it. [...] this can be confusing. In particular, it is important to stress that each box can hold only one item at a time (1989, p. 291).*

As indicated in the example, an analogy used to explain a concept may have limitations and could be open to misinterpretation. A further area for confusion with variables and assignment between them is 'after-effect'. If we draw strongly on our analogy of boxes, the notion of taking the contents of B and placing it in A would lead us to deduce that B is left empty, which is of course not the case. Describing this in Soloway and Spohrer, du Boulay's states:

> *Some novices see this as linking "A" and "B" together in some way so that whatever happens to "A" in future also happens to "B" (1989, p. 291).*

This intuition is more in line with the advanced topic of pointers. However, it illustrates the need for precise and accurate description of syntax and semantics and the potential for assumptions and confusion. This links with the concept of fragile knowledge as described by Perkins et al. (1989). A further level of difficulty for novices presently relates to the programming paradigm.

The Object Oriented Programming (OOP) paradigm has gained widespread use in industry and, as a result, it is taught on many computer science degrees. OOP was proposed to be more natural than procedural programming. Objects in the problem domain would map to objects in the solution domain. Detienne (1990) reviewed the literature researching 'naturalness' and the power of OOP. The majority of her studies relate to novice programmers and do not support claims of "naturalness". The mapping between domains and identification of objects was not obvious to the novices, who also had a need to construct models of procedural aspects of the program. Detienne's studies of expert programmers offer more support for the claimed naturalness of OOP: experts were observed to switch between OOP and procedural plans as required (Detienne, 1990). Rist (1996) suggests that OOP is not an alternative to procedural programming: it is an extension. This may contribute to the problems encountered by novices, since there is an increase of knowledge and strategies required to become a competent OO programmer.

Summary: Challenges Facing Novices Learning to Program

Many challenges facing novices learning to program have remained unchanged for many years. Advances in support tools may have reduced some difficulties and now provide a motivating context - but the essence of programming and many of its challenges persist, such as the key issue of where to start, given the numerous possible first steps that can be taken. Therefore, a task for educators in supporting novices is enabling the learners to think at various levels within a program. Supporting a novice to think and work at a scope broader than that of a single line of code is highly desirable and key to understanding the effect of localised changes on the global system (Winslow, 1996). Educators need awareness that, whilst novices may apparently be making progress, their knowledge may be fragile and/or their lack of confidence can lead to 'stopper" behaviour. Object-oriented programming may exacerbate the challenges that novices face when learning to program, rather than it being more natural to understand than procedural programming. Kölling (1999a, 1990b) outlines some of the difficulties associated with teaching of OOP, not in relation to the principles of OO but rather with complications introduced by supporting technologies: the languages and interactive development environments used in programming. Kölling presents an argument for specific tools to be used in introductory teaching and learning of OOP. The next section describes and discusses a number of types of tools that have been developed in response to the difficulties of learning to program.

II. Tools to Support Learning to Program

Various tools and approaches have been developed to support introductory computer programming (Daly, 2009). A number of such tools were discussed by a 2006 SIGCSE panel and five categories identified: visual programming tools, flow model tools, tiered tools, narrative tools and specialised output tools (Powers et al., 2006). Although a number of new tools have been added since then, the categorisation still stands as a useful tool in describing programming education tools. Visual programming, flow modelling and tiered tools all seek to alleviate

difficulties in programming in relation to syntax and flow of execution. An alternative approach is to abstract certain elements to focus on the key learning aims and objectives. Narrative and specialised output tools attempt to increase the sense of purpose and value the learner associates with programming tasks. Table 2.2 offers a description of each category and offers some examples of such tools. They can improve the motivation a learner has to engage with the task of programming. This may be a result of the increasingly screen-based world experienced by learners, who are able to interact with high-quality screen-based content. In contrast, interacting with innovative pieces of hardware perhaps provides learners with a greater novelty value. It may also be the result of the richer degree of interaction possible when dealing with physical artefacts.

Table 2.2: Introductory Programming Tools and Design Categories
(summarised from Powers et al., 2006)

CATEGORY	DESCRIPTION	EXAMPLES
Visual programming	The visual programming paradigm is used in a number of introductory programming tools to abstract above syntax and allow higher-level programming concepts to be explored. Visual programming consists of dragging and dropping program components to assemble a program. It is a general premise of visual programming that the program is always left in an executable state (Powers et al., 2006) to facilitate "tinkerability" (Resnick, 2007) or to reduce premature commitment (Green and Petre, 1996).	JPie Alice JHAVE RoboLab Scratch
Flow model	Flow modelling tools provide a visual representation of program flow, often representing the path of execution as a flow chart. Flow modelling tools allow for an external representation of the executing program that can serve as an intermediary between mental models of solution and problem domains.	RoboLab Greenfoot Scratch
Tiered	Tiered tools offer a graduated approach to programming. They offer the opportunity to perform meaningful tasks with a variable sub-set of the full language. This approach can support teaching and learning by allowing the gradual introduction of syntax and more complex programming strategies.	DrScheme RoboLab Processing
Narrative	Narrative tools use the creation of interactive games or non-interactive movies as a vehicle for programming a story. Powers et al. (2006) claim that the "ability to direct your own movie is extraordinarily attractive to a wide range of learners".	Alice JEROO
Specialised output	Specialised outputs include programs that are executed on non-conventional computing hardware, for example programming an autonomous robot which can "embody state and behaviour, physically modelling the programming solution" (Powers et al., 2006)	LOGO turtle LEGO Mindstorms Scribbler

Visual programming seeks to alleviate syntax issues by providing 'drag and drop' program creation rather than typed text. Each operation leaves the program in an executable state with immediate feedback. This may reduce the likelihood of a novice becoming a "stopper" (Perkins et al., 1989) as a result of persistent syntax errors.

Flow modelling tools provide a medium for visual externalisation of mental models of the notional machine (du Boulay, 1986). This provides a detailed representation of the program's execution path and reinforces the sequential nature of the program. In turn, this should support understanding of the cumulative effect of the program constructs, viewed step-by-step, giving access to intermediate states along the temporal dimension.

Tiered tools offer a structured and gradual increase of syntax and program complexity. This has the advantage that the one language or tool can provide a tier of sophistication appropriate to the learner and stage of learning, ranging from introductory to advanced topics.

Narrative tools seek to present programming as a creative endeavour and build on existing concepts that are familiar and engaging, as evidenced by Cooper observing enthusiasm from minority learner groups with a strong oral tradition (Powers et al., 2006). Good and Robertson (2006b) discuss the increased motivation observed in narrative game creation through "creation of a valued artefact", an idea rooted in constructionism as explored by Papert and Harel (Papert, 1980; Papert and Harel, 1991).

Specialised output tools place programming in a novel context that can improve motivation and lead to increased time on task (Powers et al., 2006). Examples include the use of autonomous robots in programming education, which has been a topic of research and practice for over 30 years, prompted by Papert's work in 1980. Moving on from autonomous robots, alternative tools such as Lego Mindstorms (Lego, 2010) have offered flexibility although most research using

Lego is focused upon building robots. Lego technology is also relatively expensive, which may limit the impact it can have. Technology that is more recent is less expensive and presents the opportunity to explore how programming education could be more tangible (e.g. Arduino, 2014). As an open source and flexible platform, Arduino enables the creation of personal physical artefacts, which can lead to additional research questions.

Specialised Output Tools

This describes the broad categorisation of different support tools, initially considering details of robots, and concluding with a final section that considers other software based solutions.

Teaching with robots as special output hardware tools is not a new idea. Papert's seminal work (1980) was aimed at teaching of school mathematics, building and programming robots. *Mindstorms: children, computers, and powerful ideas* presented a unique and fascinating prospect of a programming language (LOGO) that made computer programming accessible to young children (Papert, 1980). More than this, Papert's vision talked about and provided robust arguments for the widespread use of technology in education long before the ubiquity of computers we see today. Central to this vision was the theory of constructionist learning. Constructionist teaching theory asserts that learning can happen with greatest effect when the learner is engaged in tasks that are tangible and have real-world importance (Papert and Harel, 1991). Papert provided a concrete example of constructionism by teaching mathematics via LOGO being used to program a 'floor turtle'.

The turtle (Figure 2.1) is a simple differential drive robot with two motors. Adaptations are possible if sensors [S1 ... Sn] are added to the two motors [M1 M2] (Figure 2.2). LOGO robots were able to turn accurately to a specific number of degrees and move for exact distances. This enabled the learner to communicate via LOGO the actions the turtle needed to perform to draw geometric forms. Programming concepts such as functions were explored through the metaphor of teaching the turtle a new word. Triangle becomes a word or hook to the collection of actions

required to enact a triangle. This is incredibly powerful and was widely adopted (McNerney, 2004). As graphical user interfaces improved, economical screen-based representations of the turtle became available that gradually superseded the 'floor turtle'. Indeed modern introductory programming tools such as Greenfoot (Kölling, 2008), a Java animation environment developed to support teaching and learning of programming has a turtle sample program among many others.

Figure 2.1: Original LOGO Turtle

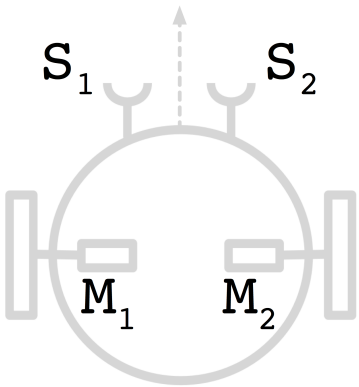


Figure 2.2: Differential Drive Schematic

Learning with LEGO

LEGO Mindstorms (2010) presented an opportunity for enthusiasts and educators, since it pooled micro-controllers and programming with a very adaptable construction kit that already had huge

market penetration. The current generation of LEGO Mindstorms, NXG, also has a huge community to support its use. There are globally organised events such as RoboCup Junior (RCJ), which invite teams of schoolchildren to compete in three subject areas using robots almost exclusively developed using LEGO Mindstorms. The three subjects are: (i) Robot Rescue: autonomous robots race to identify 'victims' in a line-following challenge that has obstacles to navigate; (ii) Robot Dance: autonomous robots perform to music in a competition judged for creativity; (iii) Robot football: autonomous robots compete in a two-a-side football match on a 90x150 centimetres grey-scale pitch.

Petre and Price (2004) present empirical evidence for 'back door learning' taking place whilst young children compete in an RCJ event. Petre and Price conducted interviews with teams taking part in an RCJ event and followed this up by conducting a detailed case study with one team. Two topics that arose from their inductive analysis are motivation and evidence of learning. One frequently reported reason for being motivated was the 'openness' of the task. Although each event culminates in a competition that identifies the best robots, there is no final definitive end point. Each year, teams can strive to improve on their efforts; education is held above competitiveness at the RCJ events. A participant in the Petre and Price study (2004, p. 151) observed:

"you can always improve it and you never have it perfect".

A number of participants also identified placing the task in a social context as a factor contributing to motivation. The ability to share ideas and the pride associated with demonstrating expertise was also reported to be important. Another participant in the Petre and Price study (2004, p. 151) commented:

"It's interesting meeting new people and showing how good you can be"

Participants were 'pulling' programming concepts rather than these being 'pushed' by mentors. They were arriving at a situation where their creative experience had led to them discovering the

need for some kind of program concepts, as described in the extract of the Petre and Price participant's transcript:

> *"What I really need is a place to put things, you know, stuff I want to remember. I don't care where it is, just so I can get it back when I want it. I could just call it back and it would tell me what's inside. Like calling a dog: 'Here Fred' and the stuff comes back." Participant (2004, p. 155).*

Petre and Price go on to offer further insights into the experience of the sometimes very young (8-9 years old) children engaging and learning with robots. A key theme uncovered was the importance of the relevance for the children – the targets were personal.

Learning with Personal Robots

> *The Institute for Personal Robots in Education (IPRE) applies and evaluates robots as a context for computer science education. IPRE is a joint effort between Georgia Tech and Bryn Mawr College sponsored by Microsoft Research (IPRE, 2010)*

IPRE builds on many years of teaching and learning using robots in computer science and artificial intelligence. An off-the-shelf robot (Scribbler) was selected and augmented with a bespoke IPRE FLUKE board that provides additional sensors. This emphasises the computing tasks rather than fabrication of a robot, as well as providing a scalable solution: it has been used with over 1000 learners. The IPRE project has developed a whole course based on Scribbler and FLUKE, resulting in a considerable array of supporting material, including course text and web resources. Blank and Kumar (2010) synthesised a number of key observations from the robots in education literature and their experiences to underpin IPRE (Table 2.3).

These motivations represent a substantial contribution of the IPRE project, having been refined and evidenced as beneficial via the IPRE CS1 module. It is interesting to note the importance of treating programming as a creative experience and that the robot should be peripheral compared to the programming experience. Refocusing on the human elements of the learning experience

rather than letting the technology drive the design has resulted in an accessible introduction to computer programming and robotics.

Table 2.3: IPRE Objectives

(1) Let the needs of the curriculum drive the design of the robot
(2) Use tools that are easy to use, scale with experience
(3) Treat the robot as a peripheral
(4) Create an accessible, engaging environment for a new, diverse population of learners
(5) Computer science is not just programming
(6) Make computing a social activity
(7) Make computing a medium for creativity
(8) Performances versus competitions

As hardware and software has improved, it has become possible for educators to work with more sophisticated and inexpensive robots. Many educators have experimented with using robots as vehicles to learn abstract skills (e.g. Kay, 2010; Klassner, 2002; Kumar and Meeden, 1998; McWhorter and O'Connor, 2009; Petre and Price, 2004; Blank and Kumar, 2010; Korchnoy and Verner, 2010). Major et al. (2012) offer a systematic review of the literature of robots in computer science education (CSE). Despite Lego continuing to develop and extend its suite of educational tools, however, one of the areas of greater growth and maturity has been in the development of *software-based tools*.

Software-based Tools

The emphasis on programming as a social, creative and demonstrable set of skills marks an important move away from more traditional knowledge-driven approaches to computer programming education, evidently with benefits to the learners (Blank and Kumar, 2010). Software-based tools permit this emphasis to be taken even further.

An overview of supporting tools compiled by Kelleher and Pausch (2005) resulted in their taxonomy of 86 different programming languages and environments specifically designed to accommodate the needs of the novice programmer. Good (2011) offers a review of five of the most popular programming learning tools, describing how programming learning environments have 'come of age' and offer good support for student-led activities. The five tools discussed by Good are Scratch (Maloney et al., 2010), Alice (Cooper, 2010), Looking Glass (Gross and Kelleher, 2010), Greenfoot (Kölling, 2010) and Flip (Good et al., 2010). Each uses a different approach to supporting learning, being borne out of somewhat different motives. Nevertheless, these five tools share a number of common elements. It used to be that significant effort had to be expended to get a computer program to do something interesting. All the tools discussed by Good offer a very rapid pay-off: from little investment, learners can produce interesting outputs. This in turn supports a tight feedback loop where learners are able to make small changes and observe their effect immediately. All tools reviewed by Good (2010) result in learners creating a valued artefact that can readily be shared among peers.

Summary: Support for Learning to Program

Powers et al.'s (2006) high-level categorisations offer a useful vocabulary when discussing tools to support learning to program. In reality, a small set of neat categories may be insufficient to describe the various tools in common use, as indicated by the fact that example tools can be tagged with multiple categories. Good (2011) argues against the drawing of a distinction between visual versus textual languages, recognising that the modern offerings are more than just languages: they are end-to-end development environments, often with a hybrid mix of visual and textual elements.

It is important to consider all tools and categories in the context of the circumstances in which they are employed. It is inadequate to treat programming difficulties solely as technical issues: tools can be used to help solve technical problems, but other contextual matters will have a substantial influence on the success and progress of a novice. For example, the amount of

supporting materials, integration into curriculum, availability and accessibility of the tools, online resources and personal support each can have substantial effects on the teaching and learning benefit. In the Institute for Personal Robots in Education (IPRE) project (Blank and Kumar 2010), an entire course and textbook was developed that addressed these non-technical issues.

Borne out of a greater understanding of the difficulties faced by learners, the set of tools on offer has been improved and now provides a good degree of support. This is a maturing field and common themes emerge across different tools pitched at different levels, such as sharing work with communities. Learners need a quick investment and reward cycle; they need to be working towards an output or product that is relevant to them. We live in a richly connected society and learning can benefit from this. As we can share pictures and stories of what we did at the weekend, many educational programming tools are enabling learners to contribute to vibrant online communities of learners (Brennan et al. 2010). Learners can be inspired and informed by the work of others and in equal measure provide the inspiration and support for those who follow them. There are significant motivational affordances to be found in sharing work and observing it being valued by others. One of the most important insights from Good's state-of-the-art paper is that five different tools had similar pedagogic qualities, despite being constructed with distinctive motives. They offer (i) rapid pay-off that supports a tight feedback loop and (ii) the ability to make a thing the learner values. The role of motivation for learning is the focus of the next section.

III. Motivating Learning

The extent to which a learner can be considered motivated to engage in a learning activity is important and, in many ways, this thesis is about creating a context for motivation and rich engagement with learning to program. Motivation is defined in the Oxford English Dictionary as "a reason or reasons for acting in a particular way" (Simpson and Weiner, 1989). If it changes desire into will and subsequent action, then it can be an important element to cultivate in the

learner when designing an engaging learning experience. This section begins with a brief description of motivation literature, including work highlighted in detailed literature reviews on motivation and learning (Deci et al., 1999; Condry, 1977).

Incentives to Learn

Skinner (1953) described the ability to modify behaviour through reinforcement in animal experiments. It was shown that if a reward was given in response to a desired behaviour, this desired behaviour by the animal was likely to recur. This reward is referred to as a task-extrinsic reward and its effect can be described as extrinsic motivation. Task-extrinsic motivation involves some outside reward that does not directly relate to the intended task but that instead satisfies an innate drive in the participant, such as hunger or anxiety reduction. An alternative position was presented by White (1959). White recounts a number of studies that demonstrate animals exhibiting behaviours that cannot be described as a desire to satisfy a primary drive such as hunger or sex. In experiments, animals demonstrated a desire to have space to explore that did not fit within the existing model of an innate drive. This behaviour was not purely functional. Behaviour of this type is called intrinsic motivation. Intrinsic motivation is 'motivation from within' or to engage in an activity for its own sake without directly perceivable payoff to the participant.

Amabile et al. (1994) offer a good description of elements of intrinsic motivation and task-extrinsic rewards. Elements of intrinsic motivation include self-determination, competence, task involvement, curiosity, enjoyment and interest. All are desirable behaviours for a learner engaged in an activity. Amabile et al. (1994) describe task-extrinsic motivation that may take any of the following forms: competition, evaluation, recognition and tangible incentives. Each of these has positive or negative connotations and so it is important to consider them in context. The literature exploring the effect of task-extrinsic rewards on intrinsic motivation and performance is complex and spans a wide range of learners and situations. To compound this difficulty, participants have been observed in contrasting and nuanced ways; this will be explored next.

Bahrick et al. (1952) explored the effect of financial incentives on both central and peripheral learning tasks. An experimental group showed an increased performance for the central task when offered an additional financial incentive, where performance was measured as time to complete the task. However, no improvement was observed with the peripheral task, suggesting that a financial reward narrows the focus of the participant and reduces the chances of incidental learning.

Anderson et al. (1976) also observed the effects of money, awards and verbal feedback on intrinsic motivation in kindergarten children engaged in a drawing activity. The participants had displayed a high degree of prior interest in the drawing task. Learners were offered positive feedback (an award or money in the experiment situations). Three different control situations were offered: no treatment, benign interaction with an experimenter (who would answer questions but not offer approval) or a final control situation with an experimenter present who would not interact with the learners. Only the verbal feedback had a positive impact on the learner's motivation and time on task, with all control situations decreasing subsequent levels of intrinsic motivation. Glucksberg (1962) also explored the difference a financial incentive made with respect to problem complexity. In a study with 128 participants, it was demonstrated that while a financial incentive improved the performance of participants engaged in a simple problem-solving task, it had a detrimental effect upon a more challenging problem.

Deci et al. (1999) found evidence that positive feedback can enhance intrinsic motivation: this was observed more strongly in younger participants than in college learners. They differentiated between tangible and non-tangible task-extrinsic rewards, finding that tangible rewards did undermine intrinsic motivation and were in some cases seen as methods of control that may forestall self-regulation.

From a review of 43 studies, Eisenberger et al. (1999) conclude that extrinsic rewards undermine intrinsic motivation where the reason for their delivery is poorly defined in the eyes of the

recipient. Extrinsic rewards have also been described as increasing speed and quantity of effort at a reduced quality (Condry, 1977).

In a learning situation in particular, Cordova and Lepper (1996) robustly locate the importance of personalisation and choice for increasing intrinsic motivation in learners. Their study engaged 72 fifth grade learners (10 to 11 year olds) who were randomly assigned to one of five groups, in a 2x2 factorial design with one control, with the first factor being personalisation and the other factor being choice. They found that there was a powerful learning benefit observed in the personalised choice condition. Learners were observed to have not only increased motivation but also displayed a deeper engagement in the task.

Extrinsic motivators are commonplace in education at all levels. Competency is typically measured with a summative assessment that results in a grade for the learner. This is an important aspect of assessment in education, though this risks fostering the type of motivation that encourages goal-oriented strategies (Condry, 1977) in learners. In addition to encouraging greater throughput of work at a reduced quality, extrinsic rewards increase focus at the expense of engagement in peripheral topics. In a learning situation, this is likely to foster shallow learning where the learner focuses only on activities that are clearly related to the examination at the end of the learning.

In conclusion, intrinsic motivation is inarguably a desirable feature to encourage in any learning experience, with evidenced learning benefits described above. The role of extrinsic motivators is perhaps a little more uncertain. Tangible rewards have been shown to narrow thinking and reduce the depth at which learners engage. However, positive feedback has been observed to enhance intrinsic motivation. Understanding extrinsic and intrinsic incentives, and how they relate to the instructional paradigms, may give pointers to the design of engaging learning experiences.

Learner-centred Education

Betoret and Artiga (2004) provide a useful visual model of four classical instructional paradigms: product-centred, learner-centred, process-centred and teacher-centred (Figure 2.3). Product-centred focuses on the output created as a result of the learning, as opposed to process-centred, which would place importance on the process used to create the product. For example, the former might be a correct answer to a mathematics question whilst the latter might be the working used to obtain the correct answer. Teacher-centred is characterised by a knowledge transmission approach to teaching rather than students constructing knowledge by gathering and synthesising information. In a teacher-centred setting, assessment is used to monitor learning in contrast to a learner-centred approach in which assessment is used to consolidate and promote learning.

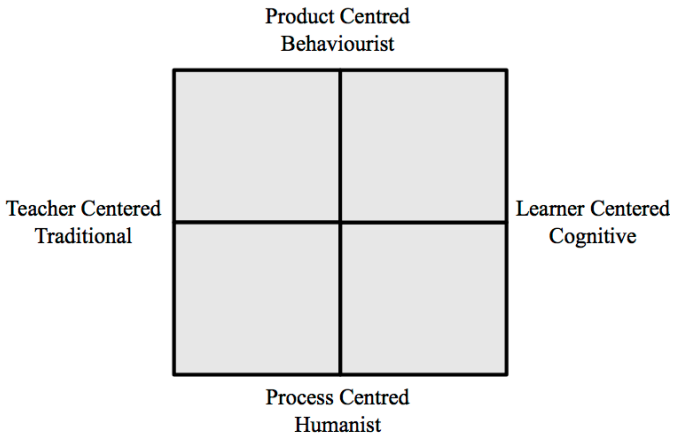


Figure 2.3: Model of Instructional Paradigms (from Betoret and Artiga, 2004)

As a high-level model, this can be valuable when considering (i) the role of the educator and (ii) the type of educational experience that is being designed. What is considered a traditional instructional approach, with knowledge transfer being followed by summative assessment, would be located in the upper left quadrant, and is associated with extrinsic motivators. Progressive approaches to learning would sit in the lower right quadrant in which the experience is focused on the process and the learner. Cordova and Lepper's work (1996) suggests that this type of a

process- and learner-centred learning experience is intrinsically motivating. Three different systems to encourage a learner-centred programming experience will be considered next.

Contributing Student Pedagogy

A crucial aspect of a successful learning experience is having engaged learners. Hamer et al.'s (2008) Contributing Student Pedagogy (CSP) aims to achieve this by enabling learners to have a prominent role in their learning experiences. In CSP, learners co-create knowledge and understanding. This pedagogy moves beyond variations of peer-assisted learning (e.g. Topping and Ehly, 1998) and empowers learners to participate actively in all aspects of the learning.

In this situation, the educator takes the role of a guide rather than that of a gatekeeper or provider of information. One advantage argued by its proponents is that it mirrors informal workplace learning, in which it is common for co-workers to support each other in learning.

CSP activities can be depicted by means of a star diagram (Figure 2.4). Its five axes represent different elements of a learning experience. Drawing a bounding shape around the axes results in a visual representation of the CSP activities taking place. The greater the distance from the centre, the greater the amount of control a learner has over that element of the learning experience.

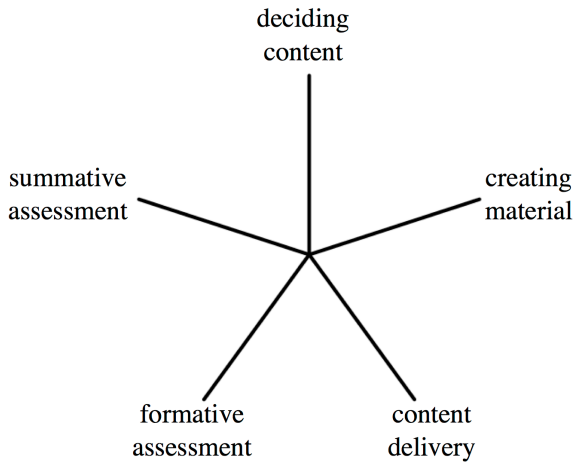


Figure 2.4: Model of Activities (Hamer et al, 2008)

Figure 2.5 gives one example of a star diagram, in this case representing a peer-review activity (Hamer et al, 2008).

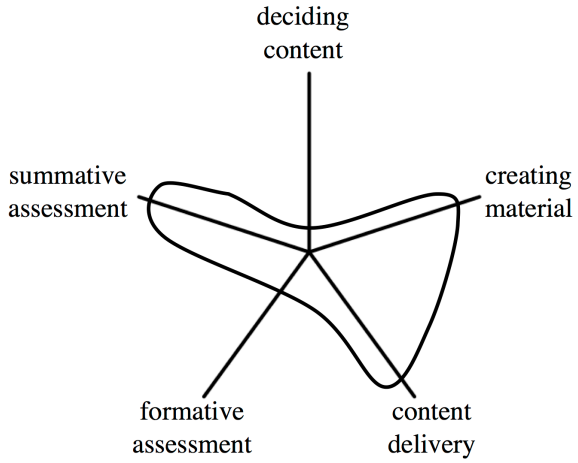


Figure 2.5: Peer Review Activity (Hamer et al, 2008)

As shown in Figure 2.5, the star diagram provides a quick visual guide to the extent to which learners are given control over their learning. As a concept, CSP has attracted some attention in the computing science education community with over 60 citations (by 2015) since the original publication in 2008. However, the majority of the publications are short conference papers that use CSP as a catchall for learner-centred learning practices. None of these publications made use of the star diagram to describe how CSP had been applied. Regrettably, there is little evidence to date of this method being rigorously investigated and built upon.

Interest Driven Learning

The Interest Driven Learning Design Framework (IDLDF) (Edelson and Joseph, 2004) offers guidelines for the creation of learning activities that utilise learner interest as a motivator for learning. The case of IDLDF is underpinned by educational psychology research that identifies four benefits of interest to motivate learning:

- natural appeal: interest encompasses a natural desire to be involved in certain activities. This was described by Renninger (2000), who argues that learning is not deliberate or conscious in situations where learners are interested in the material they are learning.

- master goal orientation: Schiefele (1991) asserts that interest leads to master goal orientation. Where interest is present, learners will strive to master the knowledge and understanding rather than just demonstrate it to a degree demanded by the educator.

- persistence and effort: Hidi (1990) performed a review of educator literature pertaining to the effect of interest. A number of studies were identified that indicated learning interest has a beneficial effect on attention, engagement and performance.

- richly and strongly connected knowledge: Renninger (2000) asserts that there are significant cognitive benefits for learners when new learning tasks are located within an area of learner interest. In areas of learner interest, existing knowledge supports the formation of new knowledge and leads to highly connected knowledge.

With the case for learner interest made, Edelson and Joseph present the Interest Driven Learning Design Framework to address two main challenges, which they label as coverage and strength. Firstly, it is quite likely that many of the learning objectives being addressed in the material being taught (called coverage) are not directly of interest to learners. This is addressed by establishing relevance and highlighting the usefulness the taught material. Secondly, the degree of the interest (called strength) may be insufficient to drive a learner to engage fully with activities. This is addressed by varying the types of motivation to supplement interest. Usefulness is defined in the IDLDF as the authentic use of skills and knowledge that goes beyond the educational setting. This discounts usefulness in terms of attaining extrinsic rewards, such as grades. There needs to be genuine value in the skills and knowledge beyond the learning experience. Pleasure, concern,

identity formation, life goals and curiosity are identified as key sources for demonstrating usefulness and fostering motivation in the learner.

The IDLDF is proposed as a four-phase process: (1) determine interest in the learner, (2) align learning objectives and learner interest, (3) where necessary use context to initiate motivation, and (4) where necessary use context to maintain motivation (Edelson and Joseph, 2004). The IDLDF is robustly located in the literature and has an intuitive feel that likely resonates with educators, who will recognise that learning benefits can be obtained by aligning taught materials to the interests of the learner. IDLDF is presented in a content-independent manner, allowing it to be applied to a wide range of subjects and educational situations. Despite accumulating over 40 citations by 2015, the IDLDF remains a theoretically underpinned but ultimately un-evidenced framework, as the various research citing the work does not make the important next step to evaluate its use in the classroom.

CARSS (Context, Activities, Roles, Stakeholders, Skills)

CARSS is a framework for learner-centred design of educational software (Good and Robertson, 2006a). It offers a comprehensive list of issues associated with the design and development of educational technologies, identifying five important areas:

(i) Context: context considers a range of constraints brought about from the situation in which the tool is to be used. These may be practicalities such as the physical environment, legal or ethical concerns or institutional constraints such as timetabling.

(ii) Roles: roles define the different roles required throughout the design process, such as design partner, project manager, technology specialist, researcher, subject-matter expert, child development experts, learning scientist and collaboration facilitator. CARSS draws a distinction between roles and stakeholders, with roles describing a function that may be fulfilled by a member

of the design team. It is possible that a role may be transient within the design team and not tied to an individual.

(iii) Stakeholders: stakeholders represent the various groups with a vested interest in the project. CARSS seeks to define this broadly including teachers, pupils, parents, industrial partners and funders.

(iv) Activities: activities describe the sequence of actions performed by the design team or any other of the stakeholders at various stages. The approach taken is one of rapid prototyping with many opportunities to engage stakeholders and enable them to inform the design process. These activities include requirements gathering, design, prototype development and evaluation.

(v) Skills: three key intellectual skills are identified as required in members of the design team. Firstly, synthetic skills are required to see problems in a new way and propose solutions. Secondly, analytic skills are needed to recognise potentially successful ideas. Thirdly, in a practical contextual, the ability is needed to describe to others the value of your idea (Sternberg, 2003). Skills and attitudes are not limited to the adult members of the design team. Good and Robertson note that it can be very challenging for children to engage in learner-centred design activities, since such activities are far removed from the typical classroom experience children will have had. The most important skill for children is the ability to engage in discussions and to possess a willingness to acquire skills required for the project.

This CARSS framework acknowledges that a fully participatory design approach may not be feasible in all circumstances. Nonetheless, including learners and other stakeholders in the design process as much as is possible has been demonstrated successfully: CARSS has been applied at both the scale of a small research project and a large national project (Good and Robertson, 2006a). The themes it identifies as important arguably could be used to inform the design of a smaller-scale learning resource or learning experience.

Summary: Learner-motivated Programming

The overreaching theme that emerges from CSP, IDLDF and CARSS is that learning is most fruitful in credible engaging learning experiences that place the learner at the centre. Each approach proposes important aspects of how to achieve this. In CSP, engagement is achieved by encouraging the learner to actively contribute and have control over the creation and execution of learning. In IDLDF, engagement is achieved by relating content to existing areas of learner interest. The mechanism of establishing interest through genuine usefulness is an important concept in fostering deep learning (Entwistle, 1991) and goal mastery orientation as opposed to goal performance orienting. CARSS underscores the importance of stakeholders in the design process and the various roles they fill. In addition, CARSS explicitly identifies a range of contextual constraints that must be navigated if a design is going to be successfully implemented.

Various approaches, models and frameworks support the creation of learning experiences and tools to support learners. Using the model of different instructional paradigms (Figure 2.3, learner-centred, teacher-centred, process-centred and product-centred), one can reflect on some fundamental components of a learning experience. This inevitably highlights some tensions; it is important to ensure learners have a rich engaging experience that can be considered progressive, process-centred and learner-centred. At the same time, however, there is an argument for some more traditional qualities such as standardised assessment.

Conclusion

The difficulties of learning to program have been studied for nearly 50 years and many challenges identified years ago endure to this day. The essence of programming remains unchanged. It requires a programmer to take a problem and describe it in sufficient detail, without ambiguity, such that a dumb machine can interpret the instructions. What has moved forward considerably is the set of tools used to support learning to program. Just as commercial development tools have matured from rudimentary text editors to powerful interactive development environments, so too

have educational tools, which have benefited from years of research and from the increased capacities of modern computers. Desirable features for education programming tools and learning experiences are increasingly being recognised as those relating to the motivation of the learner, such as personal, social and contextual elements rather than purely technical elements. Examples include the capacity to tap into and contribute to a community of like-minded learners, and the ability rapidly to make a thing that the learner values. This is perhaps where the true power of modern tools originates. They enable learners to make things that they care about and that their peers value and find interesting. This taps into a powerful intrinsic motivation that drives learners to overcome unyielding difficulties of computer programing.

The next chapter will reflect upon this background literature – novices, challenges, tools and motivations – and introduce the statement of the problem that is addressed by this thesis.

Chapter 3: Statement of the Problem

Drawing Together the Background

Supporting those that undertake the task of learning computer programming is a complex challenge. The literature spans decades and is scattered across many fields. A brief summary has been drawn together in Chapter 2. Many of the difficulties from years ago persist in modern languages. The ability to use correct syntax and trace the execution of control structures is an irremovable part of programming. However, the tools to support learners have been designed with a mature understanding of the problems they need to address. Tool designers have taken steps to simplify programming environments so that learners quickly can make things that they value. Alongside and in many cases independent of this, various pedagogic approaches and design frameworks have been proposed to enable the effective support of learners as they tackle programming

In addition to challenges, tools and pedagogies, a key ingredient to effective learning is motivation Being able to share these artefacts has also emerged as important to grow communities of learners that can inspire and support one another. The aspiration for an excellent learning experience has to include learners that are self-driven by intrinsic motivation, and have a curiosity to master skills and overcome difficulties. This requires a genuine sense of purpose and the flexibility to give learners ownership over what they are working on.

The Problem

One matter identified by the disparate nature of the background section is the fact that the valuable insights derive from many different fields of research. This thesis seeks to bring together insights from a range of them. In addition, a significant challenge is progressing research findings to a point where they can be easily adopted by educators. Many pockets of insight could be of serious value to teachers who are creating programming learning experiences. The problem that this thesis

seeks to address is how to select, combine and operationalise a set of crucial characteristics and affordances that can inform design decisions.

This collection of insights can be used as a vocabulary to aid in the discussion, critical reflection and design of programming learning experiences. The aim is not to create a set of abstract theoretical principles of learning. In contrast, the desired contribution is a set of insights to assist educators in making informed decisions about how they design learning experiences. These must be accessible to a broad community of educators for the benefit of an equally broad community of learners.

The approach taken to fill this gap has been to conduct a set of field studies that utilise a combination of qualitative and quantitative research methods. Ecological validity for the findings is achieved by conducting the research across a range of real-world learning situations. Ethical considerations have been identified and clearance to conduct the research was obtained from the ethics committee of the School of Computing, University of Dundee. The governing principle of the ethics framework at Dundee is to ensure participants are treated with respect: participants should understand what they are taking part in, that the proposed research methods are reasonable and do not result in any unnecessary discomfort for participants.

The studies are presented in chronological order, and rest upon and inform each other. Aspects of the literature are included in the studies and scrutinised in the context of the empirical work. Phenomena that emerge as a result of the design, delivery and evaluation of the studies, if not previously discussed in the literature, are also highlighted. A brief summary of the studies and the research question they address now follows.

Research Questions

Study I, Robot Dance, follows on from the background section on supporting tools. It was conducted to explore the efficacy of using robots to support an early introduction to programming.

By taking pre- and post-event measures from over one hundred schoolchildren, this study addresses the research question:

(Q1) How does use of a physical robot in an activity support learning of introductory programming?

Study II, Robot Dance in the Community, builds upon the findings of Robot Dance. It was designed to explore a similar programming challenge but delivered with less structure than in the classroom setting. The findings of Robot Dance were limited by the questionnaire-driven approach. Robot Dance in the Community addressed this by using participant observation to address the following research question:

(Q2) Given freedom in a programming activity, how do learners organise themselves?

Study III, Whack-a-mole, sought to explore an alternative physical artefact for learners to work with and how video could support different paces of learning. The study used a combination of questionnaires and participant observation to address a research question that had resulted from the first two studies:

(Q3) How does working with a physical artefact as opposed to a screen-based artefact affect learning of computer programming?

Study IV, Digital Makers, built upon all of the previous studies. It attempted to create and evaluate a highly engaging and effective learning experience in programming. In addition to lessons learned from previous studies, design decisions were made to give learners a higher degree of control over what they were making to support ownership, personalisation and purpose. A combination of questionnaires and participant observation informed the research question:

(Q4) How do personalisation, ownership and purpose in an activity affect introductory programming learning?

Insights generated from these research questions are synthesised as a set of 'Learning Dimensions'. The Learning Dimensions follow the style that Green and Petre (1996) proposed in their 'Cognitive Dimensions of Notations' framework, in which they outline a common vocabulary and reference point for the design and discussion of notations. The Learning Dimensions are a first attempt at capturing and describing key areas for design and decisions relating to learning experiences. They are not intended to be used formulaically, but rather to serve as a source of inspiration and information for educators who are designing or critically evaluating a learning experience. Chapters 8, 9 and 10 introduce and describe the Learning Dimensions.

Chapter 11 describes a web-based tool designed to assist educators using the Learning Dimensions. The application was used to evaluate a new workshop, Wee Beasties. The tool offers a high-level description of each of the dimensions, allowing the educator to capture how they intend to or have applied each dimension, before drawing their notes together into a single document that can support the design or editing of a lesson plan.

Finally, in Chapter 12, conclusions are drawn and reflections are made upon the main contribution of this thesis. In addition, interesting areas for future work are outlined. Programing is difficult yet activities that remove the difficulties run the risk of distorting what programing is. In contrast, the Learning Dimensions described here can guide and support the design of engaging learning experiences. They can help educators to give ownership to learners, and cultivate the intrinsic motivation that can drive learners to master the complexity of authentic computer programming as applied to problems about which they care.

Chapter 4: Study I Robot Dance

Introduction

Robot Dance was designed as a study to explore how working with physical robots can support introductory programming learning. Since Papert's Mindstorms (1980), there have been numerous pieces of research (e.g. Kay, 2010; Klassner, 2002; Kumar and Meeden, 1998; McWhorter and O'Connor, 2009; Petre and Price, 2004; Blank and Kumar, 2010; Korchnoy and Verner, 2010) looking at teaching programming with robots with mixed results. Often, with the notable exception of Institute for Personalised Robots in Education (IPRE), these tend to be studies conducted by undergraduate educators as 'bolt on' activities to existing teaching duties. They tend to be evaluated by means of an exit questionnaire and they contribute little more than likability data from learners. Two aims of the Robot Dance study were: (1) to make objective measures of learning effect and (2) to obtain broader insights into how the learning was being supported. By addressing these aims, this study will answer the research question:

> *(Q1) How does use of a physical robot in an activity support learning of introductory programming?*

Background

The Robot Dance workshop was inspired by RoboCup Junior (RCJ, 2011), an international event open to school pupils around the world. Each annual competition places programming and engineering challenges in a different context that will motivate different participants. RoboCup Junior teams generally develop their robots using Lego Mindstorms (LEGO, 2010) at after-school clubs in preparation for the event. Petre and Price (2004) observed evidence of 'back door learning' in learners taking part in RoboCup Junior. Learners were observed to seek programming knowledge and concepts from tutors as circumstances arose where they felt there must be a better

way of doing something e.g. repetition of code block motivating a desire for subroutines or functions.

RoboCup Junior offers three challenges: robot rescue, robot football and robot dance. The dance element of this competition offers an open problem that is useful to develop content appropriate to a wide audience. RoboCup Junior has been shown to engage a wide range of participants from young learners to university lecturers. Prior knowledge is to some extent irrelevant, as the task is creative, subjective and without a pre-specified endpoint. Often there is little difference in the output from an expert or novice, there being definite scope to over-think the problem. This has been seen to have a pitch-levelling effect.

Description

Working with multiple schools across Scotland presented a number of constraints. Practical considerations had to be made to ensure all equipment was safe and could be easily deployed and removed with minimal disruption. There was a need to fit a workshop within the school day. Most importantly, the session had to be of educational value to the learners. Formal education in Scotland is tightly governed and particularly as learners progress through the system, their time is increasingly valuable. A key impact this had on the study was that any research elements had to take minimal time out of the activity.

Furthermore, teachers typically offered only a single hour-long period to an outreach activity, or occasionally a double period. To accommodate this, the workshop was designed to have a one-hour delivery, with some flexibility to make use of any extra time. A consequence of this duration constraint was that there was insufficient time during the workshop to build a robot. Therefore, prefabricated robots were constructed in advance: Arduino-based differential drive robots (Arduino, 2014) were judged to have the required qualities of compactness and robustness. This

decision had a further benefit in that all focus was on programming rather than it being partially diverted to construction[1].

Workshop materials

Robot Dance involved programming Arduino based differential drive robots. Participants were given three things to enable this experience: the Arduino IDE (Figure 4.1), an Arduino-based robot (Figure 4.2) and clickable documentation (Figure 4.3).

IDE: The Interactive Development Environment (Figure 4.1) used was the standard Arduino IDE. Arduino boards have been designed in response to the needs of artists, interaction designers and product designers. As a result, programming an Arduino in C from the Arduino Interactive Development Environment (IDE) has been designed deliberately to present a very low barrier to the textual programming of a microcontroller. The IDE is extremely simple and uncluttered in its design, which lends itself well to use with novice programmers: it is possible to construct a solution quickly. A tiered approach (Powers et al., 2006b) to introductory programming has been taken with much of the low-level programming abstracted to an Arduino library.

[1] Previous experiences using Lego Mindstorms kits have been observed to be polarising, with some learners revelling in the mechanical challenge of constructing their robot and others being disinterested.

Figure 4.1: Arduino Interactive Development Environment

There are several buttons along the top, a large white area for the code and a black section at the bottom. The predominant button used in this session was the second from the right, which is used to upload a sketch or program to the Arduino. In the workshop, a shell program was used, with a comment indicating where the learner's code should go. The black area at the bottom displayed console information, such as compile completion and compiler errors.

Robot: The robots (Figure 4.2) had a number of capabilities directly observable in the hardware. Each robot had two independently controllable motors with small wheels. These could be driven forward or backward to achieve a range of movement. Two light dependent resistors served as eyes and allowed the robot to detect light sources. These also gave the robot the appearance of a small creature and served as a useful similarity between humans and robots. On the top of the robot there were two small buttons used to start or reset the robot's program. Also on the top was a red, white and blue light emitting diode (LED). A yellow LED was used to indicate the state of the robot prior to execution of the program.

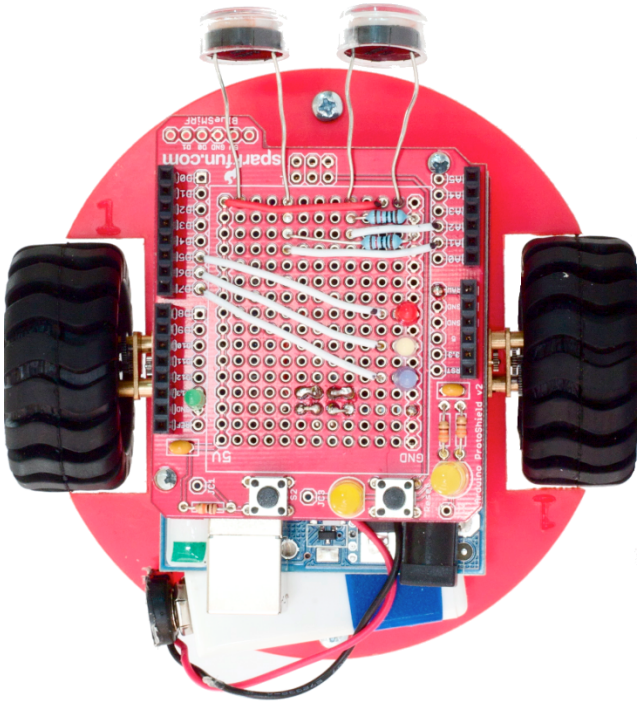


Figure 4.2: Robot

Reproducing textual syntax precisely is a common area of difficulty in introductory programming (du Boulay, 1986). Visual programming tools seek to alleviate syntax errors by removing the opportunity to produce typed errors, rather than highlighting that an error has occurred via a compiler error message. In the case of learning to program, however, this needs to be balanced against a desire to expose learners to a language and syntax they are likely to see if they pursue computing. To address this tension in Robot Dance, learners were given interactive documentation that allowed them to copy and paste functions as they constructed their programs.

Documentation: The clickable documentation (Figure 4.3) gave participants a quick reference to the basic functions or words the robot can understand. The documentation had five different tab panes that group functions related to: basic motion, complex motion, LEDs, waiting and sensing. By clicking on one of the instructions, the function can be copied to the clipboard complete with semi-colon and carriage return. It can then be pasted directly into the IDE. A further motivation to provide the clickable documentation was to encourage the advanced skill of browsing documentation to discover new things, e.g. “I know that a function can control the motors; let's see if there is one to control the lights”.

Figure 4.3: Clickable Documentation

Workshop sequence

The workshop opened with a one-slide introduction to robots. It used a popular age-appropriate film as a prompt: Disney's Wall-e (Wall-e, 2016) served as a mechanism to discuss the similarities between the Wall-e character, the Arduino robots and us. This was used to highlight the fact that humans can move, see, and make decisions. Moving was categorised as engineering and, as the robots were already constructed, the challenges that remained were computing ones: making the robot think and see, by giving a robot a sequence of instructions to make sense of its environment, visually, and take decisions to carry out some desired task.

The workshop continued with two different activities: *dance* and *follow the light*. The latter extended the former to include programming a light-following robot. *Dance* was chosen to be an open challenge, as per Robocup Junior, and to give a context that may make computer programming more appealing to female participants. There was a risk of naively fitting a task to a specific stereotype, for example believing that something pink and sparkly would appeal to all girls. The openness of a dance task was judged to be somewhat more considered and inclusive.

Dance

Once the activity was introduced, the next task was to describe the steps required to program the robot. This took three slides, introducing: (1) the capabilities of the robot, (2) the instructions or language the robot understands and (3) the IDE or 'fancy word processor' used to group together the instructions and send them to the robot. In total, this took under ten minutes and was all that preceded the **first task**: to make the robot move across the desk. It was felt that the short time to task would be key to retaining the attention of the learners. More than this, the heart of this approach to teaching and learning was that practical activity creates a fruitful learning context.

This first task introduced a key concept: instructions used to initiate motion operate as if they were light switches: `forward();` changes the state of the robot from whatever it was to both wheels moving forward. Timing was also highlighted as a means to control motion, in this case distance travelled. If the program consisted of `forward();` immediately succeeded by `stopMotion();` the robot would do nothing, in the same way that if you switch a light switch on and off really quickly, you can barely notice the light go on. What was required was a pause before the next change of state; this pause affected how long an action or dance move lasted. Thus to move 50 centimetres, the program had to change the state of the motor to forward, wait for however many seconds and then change the state of the motors to stationary. This enabled learners to explore the examples described and removed the likelihood of the learner's knowledge being inert (Perkins and Martin, 1986), which is knowledge that is present but has not been enacted or applied by the learner.

The **second task** for learners built on the first to add an additional action. The robot had to move across the desk, spin 180 degrees and return to the original position. This reinforced the concepts from the initial task, and extended distance controlled with time to degrees of rotation controlled by time between the spin command and the next state change. The term `forward` was also given a new relative perspective, as the same instructions to move the robot away from the start point would move it back to its original position. The `forward` instruction would be affected by the orientation of the robot. This began to give a concrete example of the fact that "...each instruction operates in the environment created by the previous instruction" (du Boulay, 1986). The effect of a single instruction on the position of the robot was the cumulative effect of all previous instructions in the program.

Two types of waiting instruction were used: `waitForStartButton();` caused the program and thus the robot to pause execution until the start button was pressed, `waitForLight();` caused the robot to pause until a light brighter than the ambient light was detected by the light sensors. This was used to get the robot to wait until the spotlight shone on it at the start of its

dance performance. A `wait_ms()` instruction accepted time in milliseconds. A `wait()` instruction accepted time in seconds. These were used to control the behaviour of the robot. For example, `forward()` followed by `wait(2)` would result in the robot moving forward for two seconds before moving on to the next instruction.

Learners were then given the final task: to choreograph a dance. There were two constraints: the dance had to last for exactly 20 seconds and the robot must not fall off the stage (a one square metre mat or equivalent tabletop for added suspense). Timing was straightforward and could be achieved by keeping note of the wait times used in the program. Not falling off the stage required a more heuristic approach of not moving for long times, as the robots did not offer enough precision for a more sophisticated approach. Up to this point, the robots had been started via a start button; this also had to be modified to allow the robot to start the program when the stage lights came up. An instruction for this was contained in the interactive documentation.

After the learners had been given a relatively short development time of 10-15 minutes, they uploaded their final program and gathered round the dance floor to watch each of the performances. Blank and Kumar (2010) describe the way in which a performance has wider reaching motivational effect than competition in this area. As a result, performance is included in *dance*. When delivering the workshop to larger groups that had up to ten teams, performance was applied as a series of dance-offs, with two robots performing side-by-side on two stages simultaneously.

Follow the Light

Braitenberg (1986) described how incredibly simple robots with different linkages between sensors and actuators exhibit behaviours that may be perceived as complex and human-like. The central premise of this style of robot control involves iterating the actions of (1) sense environment and (2) react by performing an action. This allows what are very small simple programs or decisions, at a local level, to generate complex behaviours at a more global scale. Papert (1980)

describes an example of this in illustrating how to draw a circle with LOGO. Mathematicians will immediately reach for complex equations that involve Cartesian coordinate systems and a global perspective; in LOGO, however, drawing a circle can be achieved very simply: move forward a little, turn a little, repeat. This principle of small simple computation producing globally complex behaviour can be captivating for learners being introduced to programming. Due to the increased complexity of this workshop, it was delivered to senior school pupils exclusively.

Follow the light used the same introduction and first two challenges as *dance*. From basic sequence and state, the concept of a variable was introduced via a blinking light. In Arduino circles, `blink` is the equivalent of `hello world`. By turning an LED on, waiting for a period, turning it off, waiting for a period and repeating, an LED would flash. Extending this, a variable could be used to store the duration of the delay, introducing the potential advantages of code readability and the ability to edit the code in a single place. This variable was then used to store the value of a potentiometer (similar to a radio volume control). This gave the ability to change the rate of flashing when the code was running. Essentially a potentiometer is a tangible object that can give the program a range of numbers (0-1023). This offered a concrete tangible representation of a possible role of a program variable.

The next skill introduced was decision: this was introduced via an almost natural language statement about instructions to maintain a constant temperature in a room: "if the room is too hot open a window or else close the window". Once the basic structure was highlighted, there was some sensed information such as "room is too hot" and if this was true, one course of action is taken; if it was false, a different course of action was taken. At this stage, the syntax of the `if` block could be mapped out. To get to the point of implementing this, it was necessary to unpack the notion of 'too hot'. This is a Boolean operation involving reading the room temperature, storing this in a variable `t` and making a comparison: is `t` greater than threshold. This formed the basis of the third task: when a bright light was shone on the robot, move forward or else stop. In other words, in a single-dimensional land, create a light-following robot. This again gave the

learners an opportunity to enact the concept of a light-following robot: by unpacking the concept of a threshold and describing it via an `if` statement in a robot, the learners were able to physically interact with the code and see the outcomes of the code they had written. The final challenge involved extending this program to a two-dimensional land. This involved establishing a connection between the two light sensors and independently controlling the motors to achieve the desired behaviour.

Overall, the key learning aim for the workshop was to offer participants an engaging programming experience using a real world language with syntax they are likely to see in future programming courses. Secondary to this was the objective of portraying the task of programming, which is often regarded as rigid and scientific, as a creative endeavour where imagination and creativity were as important as knowledge. Specifically the workshop aimed to address the following three learning objectives:

LO1 Participants should learn about flow of execution by producing a list of dance instructions and observing resulting robot movements.

LO2 Participants should learn about syntax in programming and the degree of precision required when programming as they produce their programs.

LO3 Participants in the *follow the light* sessions should gain an understanding of variables and their role in programming. This more advanced session also introduces decision and iteration through development of Braitenberg-style robots to solve mazes and follow lights (Braitenberg, 1986).

Study Design

The Robot Dance workshop had measures taken from 12 sessions, with participants from four different populations, ranging from local secondary school (two sessions), visiting youth group

(one session), university pre-application visit days (five sessions) and booked sessions at a large science festival (four sessions). The total sample comprised 132 paired pre- and post-tests with pupils of an age range of 12-16. Overall, 57 females and 75 males participated. Three of the groups received the *follow the light* intervention with the remaining nine receiving the *dance* intervention.

As the study involved interaction with human participants, ethical approval was sought from the School of Computing Ethics Committee, and granted. The committee's recommendation was that informed consent should be obtained from class teachers rather than individuals. The experimental design was not likely to result in any discomfort for participants and did not involve misleading them. The only stipulations were that informed consent was obtained and all data was held securely and separately from any identity data. As the participants included children, the researcher obtained 'Enhanced Disclosure' status: this check document is required by the Scottish government before an individual is permitted to work with children or vulnerable adults (Disclosure Scotland, 2007). In addition and in line with best practice for child protection, the researcher was always accompanied by the class teacher and was never in a position of responsibility for any class.

The Robot Dance study used a knowledge pre-test method. This was followed by one or other of the two robot-programming interventions: *dance*, which focused on syntax and sequence knowledge, or *follow the light*, which further required variables for the robot to follow a light source. Learners then completed an equivalent knowledge post-test. The test questions (Appendix I) related to the learning objectives (sequence, syntax and variables). The participants were asked to answer true, false or do not know. It was made clear to the learners that 'do not know' was an acceptable answer and it was to discourage guessing. When scores were processed, a participant was awarded a point for a correct answer, lost a point for an incorrect answer and gained or lost nothing for selecting 'do not know'.

The primary analysis of test score data involves detecting significant change between pre-test and post-test scores for each participant. A positive change would indicate an increase in performance in one or more of the learning objectives and respectively a negative change would indicate a decrease in performance. All paper tests were annotated with gender, group and participant identity (initials). This allowed the scores to be analysed with respect to the whole sample's pre-score and post-score and comparisons to be made between (i) male and female performances and (ii) the two challenges: *dance* and *follow the light* groups. Results are presented in the following order: firstly, Learning Outcomes 1 and 2 (sequence and syntax), which featured in both challenges, and secondly, Learning Outcome 3 (variables), which featured in *follow the light*.

Results

Sequence and Syntax

Tables 4.1 and 4.2 present summaries of pre-test and post-test means for sequence and syntax, indicating any significant p values from two-tailed paired t-tests.

Table 4.1: Robot Dance Results: Sequence (Range: -3, 3)

	Pre		Post		n	t	Df	Sig. (2-tailed)
	Mean	**Std. Deviation**	**Mean**	**Std. Deviation**				
Whole Sample	1.72	1.01	2.19	1.00	132	-4.75	131	< 0.05
Male	1.91	0.95	2.23	0.10	75	-2.60	74	< 0.05
Female	1.47	1.04	2.14	1.04	57	-4.19	56	< 0.05
Dance	1.43	1.06	2.03	1.14	98	-3.91	62	< 0.05
Follow the light	1.99	0.88	2.34	0.85	34	-2.78	68	< 0.05

Table 4.2: Robot Dance Results: Syntax (Range: -2, 2)

	Pre		Post		n	t	Df	Sig. (2-tailed)
	Mean	**Std. Deviation**	**Mean**	**Std. Deviation**				
Whole Sample	1.14	0.62	1.39	0.65	132	-3.70	131	< 0.05
Male	1.34	0.58	1.52	0.63	75	-2.41	74	< 0.05

Female	0.89	0.59	1.19	0.34	57	-2.81	56	< 0.05
Dance	1.10	0.67	1.38	0.68	98	-3.02	62	< 0.05
Follow the light	1.19	0.58	1.38	0.62	34	-2.20	68	< 0.05

Figures 4.4 - 4.8 illustrate the differences graphically alongside descriptive statistics that describe the distribution's limits and mean. To highlight the distribution of change in performance, the percentage of participants displaying a positive, negative and no change in performance also is noted.

Whole Sample

Overall, the sample had a mean pre-test score of 2.86 and a mean post-test score of 3.58 for sequence and syntax combined, with an improvement of 0.72. Participants in both sequence and syntax showed significant improvements (Figure 4.4). The minimum difference was -5 and the maximum difference was 10. The whole sample's change in performance was distributed as follows: 23% showed a decrease in performance, 20% had no change and the remaining 57% showed an improvement in performance.

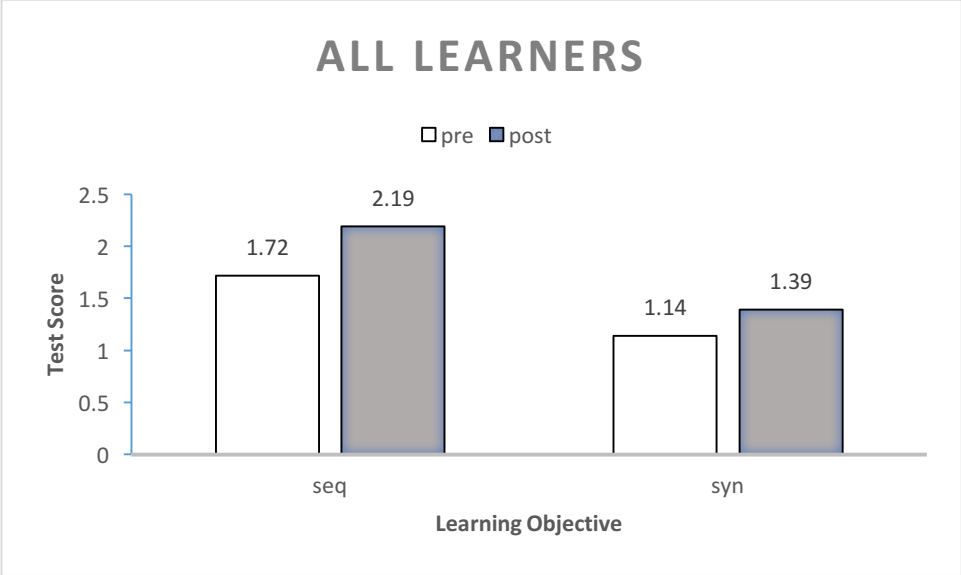


Figure 4.4: Results for Sequence and Syntax: Whole Sample

Gender

The gender split of 75 males to 57 females resulted in reasonably comparable group sizes. The male and female participants were distributed evenly throughout the 12 sessions.

The male group (Figure 4.5) had a mean pre-test score of 3.25 and a mean post-test score of 3.75, with an improvement of 0.50. The minimum difference was -5 and the maximum difference was 5. The male group's change in performance was distributed as follows: 27% showed a decrease in performance, 27% had no change and 46% showed an improvement in performance. The female group (Figure 4.6) showed a mean pre-test score of 2.36 and a mean post-test score of 3.33, with an improvement of 0.97. The minimum difference was -3 and the maximum difference was 7. The female group's change in performance was distributed as follows: 19% had a decrease in performance, 10% had no change and 71% showed an improvement in performance.

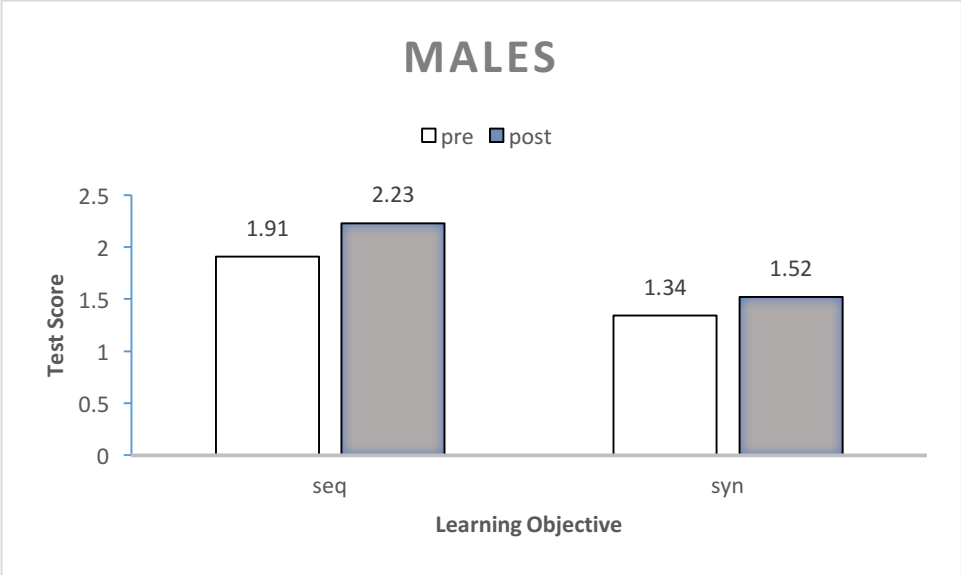


Figure 4.5: Results for Sequence and Syntax: Male Learners

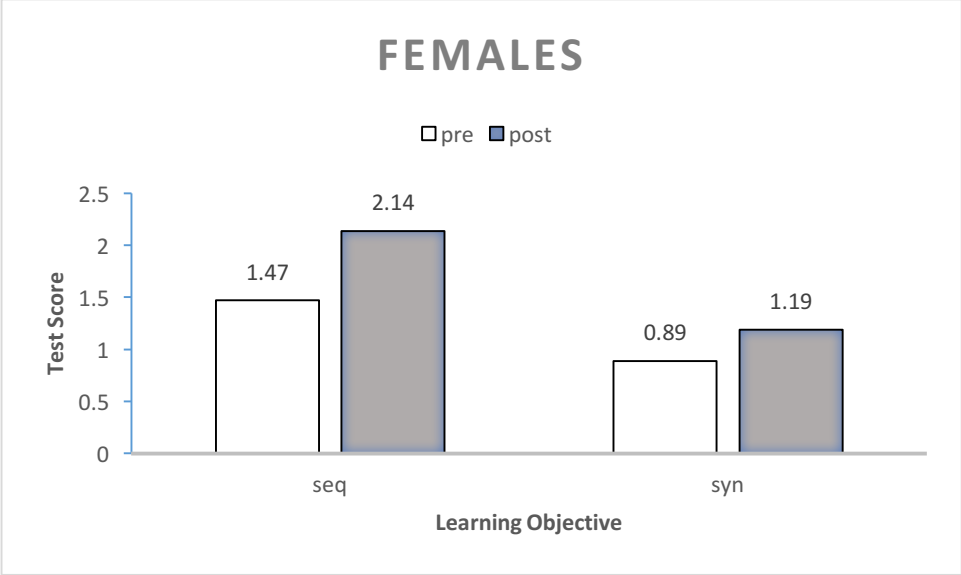


Figure 4.6: Results for Sequence and Syntax: Female Learners

Challenge: *dance* and *follow the light*

The *dance* group comprised 63 participants; the *follow the light* group comprised 69 participants. The dance group (Figure 4.7) showed a mean pre-test score of 2.53 and a mean post-test score of 3.41, with a difference of 0.88. In *dance*, both sequence and syntax were improved.

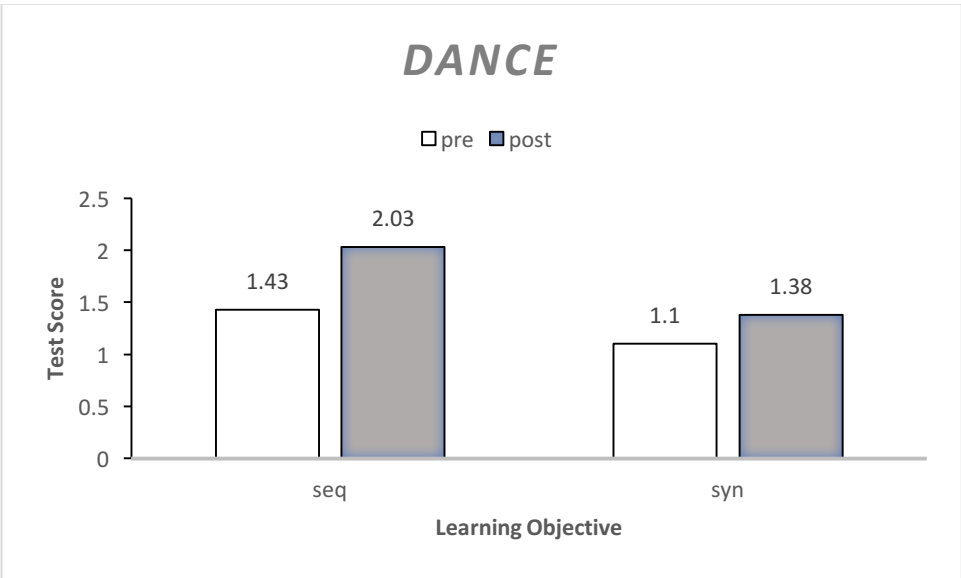


Figure 4.7: Results for Sequence and Syntax: *Dance*

The minimum difference was -3 and the maximum difference was 7. The dance group's change in performance was distributed as follows: 18% showed a decrease in performance, 20% had no change and 62% showed an improvement in performance.

The *follow the light* group (Figure 4.8) showed a mean pre-test score of 3.18 and a mean post-test score of 3.72 with a difference of 0.54. The minimum difference was -5 and the maximum difference was 7. The *follow the light* group's change in performance was distributed as follows: 35% showed a decrease in performance, 14% remained the same and 51% showed an improvement in performance.

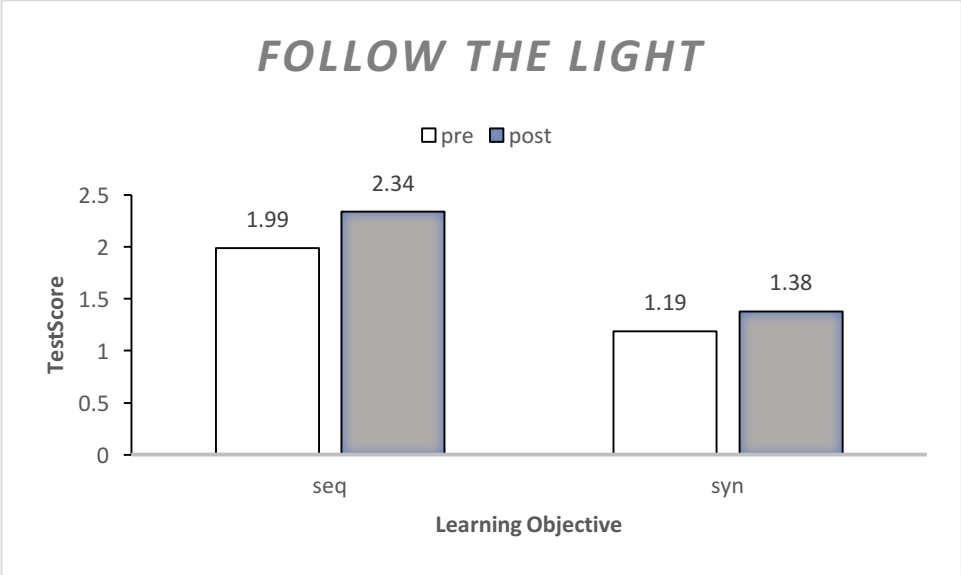


Figure 4.8: Results for Sequence and Syntax: *Follow the Light*

Variables

Table 4.3 presents a summary of pre-test and post-test means for variables. Figure 4.9 illustrates the differences graphically alongside descriptive statistics that describe the distribution's limits and mean.

Table 4.3: Robot Dance Results: Variables (Range: -3, 3)

	Pre		Post		n	t	Df	Sig. (2-tailed)
	Mean	**Std. Deviation**	**Mean**	**Std. Deviation**				
Whole Sample	0.53	0.73	0.89	0.80	132	-3.70	131	< 0.05
Male	0.70	0.78	0.98	0.79	75	-2.41	74	< 0.05
Female	0.38	0.64	0.80	0.80	57	4.62	56	< 0.05
Dance	0.32	0.53	0.62	0.73	98	4.46	97	< 0.05
Follow the light	1.11	0.88	1.59	0.50	34	3.53	33	< 0.05

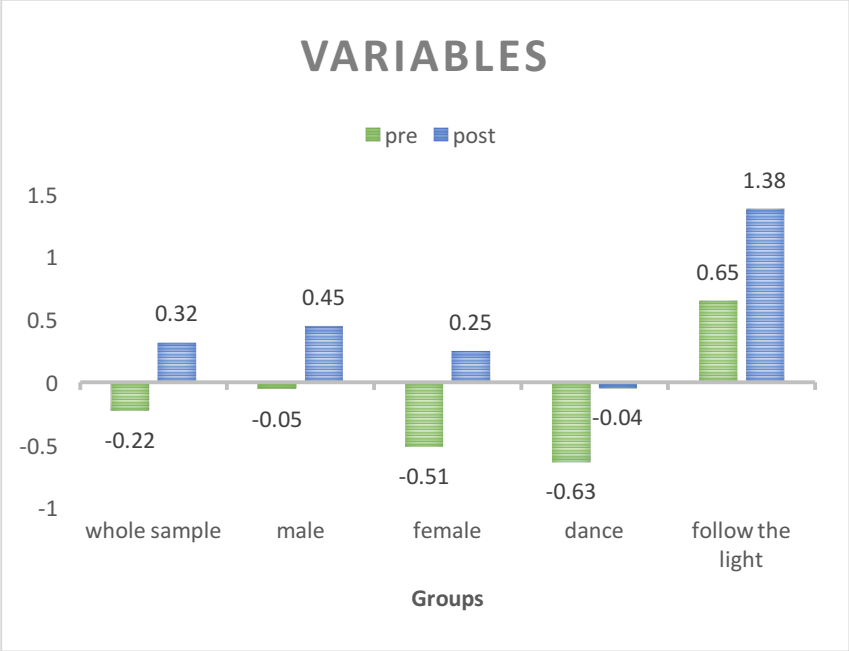


Figure 4.9: Results for Variables

All groups had differences between the mean pre-test score and the mean post-test score for variables. The *dance* group had a minimum difference of -2 and a maximum difference of 4. The *dance* group's difference in performance was distributed as follows: 10% decreased, 46% demonstrated no change and 44% improvement in performance.

The *follow the light* group had a minimum difference of -2 and a maximum difference of 4. The *follow the light* group's change in performance was distributed as follows: 14% showed a decrease in performance, 43% remained the same and 43% improvement in performance.

Discussion

Examining the whole sample (Figure 4.4), these results essentially are what would be expected: there were improvements as a result of 1-2 hours of tuition. However, the outcome - that such a wide range of learners successfully engaged in what is essentially C programming – is impressive. In a short space of time, with adequate support and motivation, learners were all able successfully to make a simple program execute and improve their knowledge in the areas of the three learning outcomes: sequence, syntax and variables.

The females' improvement was greater than that of the males. However, looking purely at change data may be insufficient. In a pre/post-test design such as this, a high pre-test score leaves less room to improve and produce a large difference. The male and female groups showed different pre-test scores and increased to almost the same level of expertise. This suggests that the female portion of the cohort had less programming expertise than the males at the outset. Robotics tasks are often perceived to be more 'geeky' and more motivating for a male cohort. One of the reasons to choose the dance task was to mitigate this and produce a context with which female learners may engage. The results support that the female participants engaged and performed well in the workshop. It is important to be mindful of the dangers of designing for stereotypes as you risk alienating users. Nonetheless, the combination of dance and robotics engaged the whole audience, with several teachers expressing their surprise that learners who had not previously shown interest in computing were now asking questions and driving forward their group's work.

With respect to the *dance* versus *follow the light* (Figure 4.6) there appears to be a small difference in performance relating to syntax and sequence expertise. The *dance* group improved slightly

more than the *follow the light* group. This may be a result of the *dance* group receiving a more focused session with less content to cover.

With respect to the improvement in the test scores for variables, the general consistency of the results is notable. One question raised is why the *follow the light* group had a positive pre-test score whereas all other groups had a negative pre-test score for variables. It is likely that this is a reflection of their age, being the only group of senior pupils. They may indeed have had more programming expertise at the outset. Irrespective of that, both *dance* and *follow the light* groups showed improvement with respect to variables, which is a good result to reflect upon further.

Further Observations

In addition to the 12 sessions reported where measures were taken, the Robot Dance workshop was delivered many other times to in excess of 1000 learners. This section reflects upon some of the adaptations and teaching practice that evolved through this time.

The first delivery of Robot Dance tool place as part of the Orkney science festival. The initial design of Robot Dance had a significant amount of up-front information about how robots are used. This was intended to establish the reason for the skills being taught. This rationale was reasonable, but after the first delivery this excess material was removed, as it involved too much content delivery without learner engagement. The concept of 'time to first task' was arrived at through this foundational work. It became clear that a key part of establishing interest and striking up a rapport with the learners involved having a 'time to first task' of around 10 minutes. This offered sufficient time to cover all the required knowledge to get the minimum viable robot program written and uploaded. The practice of demonstration followed by learner consolidation also emerged through the delivery and refinement of Robot Dance. When delivering to learners, an intuitive sense emerges of when learners are beginning to struggle to retain instructions. Robot Dance used an iterative process of content delivery and consolidation to ensure learners were able

to act rapidly upon instructions given. A key part of this became facilitation of the consolidation period. A number of facilitation strategies emerged while supporting learners in Robot Dance. When encouraged to talk through their program and explain their intentions, often the learners solved their own problems, as outlined by Chi et al. (1989) and Chi et al. (1994). Another valuable facilitation strategy was to encourage learners to simplify until an expected behaviour occurs, and then systematically re-introduce complexity. The key aim is to guide learners in the correct direction and teach them how to solve their own problems. This was observed to foster the participants' independence and remove reliance on the educator to fix problems for them.

Continued delivery of the workshop helped to identify other important features. One such was the performance element of Robot Dance, which was enhanced, as its importance became evident. Initially the dances took place on a tabletop. They were started when a bright torch was switched on and a sound was played on a laptop. As the workshops evolved, the inclusion of more powerful external speakers, a wood-effect dance floor and a stage light served to enhance the motivational effect of the end performance. In addition to the physical props, a narrative was developed that led up to this point. This drew an analogy with the experience of going to the theatre and the point at which the audience is brought to a prompt hush by the dimming of lights, the action then starting with music and sound.

Limitations

It is probable that for some learners this was not an optimal mode of learning. Kinaesthetic learners (Fleming and Baume, 2006) and Activist learners (Honey and Mumford, 1982) are likely to react positively to the hands-on nature of the session while others may find it more distracting. It is likely that the implementation constraints also affected the results. In a number of cases, the tight time for delivery resulted in post-tests being completed in a rush. To compound this, the energy level of a group of participants is likely to affect this also, as on arrival at the session the learners are fresh and after an intensive session they are somewhat fatigued. This may have been

amplified by the fact that participants were school pupils and likely to find an hour-long intensive activity tiring. Offering more time for the post-test would be a starting point.

It is important to reflect on the possible limitations incurred by study design and any uncontrolled aspects of the study. A pre/post-test design has a number of limitations. The only indication of the participant's prior experience is limited to their pre-test score. This is arguably a shallow representation of programming experience or competence. Various options for gaining a richer picture of prior experience were considered, including asking participants to list the programming languages they were familiar with or use a brief interview to determine past experience. These options were not used, as they could risk confusing participants by mentioning technologies or concepts that participants may not be aware of and the extra questions would extend the duration of the questionnaire. For the Robot Dance study, therefore, the pre-test score forms the baseline of expertise from which to measure change.

An important implication of this study design is an awareness of the potential for change. If a participant has a high degree of prior competence and performs well in the pre-test, there is little opportunity for a large improvement. For this reason, a mean change in performance may be skewed by participants who do not have anything new to learn from the workshop. For this reason, future work should explore the measurement of learning gain (McGrath et al., 2015) as well as learning change.

The Robot Dance study balanced scale with depth of inquiry. The paper-based pre- and post-tests used in the study scaled to allow engagement with multiple different learners but lacked the depth and openness of an observational study. The ability to be focused yet open-minded to unforeseen phenomena is one of the key advantages of ethnographic techniques (Ball and Ormerod, 2000). This is explored in the follow on study, Robot Dance in the Community, introduced in Chapter 5.

Conclusions

Through conducting the Robot Dance workshops, it was possible to show that a short physical robot-programming workshop yields a positive learning effect across a wide population of learners. Re-visiting the research question, "*How does a short physical robot programming activity support learning of introductory computer programming?"*, it is possible to identify some of the features of the learning experience that supported the learning effect.

Robot Dance delivered small pieces of skill and knowledge, giving learners space to explore and experiment 'hands on' with the new material. The delivery of a new concept followed by space to explore the example was repeated several times. This cycle supported a gradual increase in learner independence and task complexity.

Programming and physical robots have a tendency to be error-prone and this tight cycle of new skill followed by learner consolidation supported learners in fixing small problems. Another feature of the learning experience that was observed to motivate and drive learners was the performance aspect at the end of the session. It provided a good focal closing point and the opportunity to demonstrate new skills with peers. This was observed to drive communication in the learning groups, as the time to performance grew closer the sharing of ideas and refinements within the groups resulted in increased communication. The physical nature of the robots, dance floor and stage lights enhanced the value of the demonstration. Rather than gathering round a screen, learners were gathered round a physical location. The level of engagement was observable in the hushed silence as robots were set up and there was spontaneous applause on the completion of a dance. Audible gasps were also heard as robots meandered close to the edge of the dance floor with a risk of falling off. Learners were supported in learning to program with robots by the cyclical introduction of new skills with space to experiment with them. The growing complexity and opportunity to demonstrate the new skills resulted in a high degree of learner engagement.

The Robot Dance workshops conducted to date have been delivered within the confines of a classroom, with many of the concomitant restrictions. Workshops had to be completed in less than an hour, there was a fixed start and finish point, and the learners were restricted to secondary school age groups. A natural progression for this exploration of learning with physical robots therefore was to open the parameters wider to a range of learners, in a less formal learning environment and with a much looser structure that is driven by the learners. The next study did just this.

Chapter 5: Study II Robot Dance in the Community

Introduction

In study I, Robot Dance, learners of computer programming were observed and a number of factors contributing to the learning effect were noted. In study II, Robot Dance in the Community, learners of introductory programming skills again were observed, but in this second study the learners were given a greater degree of independence than in study I. The same core learning experience of study I is taken from the classroom circumstance and in this study it is delivered in a much looser fashion. Rather than a tight cycle of skill delivery and learner consolidation, learners were given a brief succinct introduction and left to develop their Robot Dance, asking for assistance as and when they required it. The learning experience was organised to be drop-in, therefore learners started at different times and could work as long as they wanted. Learners were also free to self-organise, which resulted in individuals, pairs, parent and child pairs and larger groups. Observation of this complex learning experience offers additional insights and addresses the following research question:

(Q2) Given freedom in a programming activity, how do learners organise themselves?

Description

Robots in the Community was a drop-in event set up in a shopping centre, with little control over who would attend. In addition to this, participants came and went as they pleased. For this reason and to maximise flexibility, a brief introduction was given to learners when they first arrived. Following this, they were left to program independently, seeking assistance if needed. In the case of some younger participants (age less than 6 years), their dance was implemented by a parent or one of the facilitators, under the direction of the child.

The brief introduction focused on what the robot's capabilities were. For study II, the original robots of study I had to be rebuilt since they were beginning to break. The only notable difference resulting was that individual red, green and blue LEDs were replaced by a single RGB LED capable of displaying a 255-bit colour range. The functions used to make the robot perform actions were introduced as instructions that the robot will understand. This analogy was then extended and the process of linking a series of instructions to produce a dance was likened to writing a recipe. The key part of the introduction involved ensuring that learners understood the 'instruction + delay pair'. The motion functions behave as if state switches: `forward();` changes the state of the two motors from being stationary, or rotating, to rotating in a particular direction that results in forward motion. The motors will continue to do this until the motor state is altered to change direction or stop. To control the distance the robot will travel, the learner must control the duration of the motor moving. This can be achieved by using the `delay` instruction between state changes. This was described to the learners as "give your robot an instruction and some time to do it". A further key part of the introduction is where to place new instructions and the importance of not editing the code skeleton.

The learners were given a very basic skeleton Arduino program to extend. To make this introduction concrete, learners were "walked-through" the program required to make the robot move forward a short distance. Once learners had successfully completed this task, the challenge of creating 20 seconds of dance moves was presented. The robot's entire capabilities were not discussed with the learners. For example, the functions for the RGB LED and the light sensors were not described in the introduction. They were contained in the documentation the learners are given, however, so the curious learner could explore these features independently.

Study Design

Robot Dance in the Community used participant observation (Smith, 1997). This approach was used to contrast and complement the test-driven approach used in Robot Dance. The purpose of

this second study was to explore some of the phenomena observed but not fully captured in the initial Robot Dance study. In contrast to the controlled, focused and limited nature of the pre-/post- test method used in Robot Dance, participant observation (Smith, 1997) provides an openness and flexibility that enables the capturing of unforeseen phenomena. In observational research, the researcher has been characterised by Hammersley and Atkinson (2007) as taking one of four possible roles: (1) complete participant, (2) participant observer, (3) observer as participant and the (4) complete observer. In the role of complete participant, the researcher is wholly concealed within the group being studied. Where the researcher is part of the group but his/her motives are known, they may be considered a participant observer. In some situations, participation in the group may come before the research topic, which results in the situation of observer as participant. The final situation is the complete observer, in which case the researcher does not interact with the group being studied. In Robot Dance in the Community, the researcher took the role of participant observer. The researcher was not a member of the groups engaged in the programming task, but was permitted to engage with the learners.

As this research involved human participants, ethical approval was sought and granted by the School of Computing Ethics Committee. Informed consent was obtained from parents or guardians at the point of involvement with the study. The study design did not present any ethical challenges, as participants were self-directed through the activity and unlikely to experience any physical or emotional distress as a result of participation. The study design did not require participants to be misled or placed under any pressure. As the majority of participants were children, all facilitators had to have Enhanced Disclosure (Disclosure Scotland, 2007). Additionally, in line with best practice for child protection, children were never left under the sole supervision or care of the facilitators. Parents were required to remain throughout the session. In several cases, parents actively participated in the activities.

The researcher was able to obtain demographic information from participants, such as age, gender and relationships amongst groups (friends, siblings, father, and daughter). In addition to this, it

was possible for the researcher to note group composition, gender split and age ranges of all 43 participants. Finally, the researcher was able to observe interactions and probe learners for their description of what they were doing. This rich engagement of learners tasked with robot programming was captured in scratch notes that were promptly developed into field notes (Hammersley and Atkinson, 2007) for subsequent analysis.

The field notes were scrutinised using thematic analysis (Braun and Clarke, 2006) to highlight observed phenomena relevant to the research question. Participants were arranged by their group composition, as this related to the primary aim of the research question. Within the group composition structure, themes emerged that informed further aspects of the research, such as expert traits, demonstrating understanding, reaction to product and independence.

Participant observation offers the opportunity to produce what Smith (1997) describes as a "thick description of the social interaction within natural settings". Learning is an inherently complex activity and utilising a sensitive and open technique such as participant observation is wholly appropriate.

Results

The learners observed comprised a group of six parents and 35 children. Parents are included where they performed an active role in assisting and encouraging as opposed to passive observation. There were 20 male children and 14 female, accompanied by four male and two female parents. The children's ages ranged from five to 15 with the majority around seven. Four distinct groupings were observed which are each described in the following sections. Figure 5.1 gives the mean, minimum and maximum age ranges. Results for each grouping are then provided in turn.

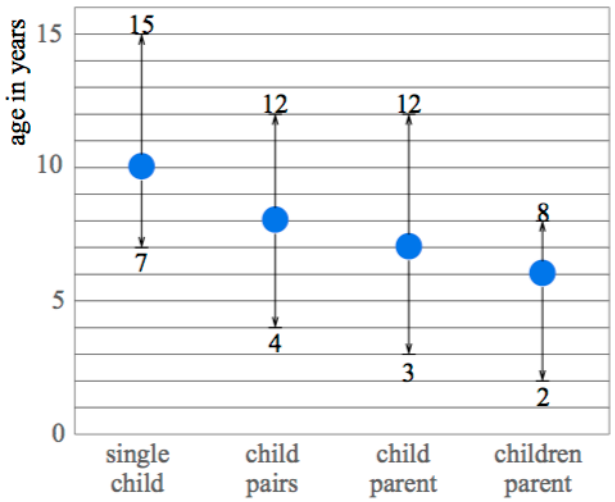


Figure 5.1: Age Ranges of Children in Robot Dance in the Community (maximum, mean and minimum age per group)

Single Child

The single child group comprised eight single participants who worked on their Robot Dance programs alone. Six of the participants were male and two female. The average age for this group was 10 years old. There were no observable differences in ability or enthusiasm relating to gender. Several learners in this group spent in excess of 25 minutes working on their Robot Dance program. With multiple attempts, they demonstrated good understanding of the relationship between program and robot actions. As a result, they were able to modify and refine their program until the desired outcome was achieved. Participants were also observed displaying excitement as a result of almost simultaneous identification of a problem in their program and identification of a possible solution. These learners were 'movers' (Perkins et al., 1989), with a positive response to problem solving. Closer inspection of the solutions they devised indicates they were indeed successful.

In addition, these learners tended towards a program that specified the desired 20-second dance with the additional finite space constraint. This is also evidence of a good personality for problem solving, the learners having self-selected the more challenging option. There was an observable pride in the final product with several learners showing their performance to their parents multiple

times. In this group, one learner made use of the RGB LED without requiring assistance. This involves understanding parameter passing and colour mixing. This information is detailed in the documentation provided and the ability to make use of this information autonomously was impressive. The length of time spent on task, the systematic testing and refinement and emotional response to bugs are all positive indicators of programming competence and a good programming experience.

Pairs of Children

Eight pairs of children were observed across the day. The average age of this group was eight years old. Three pairs comprised an older and younger sibling. There were an equal number of male and female participants. There was noticeable difference between male and female groups with respect to their emotional response to the task. Several of the female groups were extremely excited by their working robots. One pair of girls revisited the stall to make a different dance and was extremely pleased with their work:

> *"I just love this". Jane, age 9*

The older boys were less visibly excited by their robot programming. The majority of the groups were able to work independently after an initial introduction. Two of the groups got into some difficulty editing the skeleton of the program. Interestingly, these were the older and more able groups, who were exploring the bounds of what they had been shown and trying new things. Groups without technical difficulties were likely to be following instructions and exploring the programming less. In the case of the more able groups having difficulty, errors were an indicator of curiosity rather than an inability to perform the specified task. There was an observable excitement at making the robots react to the light in a number of the groups. The interactivity provided with the light sensing appeared to cause greater emotional arousal than with the single participants. There was good evidence of productive group work throughout the groups. A great deal of collaborative reasoning and turn taking was observed, with driving and navigating roles

evident in the development of the program. From a methodological point of view, this dialogue exposed much of the thinking of the learners to the observer. In the case of the individuals, the thinking remained internal unless questioned directly.

Child and Parent

There were six groups with a single child and parent working together. The average age of the one female and five male children was seven. The parents in these groups were supportive, and clearly had an interest in the activity their child was undertaking. In particular, the female participant's father had experience of working with micro-controllers. This pair spent a long time working with the robots. For the majority of the time, it was clear the daughter was driving the activity, well-supported by her father. They went through many iterations of the dance routine demonstrating a good understanding of the relationship between the code and the robot's actions. Towards the end of their session, there was a noticeable shift, with the father driving more as the daughter's attention waned. The child and parent group contained one very young participant, a three-year-old boy. He was able to dictate a series of instructions for the robot to perform and was very pleased with the result. Articulating a sequence of events was not a barrier. One participant in this group was very enthusiastic and was one of few who enquired about the RGB LED function and how to use it. After receiving an initial explanation, he worked well, supported by his mother, and was able to make use of the RGB LED in his performance. There was evidence that parent and child groups were working effectively, with all parents in this group observed supporting and facilitating their child's activities.

Multiple Children with Parents

Two groups comprised multiple children (average age of six years old) with parents. One large group of siblings worked for an extended period, well supported by their father. There was quite an age range in that particular group: the youngest was three years old and the oldest was seven years old. It was observed that this group did not have a strong understanding of the relationship

between the code and the robot's movements. The approach was more of trial and error or 'hacking' (Perkins et al., 1989), as is often observed in the novice programmer. They worked together for a considerable time and produced a dance they were all very happy with and keen to share with their younger brother. The other group in this category comprised two sisters, one of whom was a wheelchair user with significant fine motor control difficulties. They worked very well as a team, taking turns in driving the computer and suggesting ideas. They were able to produce a dance to which both sisters had an observable emotional response.

Clear indications were observed that both groups enjoyed this activity. However, they ultimately failed to demonstrate a deep understanding of the relationship between the program they were writing and the movements of the robot, as evidenced by repeated unsuccessful attempts at refining their program. This is perhaps not surprising, given the wide age range.

Discussion

There were no observable differences in programming ability with regard to gender. Both male and female learners presented a full spectrum of ability. The only noticeable difference between male and female participants related to enthusiasm. In several cases, a female participant's enthusiasm was clear, with one group returning to program another dance. In the male participants, enthusiasm was less overt.

The mean age decreased as the group's composition changed from individual, peer pair, parent and child and parent with multiple children. Unsurprisingly, the older groups were most independent. The correspondence between age and ability varied across all groups, which was expected, given that age is not known as an indicator of computer literacy or programming ability. The single and pairs of children were most likely to spend a prolonged time on task and were observed to have a good understanding of the relationship between the program and the Robot

Dance actions. These groups also showed most evidence of more ‘expert’ traits, such as testing and iterative refinement and making use of documentation.

Several individuals and pair groups were able to complete the programming task independently with minimal instruction or support. This suggests that children as young as eight can produce basic but functioning C programs, given the appropriate support.

All participants were very enthusiastic and enjoyed taking part in the workshop. This positive emotional response was not an indicator of good understanding of programming, however. It was clear that several groups were enjoying the activities but had little understanding of the relationship between the program and the robot's movements. Nonetheless, positive emotional response is an important underpinning attribute of an engaging learning experience.

Limitations

Some limitations must be contemplated with this study. Firstly, sample bias was possible: the sample observed was a reflection of the people who are in a shopping centre on a Saturday. Typically, these were children accompanied by parents. Few groups of teenagers entered the shopping centre throughout the day, which accounts for the age range and high likelihood of working with parents. The second possible sample bias effect was a result of the 'drop-in' nature of the event: people were self-selecting to participate. It is therefore much more probable that participants will have a predisposition to enjoying the type of activity we had on display. The sample size is also insufficient to assert confidently that the gender effects observed will generalise. Thirdly, a single observer made both the scratch notes and the field notes immediately afterwards. In the absence of verification, it is conceivable that possible observations were missed or misinterpreted. Finally, a Hawthorne-type of effect may have resulted from participants’ awareness of observation. These last two limitations are not judged to be major concerns, however. Regarding the single observer issue, the number and size of groups present at any single

point of time was not high. Regarding the observer effect, the participants evidently were motivated by the task, the robot and the immediate successes obtained – their awareness of the observer was very limited indeed.

Conclusion

Study II, Robot Dance in the Community, built upon Study I, Robot Dance, by exploring a much more open context for learning. It applied a different research method – observation – and a qualitative approach rather than a quantitative approach. In Robot Dance in the Community, learners were given a much greater degree of freedom to self-organise and drive the learning experience themselves than had been the case in study I. The main findings can be summarised by re-visiting the research question: "*given freedom in a robot based programming activity, how do learners organise themselves and engage with programming challenges that arise?*"

Four different groupings were observed: single child, child pairs, child parent pairs and multiple children and parents. The older and more able participants self-organised for individual or pair work and were able to cope well with the loose delivery mode. Most of the learners in this category were able to work independently following a brief introduction and demonstration. The parent and child pairs and groups tended to have younger learners and were less able to demonstrate a deep understanding of the programming skills they were learning. This reflects the age and possibly the competence of the learners. It may be that the looser delivery used in Robot Dance in the Community lacked the structure that these younger learners required.

All learners that took part in the activity demonstrated an observable emotional response to the performance they had programmed. Irrespective of the small audience, learners showed an observable pride in their creation.

Results from this study suggest that different learners require different degrees of support. In the classroom-based Robot Dance study, the high degree of structure supported all learners. In

contrast, the freedom to experiment and self-direct in the community study worked well for some learners but was more challenging for others. Robot Dance in the Community highlighted the extent to which programming has an emotional dimension. The next study investigates all of these aspects further: it examines one approach to the problem of giving learners both support and the ability to self-select pace, and explores how to get a more detailed picture of the learner's emotional response to programming tasks. The context for the next study is the importance of the affordances of a physical interface for learning programming and the emotional engagement with learners.

Chapter 6: Study III Whack a Mole

Introduction

Study III, Whack a Mole, was designed to explore how learning experiences were affected by learning with different interfaces: a physical interface or a screen-based equivalent. A recurring issue raised by the findings from Robot Dance (Martin and Hughes, 2011) was the extent to which the physical artefact mattered. Given that there are several practical issues related to using physical equipment in a learning setting, such as cost, maintenance of equipment and fragility, the importance of the physical artefact does need to be explored. The Whack a Mole study offered a comparison between two groups engaging in isomorphic learning experiences where the only difference is in the interface of the game they were programming. This study explores supporting learning to program with a physical medium that is not a robot.

Following on from study II, the content delivery method for the Whack a Mole study utilised video tutorials in an attempt to provide a high degree of support to learners whilst still ensuring they are in control of the pace at which the material is delivered. The final area of interest exposed in studies I and II was the extent to which learners were exhibiting an emotional response to the programming. Evidence from study I suggests that an engaging learning experience led to a measurable change in a learner's knowledge. Whack a Mole, study III, was designed to explore this further, attempting to capture some insights into which emotions were experienced by learners and in what circumstances. The Whack a Mole study explored this in the context of a comparison between physical and non-physical media. It aims to answer the research question:

(Q3) How does working with a physical artefact as opposed to a screen-based artefact affect learning of computer programming?

Background

Anecdotal evidence from programmers suggests that programming is an emotionally rich experience: bugs are frustrating, trapping them can be satisfying and solving complex problems can lead to increased pride in one's abilities. A further range of emotions can be evoked via collaborative working. Meyer and Turner (2002) describe the importance of emotion in an educational context. For this reason, study III includes a measure of emotional response to learning activities.

The emotional response to learning with technology has been well studied. D'Mello (2013) conducted a review including 24 studies, noting that many learning contexts resulting in engagement had comparatively low reporting of negative emotions. Pekrun (1992) conducted a detailed literature review from 1974 through to 1990, which was later extended to 2002 (Pekrun et al, 2002). This review included studies attempting to establish links between emotion and learning and achievement. Their review highlighted a bias in the research towards test anxiety: in excess of 1200 studies were found in this area, with other emotions receiving single digit or tens of studies at most. This reveals that broader emotion in an education context is an understudied area. Pekrun et al. (2002) proposed a set of nine academic emotions that are linked to achievement and learning. As well as anxiety, these include emotions that are positive and negative, and activating and deactivating: *enjoyment, hope, pride, relief, anger, hopelessness, shame* and *boredom*. The validity of this set of academic emotions and their link to learning and achievement was established through a number of studies utilising complementary research methods. Their findings imply students experience a wide range of emotions in an academic setting, with positive emotions represented in similar proportions to negative ones. Their findings also argue for emotion-oriented design of learning environments (Pekrun et al., 2002).

Although the study of emotional response to programming is limited, some interesting work has been done (e.g. Bosch & D'Mello, 2015, Bosch et al., 2013, Good et al., 2011). Bosch et al. (2013)

sought to map the emotional states a novice experiences and their relative proportion, as well as explore the co-occurrence of emotional states and the relationship between interaction events. In addition, they mapped transitions between emotional states. They used participant self-reporting at a very high frequency, sampling every 15 seconds. Following a 30-minute programming exercise, the participant is shown a web camera still of their face and the programming tool they were using at 100 random points in the session. At each of these points, they are asked to note their emotional state and asked optionally to note a second emotional state. In this study, 13 emotions are offered to participants: *fear, sadness, disgust, flow/engaged, anger, confused/ uncertain, surprise, natural, frustration, boredom, happiness, curiosity, anxiety*. This set has some overlap with that of Pekrun et al. (2002). This approach offers a rich picture of the frequency of emotions, although it does not capture the strength of the emotion. For example, happiness could be mild in response to a small success or intense if a substantial challenge has been overcome. This is a result of the primary research aim being to identify frequency of emotional states and transitions, rather than their intensity. Bosch also notes the limitations of the approach and the accuracy of participant self-reporting. Reflecting upon this, it would be interesting to attempt to determine the repeat validity of participants' responses, by offering them a number of situations multiple times and assessing if they report the same emotion. Although a young field of study at present, the studies conducted by Bosch and colleagues may inform the design of affective programming learning environments that can make decisions based on the learner's emotional state.

Good et al. (2011) have explored self-reporting of emotion to a quite different end. They conducted a study that evaluated two different approaches for students to self-report their affective state in an attempt to help students self-regulate their emotions. The study used a computer-based widget and a tangible device called Subtle Stone (Alsmeyer et al., 2008). The study concluded that there was a preference among students for the Subtle Stone. It had a number of advantages: it was more visible and increased the students' awareness of their emotional state. It also provided a visible representation that other students could see and respond to. The Subtle Stone can be

regarded as a physical application – a concept that is explored in Chapter 4 with Robot Dance as the context for learning to program). This is a single unambiguous artefact. The interface only does one thing and does it well. In the circumstance where a desktop-based solution is used, this becomes yet another thing competing for attention on the same communication channel as other interactions. In Good's study, a set of six emotional states were used: *enjoyment, pride, frustration, boredom, nervousness* and *confidence*. In the desktop application, the intensity of each state was also captured. In both of the studies discussed, the restricted set of emotions is appropriate because participants were required to report emotional state multiple times. Choosing between a list of 5 items and 50 are quite different tasks for the participant.

In study III, Whack a Mole, emotion was sampled as an indicator of engagement and as a potentially discriminating variable between the physical and non-physical setting. Details of the study and method, which were shaped by the studies described above, are described next.

Description

Whack a Mole is a game found in a number of cultures with different names, such as Splat the Rat, or Simon in the US. The essence of the game is simple: it tests reaction time via the ability to respond speedily to a series of stimuli. In the Arduino version of the game devised for this study, each of four LEDs has a corresponding button. When the light comes on, the player must press the corresponding button to progress through the game. In the simplest version, a light comes on at random and stays on until the corresponding button is pressed. This results in a 'playful interaction' but lacks some of the key elements that make a game. For example, it lacks user feedback: there is no indication if user is progressing other than via a subjective sense of getting quicker.

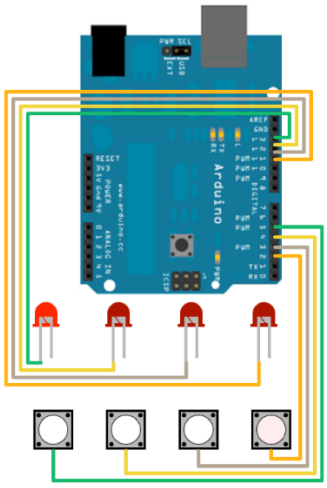


Figure 6.1: Screen-based Whack a Mole

There is also no defined goal at which a learner could aim. For example, if there were a goal relating to time, a player could strive to respond more quickly. The simple version of the game can be extended to introduce a timer for the light to stay on for a finite amount of time. This introduces a controllable element of difficulty. It is possible to provide user feedback when errors are made. An important feature is logging of correct pressed and incorrectly pressed buttons.

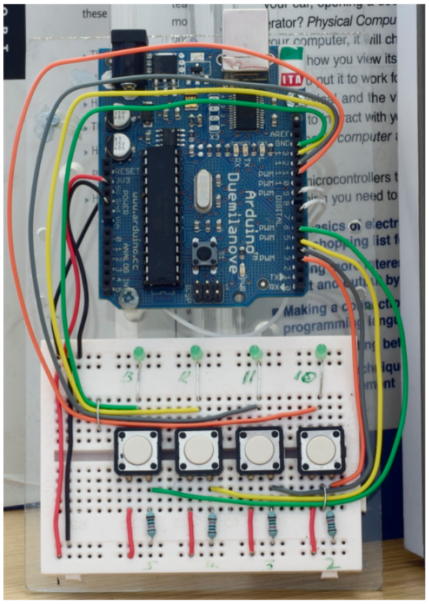


Figure 6.2: Physical Whack a Mole

Whack a Mole involved two phases for all learners. In the first stage, learners engaged in a controlled piece of tuition. The taught material was delivered via three specific worked examples. In the second phase, learners were required to demonstrate their understanding of the taught material from the first stage by applying the taught material to a novel problem.

A pilot version of this study was performed with volunteer student pairs and individuals. This identified potential problems. Firstly, if the learning material were delivered by the facilitator, there was potential for different aspects of the taught material to be emphasised with different groups. Secondly, there was a risk that the tuition would become a dialogue between the facilitator and the learner, resulting in different learner experiences. Whilst dialogue is highly desirable in a typical learning situation, it was undesirable in the situation of this study, since it could result in each group of learners having a significantly different learning experience. In the wake of these insights being revealed by the pilot, a set of learning materials was developed as a series of video tutorials. These were designed to ensure that the tuition given to the learners was consistent across multiple deliveries.

A set of four short video tutorials (2-3 minutes) was produced for the non-physical and physical version of the study. The single difference between the non-physical and physical videos was in the part of the video that demonstrated a completed task. In the non-physical videos, the screen-based Whack a Mole system was shown to demonstrate the taught code working. In the physical videos, this view changed to the physical game with LEDs, buttons and the visible Arduino.

The first video contained a brief introduction to the Arduino programming environment. It outlined the workflow of programming Arduino: code, compile, upload and test. This video also explained where the learner's code should be placed via the programming environment, as in each case there is a minimal code skeleton. The final part of the introductory video describes how to use the clickable documentation, which includes all the relevant Arduino functions required for the tasks and a brief description of what each does.

The second video walks the learner through the task of making a light blink (Figure 6.3). This is a traditional starting point for Arduino and is considered the equivalent of a `hello world` program.

```
1) int led = 13;
2) void setup(){
3)   pinMode(led,OUTPUT);
4) }
5) void loop(){
6)   digitalWrite( led, HIGH);
7)   delay(1000);
8)   digitalWrite( led, LOW);
9)   delay(1000);
10) }
```

Figure 6.3: Code Snippet for Task 1 Blink

Given that the Arduino has no straightforward method of displaying text, flashing an LED is the simplest program that does something observable. For both physical and non-physical groups, this task introduces digital output. Digital output requires the defining of a pin as an output. This involves making a conceptual mapping between the Arduino header-1, where the component is physically or virtually inserted, and the code that will control this pin and its attached component.

The learner must then use the `digitalWrite()` function to change the state of this pin from high (5 volts) to low (ground). This exercise shows the learner how to use a variable as an abstraction device to store the pin number. For example, if an LED is connected to pin 13, declaring an integer variable called `led` and storing the value 13 allows the variable with a descriptive name to be used in place of 13. This simplifies the code: instead of modifying the state of a pin number directly, the variable name adds meaning to the functions with which it is used. An example is `digitalWrite(13,HIGH);` to `digitalWrite(led,HIGH);`. To control the flow of execution the `delay` function is used to introduce a time space between state changes. This example also gives learners the chance to become familiar with the structure of an Arduino sketch: the `setup()` function runs once to initialise the board and the `loop()` function iterates infinitely to carry out the intentions of the programmer.

A second video walked through the code (Figure 6.4) for making a momentary light switch. This extends the previous example to include digital input. The learner has to identify a pin to be used with the button as a digital input. The idea of using a variable to abstract the pin number is also used to reinforce the concept. The learner must use the `digitalRead()` function to retrieve pin state information. This requires understanding that a function may have a return type and at execution time, the function call can be resolved to return type. It is possible to treat the `digitalRead()` function as its return which is HIGH or LOW, when a variable is used for the pin number this then reads as testing the state of the given component.

```
1)    int button = 2;
2)    int led = 13;
3)
4)    void setup(){
5)      pinMode(button,INPUT);
6)      pinMode(led,OUTPUT);
7)    }
8)
9)    void loop(){
10)     if (digitalRead(button) == HIGH){
11)       digitalWrite( led, HIGH);
12)     }else{
13)       digitalWrite( led, LOW);
14)     }
15)   }
```

Figure 6.4: Code Snippet for Task 2 Light Switch

Learners were then introduced to the `if` statement, which allows them to make a decision. In this case, they can make a decision based on the state of the button. If the button is pressed or HIGH then the LED is turned on or else the LED is turned off. This works because the main body of the code is contained in a void loop, which iterates as long as the Arduino has power.

The third video introduces the concept of an array as a device to simplify having multiple physical or virtual buttons and lights (Figure 6.5). Where before a single variable was used to abstract the button or LED pin, now an array can conveniently handle a collection of buttons or pins. Four physical buttons in sequence connect to consecutive digital general-purpose input/output pins (GPIO) pins that can become collected as an array of integers in the code.

```
1)   int[] button = {2,3,4,5};
2)   int[] led = {13,12,11,10};
3)   ...
9)    void loop(){
10)     for(int i=0;i<4;i++){
11)     if(digitalRead(button[i]) == HIGH){
12)       digitalWrite( led[i], HIGH);
13)     }else{
14)       digitalWrite( led[i], LOW);
15)     }
16)  }
```

Figure 6.5: Code Snippet for Task 3 Multiple Button

This required learners to use array notation to specify and initialise two arrays and form the association between the physical or virtual component, the IO pin and the code. The learners also had to use a fixed loop to iterate through the array, which is a typical strategy for combining arrays that are iterated together. This example highlights how the array index can link two concepts, in this case the buttons and the LEDs. When button `i` is pressed, LED `i` will be illuminated. This is a key concept for the second stage of the study, which required learners to demonstrate their understanding of the programming concepts taught via the video tutorial supported examples. The challenge was for learners to devise an algorithm for a Whack a Mole game that (i) demonstrated understanding of the concepts that had been taught and (ii) used some additional features found in the documentation, such as the random function.

The algorithm for the Whack a Mole game (Figure 6.6) consists of turning on a random light, waiting until the corresponding button is pressed and then picking another random light. This requires learners to demonstrate all the taught skills in context and integrate them into an application.

```
1)  int[] button = {2,3,4,5};
2)  int[] led = {13,12,11,10};
3)  int turnOn=0;
4)
5)  void setup(){
6)    ...
7)    turnOn = random(4);
8)    digitalWrite(led[turnOn],HIGH);
9)  }
10)
11)  void loop(){
12)    if(digitalRead(button[turnOn] == HIGH){
13)      digitalWrite(led[turnOn],LOW);
14)      turnOn = random(4);
15)      digitalWrite(led[turnOn],HIGH);
16)    }
17)  }
```

Figure 6.6: Example Code for Whack a Mole game

Study Design

The Whack a Mole study ran as part of an undergraduate module in Physical Computing. This module is taught to Level 1 (first year) applied computing, computing science, product design and interaction design learners in the School of Computing, University of Dundee. The class was organised into small practical groups of three or four. To ensure an optimal staff to learner ratio, the class was separated into two separate sittings. The two lab groups alternated between taught sessions and independent sessions. In one week, group A would have a taught lab while group B would engage in an independent lab assignment. The following week, the sittings were reversed. Learners were assigned randomly to either group A or group B at the start of semester and these groupings were used in the delivery of the Whack a Mole study. In the first week of the study, the taught group received the physical Whack a Mole intervention. This group had 22 participants of which 14 were male and 8 were female. The following week, the groups switched around and the taught group received the non-physical mole intervention. This group had 16 participants, 15 of whom were male.

As this study involved human participants, ethical clearance was sought and obtained from the ethics committee of the School of Computing in the University of Dundee. The study design did not present any risk of significant emotional or physical discomfort. However, the review of the study design and the ethical considerations did identify a conflict of interest, since the researcher was also the class lecturer for the module. Given that the study would take place as part of the module, it was essential to ensure that the activities outlined in the study were of value to the learners and warranted the use of module time. This was achieved by having the study reviewed by the module coordinator, who did conclude that the study made appropriate use of module time.

Two methods were designed to capture appropriate data for study III. Firstly, a paper-based questionnaire was designed to test knowledge and understanding of arrays. Secondly, a method was devised and piloted to capture a learner's emotional response to programming. These two methods are described in the next section.

Knowledge and Understanding

A paper-based questionnaire was designed to measure changes in knowledge and understanding of arrays. The questionnaire had three distinct parts (Appendix II). The first part contained four questions to test the learner's general level of knowledge of arrays, independently of any context or specific situation. Possible answers were *true*, *false* or *not sure* (Figure 6.7).

	true	**false**	**not sure**
Q1) An array is a group of items of the same type.			
Q2) Arrays start at item 1 e.g. array[1]			
Q3) An array is a contiguous collection of items of the same type.			
Q4) New items cannot be added to an array when the program is running.			

Figure 6.7: Questionnaire Part 1: Knowledge Questions

Question one relates to a key feature of an array: the data type of the elements. Question two tests knowledge of the starting index of an array, which in C style languages such as the Arduino language is element 0. Question three probes more deeply and requires the learner to have some model of the memory allocation and process associated with array use. Question four relates to the run-time inflexibility of an array in C and the fact that there is no method of dynamically changing the size of arrays at run-time.

The second part of the questionnaire is designed to explore the learner's level of understanding of arrays. Question five asks the learner to "*describe in your own words (and pictures) what an array is in the context of computer programming*". Constructing a description of a concept and externalising it in words and images requires a good understanding of the concept. It deliberately lacks a pre-defined framework into which learners could slot their knowledge. There is also no opportunity for learners to guess the answer.

The third and final part of the questionnaire requires the learners to respond to three questions associated with given code snippets that demonstrate array use within a small program. Question six (Figure 6.8) contains a fixed loop iterating from 0 – 9. Inside the loop, the counter variable is passed to a display function to display the value. This question tests the learner's understanding of the mechanics of a `for` loop in C syntax. An expected common error would relate to interpretation of the boundary point, `i <10`. In addition to this, there is a fair amount of syntax involved in the `for` loop, which learners must be proficient in if they are to produce the correct answer.

Given the following:

```
for(int i=0;i<10;i++){
  display(i);
}
```

What would you expect to be displayed?

Figure 6.8: Questionnaire Part 3 Q6

Question seven (Figure 6.9) provides the syntax for the declaration and initialisation of a five element array in Arduino C. This question tests understanding of the notation associated with array initialisation and the specifics of indexing and boundaries.

Given the following:

```
int[] array = {0,6,9,3,2};
```

What is array[0] equal to?

Figure 6.9: Questionnaire Part 3 Q7

Question eight (Figure 6.10) relates questions 1 and 2 to test the learner's understanding of array syntax and the use of fixed loops to mutate the values of each element in the array. To answer this correctly, the learner must demonstrate an understanding of using a fixed loop to iterate through an array.

Given the following:

```
int[] array2 = {3,4,5,2,3,5,6,7,9};

for(int i=0;i<8;i++){
  array2[i] = 0;
}
```

What would array2 contain now: ___ ___ ___ ___ ___ ___ ___

Figure 6.10: Questionnaire Part 3 Q8

Before beginning the first stage, the learners were given the questionnaire to complete independently under exam conditions, i.e. without conferring with peers and without external resources. After completing the study, participants were asked to complete the post-test

questionnaire. In addition, participants were also given the emotions questionnaire (described next) and advised how to complete it.

Emotional Response

The method designed to measure emotion was minimally disruptive for the learners. The decision was made to design a post-test questionnaire that learners could fill out as a reflective process. The studies discussed earlier involve multiple sampling, identifying the points at which an emotion occurred and any transitional states. A high frequency of samples requires a small set of possible participant responses and ideally the reconciliation of similar emotions, such as *calm* and *content*. The approach taken in Whack a Mole is the opposite. As the response from the participant is sought once at the end of the study, a broader range of emotions can be included. The instrument is not designed to measure when the emotion occurred in relation to other emotions. Instead, it is designed to capture *why* a state of emotion occurred. With more time available and without repeat sampling fatigue, participants are able to respond to a larger range of emotions and offer contextual information about what they were doing and why the emotion occurred. Where similar emotions are present, this provides several opportunities for a subtly different trigger to elicit feedback from learners. Amusement, elation and pleasure all fall under the heading of *positive lively* but may be attributed to different activities. For these reasons, a new method to obtain emotion data was designed, based on an ontology of emotional states: the Reflective Emotion Inventory (Appendix III).

The Reflective Emotion Inventory (REI) has been designed to capture emotional response in individuals. It is a reflective tool, designed to be delivered at the end of a session. It encourages learners to think back over their experience and indicate if they felt any of a range of emotions. The list of emotions used for the REI was derived from the HUMAINE project (Petta et al., 2011). HUMAINE's 'Emotional Annotation and Representation Language' proposes a core of 48

different emotions arranged into 10 sub-categories (Table 6.1). Table 6.2 lists the REI components developed for study III.

Table 6.1: HUMAINE Emotion Categories (Petta et al., 2011)

Negative			**Positive**		
Negative & forceful	1	Anger	**Positive & Lively**	25	Amusement
	2	Annoyance		26	Delight
	3	Contempt		27	Elation
	4	Disgust		28	Excitement
	5	Irritation		29	Happiness
				30	Joy
				31	Pleasure
Negative & not in control	6	Anxiety	**Caring**	32	Affection
	7	Embarrassment		33	Empathy
	8	Fear		34	Friendliness
	9	Helplessness		35	Love
	10	Powerlessness			
	11	Worry			
Negative thoughts	12	Doubt	**Positive thoughts**	36	Courage
	13	Envy		37	Hope
	14	Frustration		38	Pride
	15	Guilt		39	Satisfaction
	16	Shame		40	Trust
Negative & passive	17	Boredom	**Quiet Positive**	41	Calm
	18	Despair		42	Content
	19	Disappointment		43	Relaxed
	20	Hurt		44	Relieved
	21	Sadness		45	Serene
Agitation	22	Shock	**Reactive**	46	Interested
	23	Stress		47	Politeness
	24	Tension		48	Surprise

Table 6.2: Components of Reflective Emotion Inventory

Part 1	Part 2	Part 3
Which emotion has been experienced (see Table 6.1)	Degree of intensity for each: 4-point Likert scale	Comment: *why* and *when* was this emotion experienced

The REI questionnaire captures three things. (1) The learners are first invited to scan through the list of emotions and indicate if they have experienced any of them. (2) Following this, they can indicate the degree of arousal or intensity for each of the experienced emotions on a four-point unipolar Likert scale (Cummins and Gullone, 2000). A unipolar Likert scale was selected for two reasons. Firstly, given that the REI contains many emotions, there was a preference for a unipolar scale because it is easier for users to respond to than a bipolar scale that places opposites at either end of the scale. Secondly, the REI is intended to be a reflective tool that captures emotions experienced over a period. It is therefore quite possible that opposing emotions will be experienced at different times throughout the event. (3) Once the learners have noted emotions they have felt and the degree of arousal, they are encouraged to offer some contextual information in a free-text response space. The purpose of this is to describe why they experienced the given emotion. An example response might be: Annoyance, 3, "Getting the wires in the correct place".

This two-part study design offers the ability to capture change in knowledge and emotional response to programming. Analysing the result will give an insight into any difference between groups.

Results

Knowledge and Understanding

Change in knowledge and understanding of learners was measured with a paper test given at the outset of the session and at the conclusion of the session, as described in the study design. The following section considers the non-physical and physical groups' results, first as a whole and then considering each individual question.

Table 6.3 presents an analysis of the scores of both groups.

Table 6.3: Non-physical and Physical group scores (%)

	Pre-test		Post-test		n	t	Df	Sig. (2-tailed)
	Mean	Std. Deviation	Mean	Std. Deviation				
Non-physical	75.63	14.59	78.75	10.25	16	-1.32	15	> 0.05
Physical	57.27	20.74	60.90	19.50	22	-2.01	21	> 0.05

The non-physical group scored a mean pre-test score of 76% and a mean post-test score of 79%, with an improvement of 3%. The physical group scored a mean pre-test score of 57% and a mean post-test score of 61%, resulting in an improvement of 4%.

Figures 6.11 and 6.12 give histograms of the pre- and post-test score distribution overall for the non-physical (Figure 6.11) and physical (Figure 6.12) groups. Table 6.4 presents the mean score per question for the each group of pre-test and post-test.

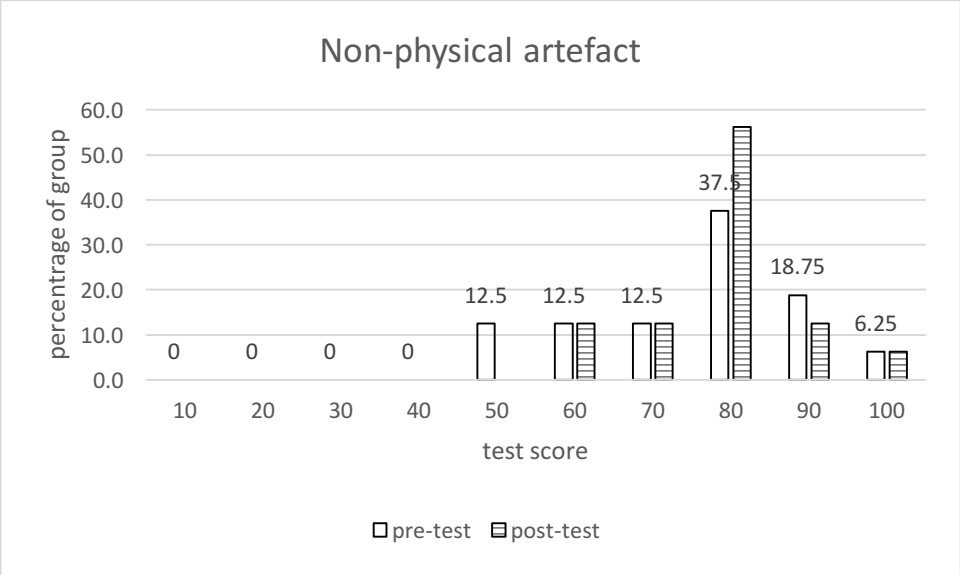


Figure 6.11: Distribution of Pre- and Post-Test Scores: Non-Physical Artefact

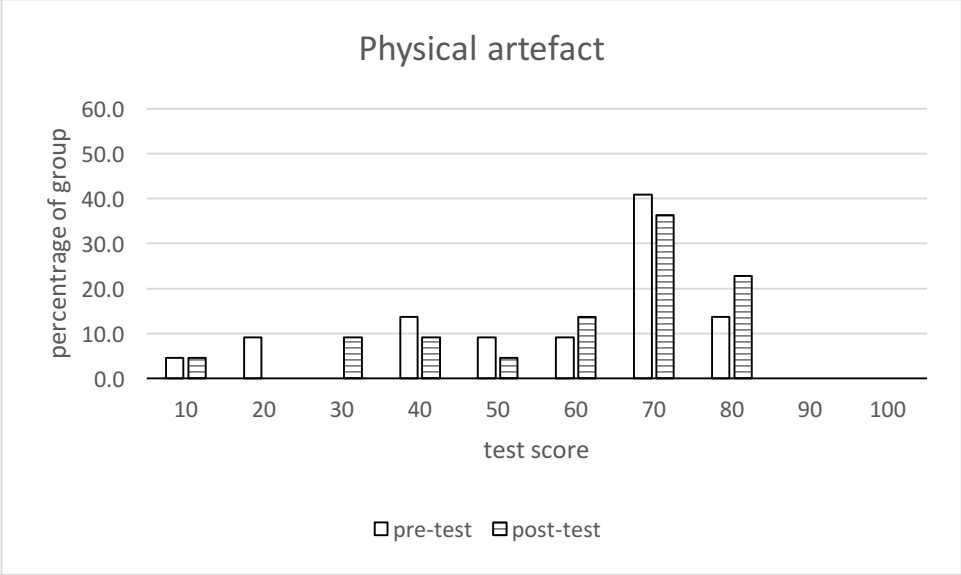


Figure 6.12: Distribution of Pre- and Post-Test Scores: Physical Artefact

Table 6.4: Whack a Mole Mean Score per Question

	Pre-test		**Post-test**	
	Non-physical	**Physical**	**Non-physical**	**Physical**
Question 1	100	95	100	95
Question 2	100	91	100	82
Question 3	94	64	94	68
Question 4	38	45	38	64
Question 5a	60	61	66	70
Question 5b	44	27	56	27
Question 6	81	50	81	55
Question 7	100	73	100	73
Question 8	81	5	89	5

Questions 1 and 2 proposed a low-level description of an array to test knowledge of the array syntax and the basic concept of an array. Question 3 proposed the array as a contiguous collection of items, probing deeper knowledge of the memory management associated with arrays and the inability to append items to an array at run-time. Question 4 proposed that it was not possible to add a new element to an array at run-time. Question 5 required participants to describe in their own words and pictures what an array is. The scoring for this was split into two sections: 5a

related to the textual description offered, one point being given for a superficial yet correct description of an array as a collection of items of the same type. For an answer to receive two points, it had to contain more detail, referring to the allocation of memory or size at run-time. Part 5b offered a point for a supporting diagram that added to the answer. In each case, zero points were awarded for an incorrect or absent response. Table 6.5 shows the distribution of pre-test and post-test results for the non-physical and physical groups for question 5a. Figure 6.13 gives the average score for each group.

Table 6.5: Distribution of Question 5a Pre-Test and Post-Test Scores per Group

Score	0 (incorrect or no response)	1 (correct but superficial response)	2 (correct and detailed response)
Non-physical pre-test	19	44	38
Non-physical post-test	6	56	38
Physical pre-test	32	14	54
Physical post-test	18	23	59

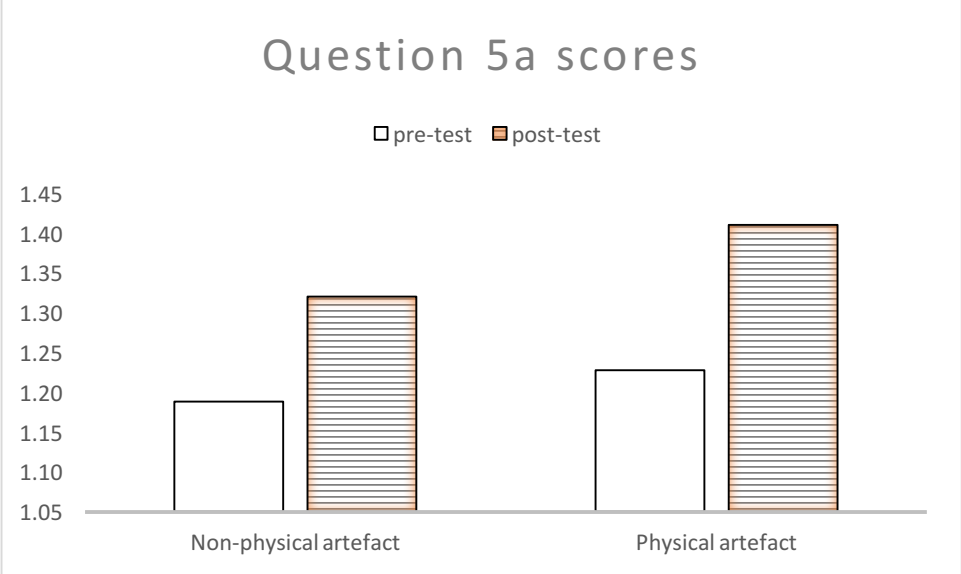


Figure 6.13: Question 5a Mean Score per Group

Both groups demonstrated improvement with this question. The non-physical group produced a good pre-test mean yet there was improvement in the proportion of the group that provided a shallow description (score of 1) of an array. In the post-test condition, only 6% of the non-physical group were unable to offer a description of an array (score of 0), demonstrating the group had a good grasp of what an array was. In the non-physical group, there was no change in the percentage of the group that provided a detailed description (score of 2) of an array from pre- to post-test.

In contrast, the distribution in the pre-test for the physical group was closer to bi-modal, with one third of the group unable to describe an array. There was a subsequent reduction of 14% of learners unable to answer the question. The physical group increased both the shallow and detailed description of the array by 9% and 5% respectively, suggesting an increased understanding of arrays.

Considering question 5b, 47% of learners in the non-physical group successfully used a diagram to illustrate their answer in the pre-test. In the post-test, this had increased to 60%. In the physical group, only 27% of learners chose to illustrate their description. This did not change from pre- to post-test (Table 6.6). The type of diagrams produced were typically a collection of adjacent boxes positioned either vertically or horizontally. In some cases, these were annotated with data and indexes.

Table 6.6: Percentages of Groups with Successful Diagram (Question 5b)

	Pre-test	**Post-test**
Non Physical	47	60
Physical	27	27

Question 6 presented a code snippet containing a `for` loop with a counter variable `i` initialised to 0 and a termination criteria of `i<10` stepping in increments of 1. Inside the body of the loop, the counter variable was being displayed. Learners were asked to describe what the output of this

code snippet would be. This was intended to measure understanding of fixed loops, including syntax, and importantly the understanding of the boundary condition `i<10`, meaning that the loop will display numbers from 0 - 9. The non-physical group performed well in this test, with a mean pre-test score of 81%. This did not improve in the post-test. The physical group had a pre-test mean of 50%, which improved to a post-test score of 55%.

Question 7 required learners to demonstrate understanding of array syntax and retrieval of data from arrays with indexing. The non-physical group scored 100% in the pre- and post-test score. The physical group scored 73% for this question and showed no improvement in the post-test.

Question 8 required learners to demonstrate understanding of the strategy of using a fixed loop to initialise an array. The non-physical group scored 81% in the pre-test and improved to 89% in the post-test. The physical group performed very badly with this question, scoring only 5% in the pre-test and failing to improve this in the post-test.

In each group, there were three distinct classes of learners. Some learners improved their performance, some showed no change and some performed worse in the post-test. The physical group performed slightly better than the non-physical group across all aspects, with a greater percentage of the group improving and fewer reducing their pre- to post-test performance (Table 6.7).

Table 6.7: Percentages of groups with differences in performance

	Non-physical	**Physical**
improved	25	32
no change	62	59
decreased	13	9

Emotional Response

Emotional response data was collected by the Reflective Emotion Inventory (REI) described in the study design. Of the non-physical group, 16 learners completed an REI. In the physical group, 22 participants completed an REI. The REI captures 48 different emotional states that may occur throughout the process of programming. The results are presented at the level of 10 sub-categories, with the first five being negative and the latter five positive. The intensity scale for each emotion ranged from 0 to 3, where 0 indicated no emotion and 3 indicated the emotion occurred intensely. There is also a space for contextual response about when or why the emotion was experienced.

Figure 6.14 shows the mean response from each sub-category of emotion for both groups. The non-physical group did not offer free text comments to contextualise their emotions as readily as the physical group did. They expressed *negative forceful* emotions that were cited as being the result of problems with the code: "code errors" or "sorting some issues with the program". Several participants in the non-physical group intimated feeling envy when other groups had their program working before they did. Several learners also expressed a feeling of friendliness as a result of working in a group. One group noted a feeling of worry "if they could complete the task on time". In contrast, other groups indicated a sense of boredom at being finished early.

Positive emotions for the non-physical group were largely cited because of completion of the task and "getting it working". This was attributed by many participants to a feeling of amusement, joy and happiness. The contextualising of positive emotions was as frequent as that of negative emotions. However, the reasons cited for a positive emotion were far less diverse.

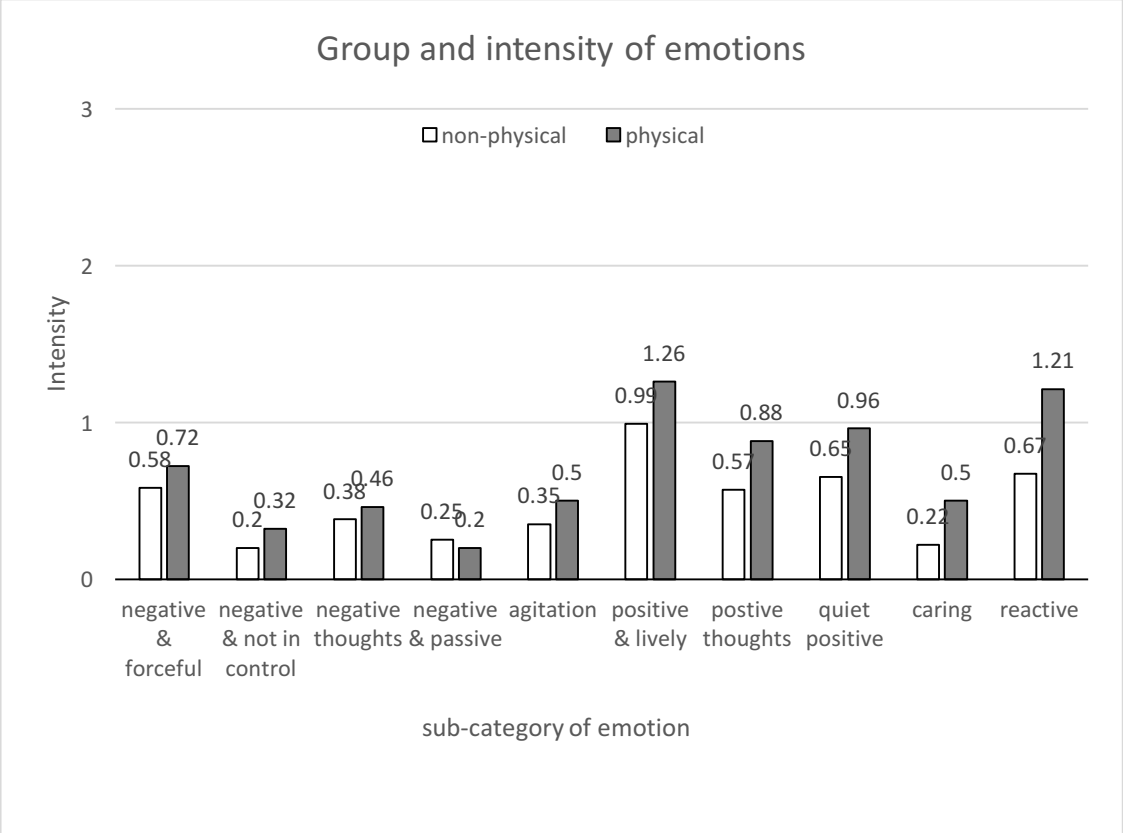


Figure 6.14: Strength of Emotional Responses: Whack a Mole

The physical group offered a number of comments for each sub-category of emotion. Negative emotions were attributed to a range of features of creating the Whack a Mole game. One of the most frequently cited situations resulting in negative emotions was wiring. Many participants just stated the single word "wiring", while others elaborated. Responses include "when the wires fall out", which is a common problem if jumper wires are not cut long enough or well organised. As with programming, there is often a tendency by the novice to get "stuck in" to the task and not plan their actions well. "Getting the wires in the right places" was also expressed as a problem by some (the pitch of the breadboards used is one hole every millimetre, which can be problematic). Specific components were mentioned by some: "getting the LED the right way" was noted by one participant, with another noting "wiring up resistors". LEDs have a polarity and require both the signal and ground voltage wires to be in the correct position. Resistors, on the other hand, do not have polarity but are very small, and placing them into breadboards can be problematic.

These type of difficulties were most prevalent under *negative forceful* category, with learners frequently associating these difficulties with feeling anger and annoyance. This category was the most strongly reported negative emotion in the physical group. To a lesser extent, these difficulties also appeared under the *not in control* category, such as rage. Several participants cited negative thoughts related to whether their build would work or not. Also in the negative thoughts category, frustration was related to wiring-up of the build. Interestingly, frustration was also cited in response to poorly specified compiler errors. It is fair to say that the Arduino IDE provides much more novice-friendly compiler errors than an industry standard IDE such as Eclipse or Visual Studio. Nonetheless, there are inevitably situations where there is disconnect between the error, the specific line of code and the description offered in the IDE. One or two of the learners expressed passive emotions such as boredom, at being finished early. Being stressed was also noted by several individuals in response to the system as a whole (wires and code) not working, or being unsure as to whether they would complete the build on time or not.

Positive emotions were contextualised with free text comments less richly than negative emotions. However, positive emotions were given greater intensity than negative emotions. *Positive and lively* was the most strongly reported emotional category of all. This was heavily noted by participants because of completing the build: "when it worked", and when engaging with the product of their work: "playing the game". People also noted a feeling of happiness at getting their task completed. The second most strongly reported emotional category was *reactive*. This was cited as interest in "learning new things". One participant explicitly noted interest in the logic they had arrived at in developing the Whack a Mole algorithm.

When considering the non-physical group's emotional response grouped together as positive or negative, there was a noticeable difference between the positive and negative emotions reported. Positive emotions were experienced by all participants to a greater extent than negative emotions.

It is notable that in the physical group there was a difference in intensity of positive and negative emotional categories as a whole. With the exception of *caring,* all the positive emotions have greater intensity than the negative ones. This matched the rich contextual data offered by the physical group. Where learners worked with the physical artefact, they had a strongly positive experience. Two of positive emotions reported by the physical group are notably greater than that of the non-physical group: *positive & lively* and *reactive*.

Discussion

Knowledge and Understanding

As the results indicate, despite allocating participants randomly, the two groups are clearly of different academic abilities or levels of experience (Figures 6.11 and 6.12). The non-physical and physical groups were statistically different as indicated by the histograms of result distribution. The histogram for the non-physical group (Figure 6.11) shows a tight normal distribution centred on a very high mean. The physical group has a slightly skewed distribution in pre- and post-tests, with several particularly weak scores (Figure 6.12).

It is striking that in both groups, around two-thirds of learners showed no change in knowledge or understanding about arrays and associated strategies (Table 6.2). The most likely explanation for this is the fact that when both the non-physical and the physical groups are ordered for performance, the top two-thirds of both groups have very high pre-test scores, leaving little room for improvement. It is therefore likely that this study took place too late in the teaching period and offered a substantially reduced opportunity for the interventions to create a change in knowledge and understanding. The groupings for this study having also proved problematic. The two randomly allocated groups have differing levels of pre-test scores and this may have reduced the sensitivity of the questionnaire to detect improvements between groups. The individual knowledge and understanding questions are now considered for qualitative purposes, to identify areas where prior knowledge reduced the sensitivity of the pre- and post-tests.

Questions 1 and 2 were answered correctly by almost all participants in both the non-physical and physical groups. This indicated that in both groups, this knowledge had been acquired by the learners elsewhere and unfortunately indicated that test had little sensitivity to change with respect to that knowledge area. In Question 3, the non-physical group had a pre-test score of 94%: this indicated prior knowledge with little room available for improvement.

Question 4 had contrasting results: there was no change in value at all for the non-physical group, but a significant improvement for the physical group. This could be because of the physical representation of the array in the form of the physical buttons and LEDs.

Both groups demonstrated improvement in their answers with Question 5. After Whack a Mole, there was a noticeable reduction in the proportion of the physical group who were unable to describe an array.

Question 6 did not elicit any particular insights. In Question 7, the non-physical group evidently were familiar with this syntax from previous programming experience and therefore the question did not provide any insights.

Question 8 raised a concern, because the physical group scored so badly in the pre-test and did not improve in the post-test. Given that all groups had used the relevant syntax in the course of the study, it might be presumed that the physical group learners approached the task in a shallow fashion and copied syntax without attempting to understand the syntax.

To summarise, only Questions 4, 5 and 8 provided useful information about how study III had affected knowledge and understanding.

Emotional Response

Firstly, with regard to the REI, a low response was noted for the free text component of the REI. This is unsurprising, given the additional effort required by learners to verbalise the contexts in which they felt a given emotion. Secondly, considering the two different groups, the REI did establish different responses from the two groups. The non-physical group noted envy and friendliness. Taken together, these describe a competitive situation well, particularly if a tight group has formed and they are keen to demonstrate their ability relative to the other groups they are working with. The REI also established different emotions about completing the test on time. Differing rates of task completion is evidence that the non-physical group, despite having a high knowledge and understanding test score, still contained a range of abilities.

With the physical group, almost all of the positive emotions have greater intensity than the negative ones. Most often, the anecdotal references to programming and emotion are focused on negative feelings. The free-text contextualisation presented here shows that participants frequently experienced many causes of irritation that are well reported in the literature, including unintelligible compiler errors and syntax errors.

It is interesting that the physical element in many respects confounds many of the areas of programming difficulty. Breadboarding with electrical components is an inherently finicky task requiring good eyesight and a steady hand. It also has many of the same problematic features as programming, such as error-prone nature, requiring high degree of detail, tracing of routes through a connected network and a one-to-many mapping from problem to solution. In addition to these problems, whilst programming offers compiler errors to assist the learner in trapping errors, there is no such support when wiring breadboards. As a result, errors in electrical circuits are often very difficult to identify. It seems counter-intuitive therefore that placing programming and electrical prototyping activities together can improve the emotional response to the programming experience. The dominance of positive emotions being reported suggests that this

happened in the Whack a Mole study. The results suggest that creating a functioning physical mole game has presented a sufficient challenge for most across a range of skills. The resultant completion of the task generated an emotional response that outweighs the 'pain' endured in working through the task. One theory to propose is that this is a result of the different bandwidth of interaction offered by the two systems. In the non-physical group, learners can only interact with a single device, namely the PC being used to program the virtual mole, giving a screen to offer feedback to the user and a mouse and keyboard to accept input. In constructing a mole game with the physical system, the Arduino, buttons and LEDs used to make the tangible game all increase the bandwidth of interaction. This may contribute to the richer more positive emotional response from learners in the physical group. This is depicted in Figures 6.15 and 6.16.

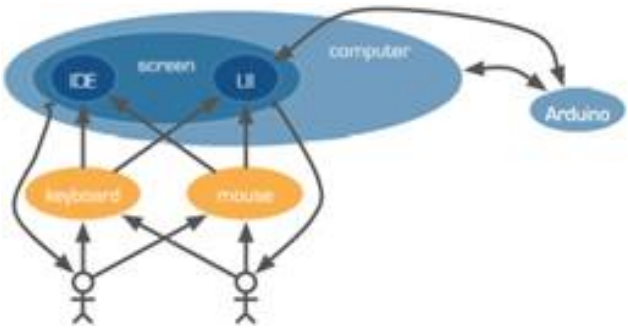


Figure 6.15: Non-physical Artefact Interaction Pathways

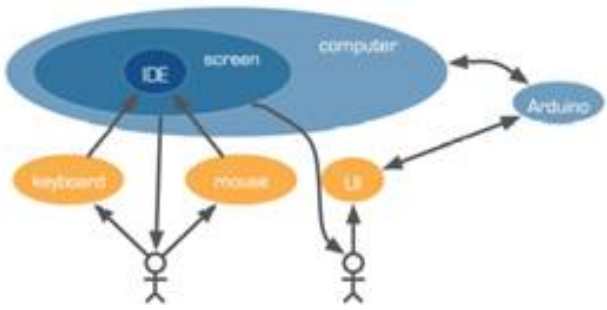


Figure 6.16: Physical Artefact Interaction Pathways

Having a low ratio of negative emotions to positive emotions may signify a learner who will do well with programming. It resonates with Perkin's findings of movers and stoppers. The ability to take greater pleasure from the competed task than displeasure experienced by the challenges on the road to success may be an important attribute for an aspiring programmer.

In the Whack a Mole study, all participants were able to complete the task. Unfortunately, across both groups, around 60% of learners showed no measured improvement in performance, which draws into question the efficacy of this approach. The Whack a Mole study also used video tutorials to replace delivery of content by a facilitator. This was in response to findings from Robot Dance and Robot Dance in the Community. The studies contrasted the benefits and pitfalls of tightly delivered content versus a much looser learner-led approach. The intention was that the video tutorials would enable learners to control the pace of content delivery. While this was effective up to a point, it suffers the same problem as giving someone a list of directions rather than a map. If they fail to act upon one of the directions, they can become lost. A list of directions also offers no contextual information for exploring different routes or other points of interest along the way. One of the key advantages of facilitating a session directly is the ability to identify spontaneous learning opportunities. The instructional videos have served as narrow routes for learners to take. In addition, from a reflective standpoint, using video content to support delivery left the facilitator far more removed from the process than when tight cycles of delivery and consolidation were used.

There was an observable difference in the degree of engagement of different groups with the finished artefact. Several of the participants in the physical group could be seen taking pictures and videos to share on social media. This indicates a degree of pride and desire to share their work that was not observed in the screen-based group.

However, the lack of difference in performance between the physical and screen-based groups indicates that many of the motivational affordances of the learning experience were not tightly

coupled to the tangible nature of the artefact. The task was fairly closed and offered a good degree of support for the learner. The playful interaction that was created may have been enhanced in the physical setting, yet it was still an effective motivator in the non-physical setting. Screen-based simulation may be an effective method of resolving some of the problems associated with using physical hardware such as cost, fragility and ongoing maintenance.

Limitations

One limitation of the Whack a Mole study resulted from the composition of the non-physical and the physical groups. It is not ideal to have two groups of different sizes and of different academic abilities. A solution to the problem would be to administer the pre-test and then create groups based on the score. Unfortunately, it could be problematic to implement paper tests in a single study and in this case, it would have disrupted the established groups within the class.

One of the challenges with a pre/post-test methodology is pitching the test difficulty correctly to ensure maximum sensitivity to the phenomena being observed, which in this case related to knowledge and understanding of arrays. The pre-test knowledge results suggest that in many cases an understanding of arrays has developed prior to the study. As a result, for many of the learners the measure had limited sensitivity. Despite these difficulties, the Whack a Mole study offers some valuable insight into the differences observed in novice programmers working with non-physical and physical media. This study contributes to multiple elements of the Learning Dimensions, which are introduced in Chapter 8.

Conclusion

The Whack a Mole study aimed to explore how learning with a physical device differed from learning with a screen-based equivalent. The main findings of this study can be summarised by referring to the research question posed: *how does working with a physical artefact as opposed to a screen-based artefact affect learning of computer programming*?

There was no noticeable difference in learning effect measured between the two groups, indicating that the physical interface did not measurably contribute to learning. The relatively small learning effect observed in both groups can be explained partly by the high pre-test scores for both groups. There was little space for improvement. However, there was a difference in emotional response to the learning experiences. Both groups described a range of negative emotions with similar levels of strength and for similar reasons. Both groups also noted a similar range of positive emotions. However, the physical group noted a greater strength of positive emotions associated with the learning experience. Study III also indicated that video-based teaching materials do not offer the opportunity for the interaction and subtle response that a facilitator can provide in probing areas of difficulty for the learner.

Studies I to III explore relatively closed problems supported by varying degrees of structure in a relatively short space of time (between 1 to 2.5 hours). One of the main tenets of constructivism is that learners should engage in projects that are relevant to them and the world around them (Vygotsky, 1980). In the Whack a Mole study, there was no difference in learning effect between the screen based and physical group. If this session were to be adapted to enable a greater degree of flexibility, for example allowing learners to design an interface for the Whack a Mole game, there would be no additional programming overhead to create a physical game. All that would be required would be longer wires for the buttons and LEDs that could be embedded in any number of craft materials. For the same to be done with a screen-based solution, additional skills would need to be taught, adding to the complexity of the session.

Re-considering the literature, it is worth noting that the sample task learners engaged in for the study by Bosch et al. (2013) was a traditional CS1-style maths based problem. Although this problem type is valid, it represents what Robins et al. (2003) argue is a knowledge-driven approach to programming education. This thesis argues for an approach to programming education that is more stimulating and framed within a context of value to the learner. The approaches taken in this thesis suggest that this style of context-free programming education is

part of the challenge of teaching programming. One of the most powerful affordances of physical computing is the ability to take intangible things and make them physical, for example using an LED to indicate state. In the next study, this ability to embed programming very flexibly in a rich and meaningful context will be explored. Study IV, Digital Makers, will utilise all the findings thus far for session design and move further to empower learners to create programs for problems they define and personally identify with.

Chapter 7: Study IV Digital Makers

Introduction

Study IV, Digital Makers, rests upon the findings of the previous studies and crucially adds an additional layer of learner ownership, personalisation and purpose. In the Robot Dance and Whack a Mole studies, learners had been constrained to solve a puzzle devised by the educator. In study IV, Digital Makers, design decisions were intended to make the product less constrained for the learners. This made it possible for learners to apply their newly acquired programming skills to solve a problem of their own.

In the Robot Dance study, a tight cycle of content delivery and learner consolidation was shown to be effective for a wide range of learners. Whack a Mole attempted to mimic this but increased the learners' control over the pace by using video tutorial, with mixed results. The Digital Maker study built upon the Robot Dance approach with a gradual loosening of the cycle as the session progressed. In the Robot Dance in the Community study, it was shown that if given choice, learners would form different learning groups. This freedom was given in the Digital Makers study for the second part of the workshop. In contrast to previous studies, Digital Makers was a day-long event, giving much more space for learners to acquire skills and then apply them. It aims to address the research question:

(Q4) How do personalisation, ownership and purpose in an activity affect introductory programming learning?

Background

The Digital Makers study was part of One Day Digital, a series of digital making events organised by Nesta Scotland (2014). The aim of the Nesta event was to give 400 young people a taste of digital making. A range of five workshops was assembled, with topics including programming, stop motion animation, web making and games programming. The events ran on four consecutive weekends in Dundee, Aberdeen, Glasgow and Edinburgh respectively. Each event had five one-day workshops on offer to young people, starting at 10:00 a.m. and concluding at 5:00 pm. The events were advertised widely and participants had to register for the workshops in advance. Digital Makers was one of the five workshops delivered at each venue.

Physical Computing

Physical Computing is the construction of a digital device that uses a range of physical input and output components. The Arduino has emerged as the dominant microcontroller in the field of physical computing. Originally, the Arduino was developed to assist interaction design learners to build prototypes with digital functionality. From this beginning, Arduino has grown to be a very powerful and accessible physical computing board to work with. One of the strengths of Arduino is the fact that the project has both open source software and hardware. This has resulted in a vast and varied community of learners, makers and professionals building a wide array of things (Banzi, 2012).

Building physical computing projects requires a range of interrelated core skills including electronics, craft and computer programming. Each of the core skills mentioned share some common features. They all have elements of design, creativity and problem solving and they often have a specific notation to support needed specification of designs and code. These shared features offer some interesting opportunities for cross-domain learning. For example, if the project is an alarm clock, there is a need to specify the electrical components to be used and their relation to each other: buttons, display and audio output. There are formal notations for this task and at some

point there must be a transition from a circuit diagram to a physical layout such as a printed circuit board. In terms of the software, there is a need to specify the various functions of the alarm clock the end user can perform: set an alarm time, turn an alarm on and off, and so on. Various notations support the transition of a software design task from conceptual design through to the final notation of source code. The same is true for the physical product: the materials used, the form and the sequence of interaction all add to its design language, which can be captured in various notations, such as sketching, storyboards and mood boards. Each of these elements must come together for the alarm clock to function and fit the needs of the intended user.

This range of desirable competencies affords a degree of flexibility to the individual learner's experience. Some projects may have very well established design features that demonstrate good understanding of the problem and the user who will engage in the technology. Other projects may be light on design and user consideration, but may be technically sophisticated with extensive electrical or computational ability demonstrated. When a group forms containing individuals with a range of such skills, there is a good opportunity for the learners to see the value of collaboration and varied contributions to a project.

Physical App

A physical app is essentially a tangible equivalent of a mobile device app. In contrast to the burgeoning functions and features found in modern desktop applications, the mobile app is a much more compact piece of software. It often delivers just one item of functionality, but does so very well. The app model is centred on the concept that an app will provide the user with a single well-defined function that is a solution to a single problem, such as a bus timetable or a client for a social network. The user can then assemble a highly personal range of apps to fit their own individual needs. One key advantage of the app is its simplicity: reducing the functionality simplifies the interaction and user experience.

A physical app aims to achieve all the qualities of its mobile counterpart. It should be a simple compact physical computing object. It should solve a single well-defined problem. This naturally lends itself to educational workshop activities with a truncated time for completion. Rather than attempting a complex multifaceted project, the aim is a single purpose project that is buildable in as short a time as possible. In contrast to a vertical prototype, which is a technical demonstrator for part of a system, the physical app stands on its own. This is important so that the learner has the opportunity to experience the full development cycle from idea, design, build and test, to demonstrate.

Description

There were two parts to the study. In the morning the learners were walked through the process of wiring and programming some components with their Arduino; for this stage learners worked as individuals. In the second stage, learners were given the chance to self-select groupings and build a physical app utilising the morning's teaching. The study ended by giving all groupings an opportunity to share their idea and resultant physical app with the whole group. These stages are now described in detail.

Morning: Laying the Foundations

The learners who attended these events came from a relatively large geographical area. There was therefore a good chance that individuals would not know each other. For this reason, the first activity was planned to 'break the ice' and set the scene for the day of making, sharing and appraising the work of their peers. A volunteer was sought from the group; the volunteer was placed at one end of an open space with his/her back to a small box. The rest of the learners gathered round. The volunteer was then instructed to throw small soft balls blindly over their head in the rough direction of the box. The rest of the group was encouraged to offer advice and direction on what the volunteer must do to get closer to getting a ball in the box. The group was coached as necessary. When a ball finally made it into the box (which on all occasions it did), a

discussion was facilitated with the group. Example discussions related to how the volunteer felt when a little stressed and on the spot, and what was found to be helpful, such as general support ("you are doing well") or specific feedback ("angle is good but a little more power"). This was used to highlight that demonstrating something you have made to a group can be stressful and make you feel a little exposed. This stress can be alleviated, however, in a supportive environment. The other important aspect is that when someone suggests a change to a piece of work or action, it is not necessarily a negative thing or a criticism of what has been done. Often it indicates they have thought about the problem, reflected on your solution and have identified a possible improvement. This should be taken as positive, and a sign of respect and consideration of your work.

Following the ice-breaker, participants were given the knowledge and understanding pre-test to complete individually. This led into the taught component of the day where wiring and programming of a range of Arduino components was taught. In this study, a facilitator was used instead of prepared video content, primarily because group sizes were small and because the video approach in Whack a Mole clearly was less engaging and flexible than a facilitator. The delivery mode was supported by a document projector and a projection of the programming environment (Figure 7.1).

Figure 7.1: Teaching Set-up in Physical Apps, Glasgow

In small sections, the wiring-up of a component was demonstrated and described, and then carried out by the learners, with individual support as required. This was very informal and small groups allowed a good degree of dialogue between tutor and learner. Following this, the programming of the component was demonstrated and then carried out by the learners. In this iteration of short demonstration followed by enactment by learners, the learners completed the following tasks: making an LED blink, using a potentiometer to control the blink rate and using a button to make the LED blink when pressed. The first set of examples took around 40 minutes to complete.

To change the activity and introduce a creative disruption to the flow of tuition, the participants were then guided through an idea-generation session. Equipped with Post-Its and marker pens, learners were asked to identify three things that make them excited and note them down concisely on the post-it wall. Learners were then encouraged to bring their Post-Its to the front and stick them on a predetermined part of the wall that was visible to the group throughout the day: the 'wall of situations'. The ideas gathered together on the wall served as an information radiator (Sharp et al., 2009) for use later in the day. This process was repeated for things that make them cross and for things that make them stressed. The purpose of this was partly to move the learners out of their seats and force them to change where their attention was placed. The wall of situations also served to condense physical app ideas to form around later that day. Bringing all the ideas together allowed learners to react to each other's experiences and stimulated memories and new ideas.

The learners were then guided through some additional Arduino output devices: servo, speaker and red green blue (RGB) LED. The servo and speaker both offered the opportunity to show learners the examples that are built into the Arduino IDE. In particular, the servo requires an external library; this offered the opportunity to describe how software libraries are used as structuring tools. Having been shown the Arduino examples (which are very accessible and well documented), the learners had a way to explore further capabilities of the equipment they were using after the teaching had concluded.

The final example the learners constructed was a red, green and blue colour mixer. With a single RGB LED and three potentiometers, a physical colour mixer was constructed. This task requires a relationship between the potentiometers and the intensity of the red green and blue component of the LED to be established. The potentiometer provides a value in the range 0 to 1023 and the intensity of the LED output is given a value in the range of 0 to 255. This requires various built-in functions and the use of variables and assignment. In a natural progression, the learners were shown how to group this now quite complex program into a single user defined function and how to alter this so that the colour of LED was specified by three parameters passed to the function. Extending this further and utilising the random function and bringing in some sound with loudspeakers, playing beeps of a program specified tone, the learners created a light and sound show.

Afternoon: Sketch and Build

The afternoon started with the group revisiting the post-it wall of situations that excite, irritate and stress them. Learners were asked to pick several Post-Its they could relate to and expand upon them. The idea of the physical app was then described: a single-purpose object, like a kitchen appliance, which will perform one task or solve one problem well. Finally, the learners were given three hours to build a physical app based on one of the ideas from the selected Post-Its, with support available as required.

The participants were introduced to the technique of storyboarding developed by Disney in the 1930s: creating a series of linked rough sketches that communicate an interaction or chain of events. The storyboard technique was used to support their thinking and help them to articulate their physical app idea. The system they were constructing was inherently interactive and thus hard to capture in a single sketch or diagram. The advantage of storyboarding soon became clear: several designs can be explored in a short space of time without the expense of building and programming them. The act of formalising a sequence of events and interactions can also be

helpful in making a learner's ideas more concrete. After some time was spent developing several storyboards, all storyboards were gathered in. At random, a storyboard was selected from the stack. The learner responsible was encouraged to share what problem s/he was solving and how the design would solve the problem. The storyboard serves as a link to the learner's description. Feedback can then be received from the rest of the learners. This further shares ideas between the group and encourages individual learners to reflect on the work of others.

At this point the learners were encouraged to form groups and attempt to build one of their physical app ideas. The group formation was left entirely in the hands of the learners. If it appeared that an individual was having difficulty getting into a group, however, they would be offered support if they desired it. Throughout the four sessions, a total of 24 physical apps were developed by ten individuals, seven pairs and seven groups. Throughout the building time, learners were left to work independently. The tutor and two additional helpers were available to offer support as groups required it. Their role was clearly defined as one of facilitation. Learners were to pursue their own ideas and facilitators were on hand to support and tune (where necessary) these ideas to fit within the confines of the workshop, the equipment available and the available time. For example, one group was keen to make a model police car as a toy for a younger brother. Initially the team hoped to make a car that could drive about. The facilitator talked through the technical challenges that this presented and re-focused the team on more achievable static model with lights and sound. The key to facilitation is including the learners in the decision-making process.

To allow participants to construct convincing prototypes, they were given access to various craft materials including balsa wood, modelling clay, foam board, hot glue, various marker pens and a selection of card. Many of the physical apps made use of these materials to embed the technology in a physical form. When the build was concluded, the learners were asked to complete the knowledge and understanding post-test.

The final activity involved getting the individuals, pairs and groups to share their physical apps. Given the time and materials, the physical apps were best described as working prototypes held together with tape, hot glue and blue tac with an Arduino at the core. Across the four sessions, the learners explained very well what they had made and why. This was well received by the other learners. Before the workshop ended, participants were asked to complete the emotional response questionnaire, making a first pass to indicate what emotions had cropped up throughout the day and then going back to offer some contextual detail.

Study Design

A combination of quantitative and qualitative methods was employed to evaluate this workshop. Questionnaires were designed to measure changes in knowledge and understanding, and emotional response. In addition to the questionnaires, digital photographs and videos of each of the completed physical apps were captured for subsequent review and analysis. The following section describes in detail the study design decisions made for each of the following parts of the study: measuring change in knowledge and understanding, measuring emotional response and understanding the sophistication of the physical apps.

As this study involved human participants, ethical approval was sought and granted by the School of Computing Ethics committee. As part of the digital making initiative organised by Nesta, the study did not present any significant ethical dilemmas: it was not replacing or affecting statutory education; it involved neither working with vulnerable participants nor misleading participants. Participants were recruited by Nesta, which also enforced the only exclusion criteria, which was that participants were over 16. The study design used a variety of pre- and post-measures and participant observations that were considered appropriate and in keeping with the inquiry described to the Ethics Committee. Informed consent was obtained from all participants, as was approval for use of digital photographs and videos for research purposes.

Knowledge and Understanding

The basic measurement of this part of the study is to investigate if a change in knowledge and understanding (KU) has taken place in the individual learners as a direct result of taking part in the activities involved in making a physical app. A pre/post-test method was adopted to obtain a baseline of learner KU and then a post-test of KU was used at the conclusion of the study to determine any alteration from the baseline.

The only inclusion criteria and prior information about the learners was their age and the fact they had self-identified an interest in the workshop, based on the event description. Appraising programming competence is challenging. A range of aspects of programming can be language-specific such as keywords, built-in functions or structural elements of a language. In contrast, there is a range of language-independent concepts such as loops, decisions and assignments, which are elemental features of computation. The programming concepts are often enacted by the keywords and language specifics, which introduces some potential for confusion. For example, a `for` loop or a fixed loop are potentially synonymous depending on the language. This is compounded by what du Boulay (1986) identified as Orientation. Programming is a commonplace term that can be interpreted differently by different people. As an example, some may describe competence with HTML, Scratch or Java as programming competence, but each of these includes arguably quite different skills. For this reason and in the absence of any standardised test of programming competencies, a pre/post-test design was selected, as it can be tailored to measure programming ability in areas that are of interest to the study. The design of this test built upon lessons learned from previous studies in which aspects of a pre/post-test approach had been problematic.

The questionnaire designed contains eight multiple-choice questions: four are knowledge-related and four pertain to understanding (Appendix IV). The learner was presented with a statement demonstrating an element of programming knowledge or understanding, and given the

opportunity to indicate whether it is true or false. In addition, the opportunity to indicate they do not know was also given as a strategy to reduce guessing by learners when knowledge is not present. When scoring answers, a learner receives a point for a correct answer, loses a point for an incorrect answer and receives no point for selecting not sure. This was to disambiguate between misplaced knowledge and absent knowledge (Perkins and Martin, 1986). This also offers the ability to see where a learner has transitioned from no knowledge to correct knowledge and when a learner has transitioned from incorrect knowledge to correct knowledge.

Emotional Response

In the Digital Maker study, the emotional response of the learner was measured once again using the Reflective Emotional Index that had been developed for study III, Whack a Mole. As before, the REI encourages learners at the conclusion of a learning experience to reflect and identify emotions they may have experienced and the intensity of each of these experienced emotions. In addition, space for free text response is given to offer insight as to why and when a particular emotion was felt by the learner (Appendix V). A detailed description of this measure is provided in Chapter 6. This is an important part of the physical app study as it offers an insight into the learner's response to the activity. Considering all four studies in this thesis, the Digital Makers physical app study is the one of greatest duration and it offers learners the greatest degree of freedom in what they are building and working towards. The learner's emotional response to the activity is a good indicator of the value of this freedom.

Review of the Physical App

At the conclusion of each session, each group presented its completed physical app, describing what it was, whom it was for and what problem it aimed to solve. These presentations were videoed to provide a persistent record of each of the physical apps created. Videos then were reviewed and a qualitative analysis performed to derive an understanding of the sophistication of the different builds using a card sort system. The previous measures, knowledge and

understanding and emotional response, all relate to the learner's internal experience of the activities associated with creating a physical app. A further measure, 'sophistication', scrutinises the product of the learner's efforts with awareness of the context for which the learners are creating the app. In many ways, 'sophistication' captures the extent to which learners have embraced learning to program in a rich context as well as being able to use the taught skills successfully. As a measure, it considers three elements: (i) does the idea consider the context and for whom the app was being built, (ii) does the build have a strong aesthetic and (iii) is the physical app is technically complex.

Results

The physical apps study engaged 48 young learners, with a range of experience and different backgrounds from across Scotland. There was a substantial gender bias, with the majority (83%) of the participants being male. The following sections present the findings from the three areas of inquiry described in the study design: measuring change in knowledge and understanding, measuring emotional response and judging the sophistication of the physical app. Finally, the composition of groups will be considered with respect to the sophistication of the physical app.

Knowledge and Understanding

Knowledge and understanding are perhaps the most measured outputs from a learning experience, and for good reasons. In the simplest sense, the purpose of creating and engaging in a learning experience is to increase knowledge and understanding of a given topic. Thus, the success of a given learning experience is reflected well by an improvement in the learner's knowledge and understanding. In the physical apps study, change in knowledge and understanding was measured by administering the same eight question multiple choice paper test at both the beginning and the end of the study. Of the eight questions in the questionnaire, four related to programming knowledge and four to programming understanding.

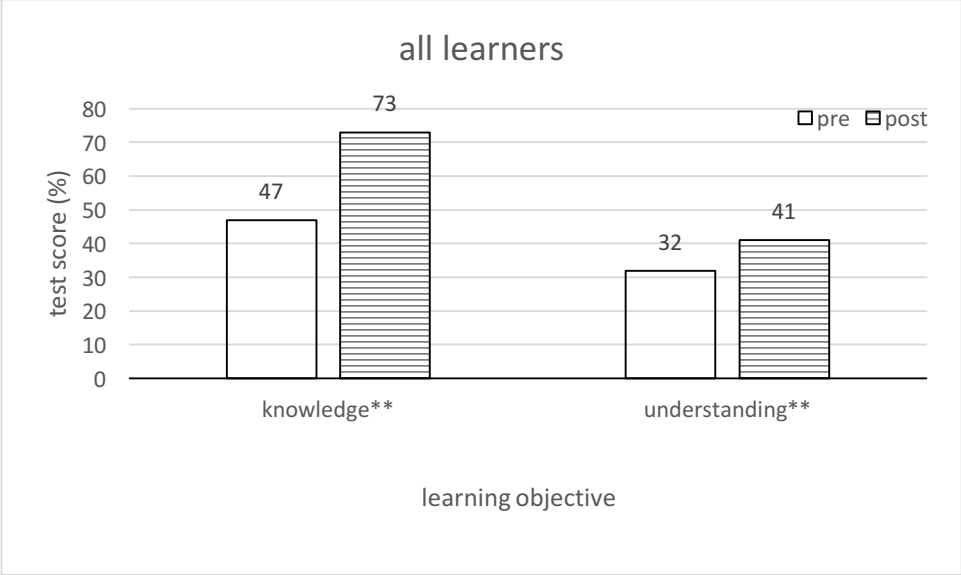


Figure 7.2: Results for Knowledge and Understanding: Whole Sample

Mean knowledge scores improved by 26% and mean understanding test scores improved by 9% (Figure 7.2). The mean post-test score (knowledge and understanding combined) showed an improvement of 34% as a result of the activities undertaken. This is evidence that programming knowledge and understanding has increased significantly because of the physical apps study. Looking at individual questions offers a greater insight into these findings and particular areas of change.

Table 7.1: Digital Makers Results

	Pre-test		**Post-test**		**n**	**t**	**Df**	**Sig. (2-tailed)**
	Mean	**Std. Deviation**	**Mean**	**Std. Deviation**				
Whole sample	39.52	25.09	56.73	21.24	39	-5.05	38	< 0.05
Knowledge	46.79	34.97	72.79	25.12	39	-4.89	38	< 0.05
Understanding	32.05	22.90	41.47	26.49	39	-2.65	38	< 0.05

Figure 7.3 depicts the change of learners' answers from pre-test to post-test. The key is as follows:-

i. The letter below each question number indicates if the question tested knowledge or understanding.

ii. The correct answer is circled.

iii. The *do not know* option in all cases is labelled *d*.

iv. The other answers are distractors.

v. A labelled arrow indicates a change from pre-test answer to post-test answer. Labels indicate the percentage of the class that changed their answer in this way.

This visualisation provides a quick overview of test performance and the changes resulting from the workshop. For example, in questions 1 and 6 there was no change towards the correct answer. In contrast, in question 3, there is a change from *do not know* (d) to the correct answer (b), as well as also a small change to one of the distractors (a). In question 1, there was a decrease in learner uncertainty. However, this was distributed between the two distractor questions rather a change to the correct answer.

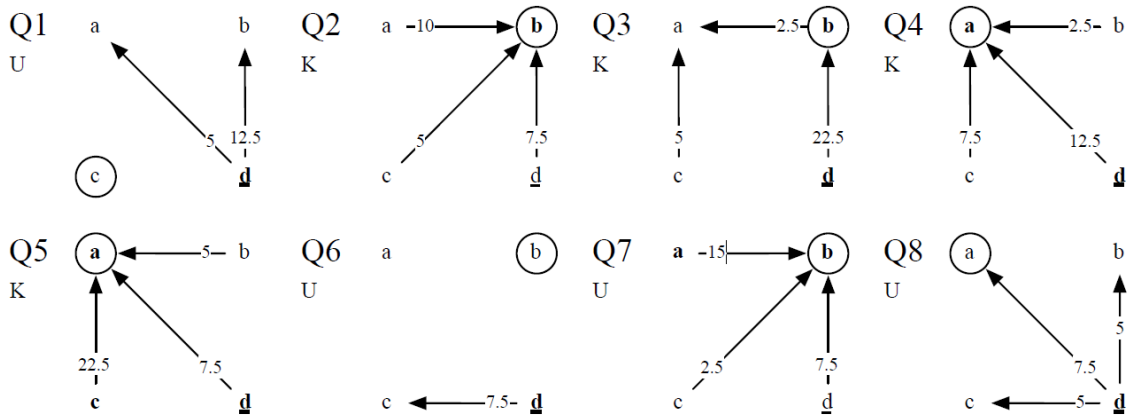


Figure 7.3: Pre- to Post-test Learner Choice Change (% of population)

In Questions 2, 4, 5 and 7 the number of learners selecting the distractors or not knowing the correct answer decreased, resulting in a substantial increase in the number of learners demonstrating correct knowledge or understanding. The question and answer choices are now presented in the tables that follow, with the correct answers highlighted.

Question 1 relates to learner understanding of the assignment operator and variable mutation (Table 7.2). Answer (b) was the most popular response with 60% of participants selecting this; following the workshop this rose to 72.5%. This indicates a partial understanding of assignment as a + 10 was interpreted correctly. However, the lack of an assignment in the final statement was not considered by participants. There was a drop in learner uncertainty, although this was distributed between incorrect answers.

Table 7.2: Question 1 KU test

Q1	`int a = 0;`
	`a = a + 10;`
	`a-2;`
	The code above will make what happen.
		UNDERSTANDING
		pre
			post	**diff**

a	The variable a will equal 0.	2.5	7.5	5.0
b	The variable a will equal 8.	60.0	72.5	12.5
c	The variable a will equal 10.	7.5	7.5	0.0
d	Not sure.	30.0	12.0	-17.5

Question 2 tests knowledge of decision syntax and Arduino function returns (Table 7.3). The knowledge of decision is core to programming, and so learners with prior programming experience are likely to perform well in this question. The pre-test response was good with 60% of participants selecting the correct answer. There was an increase in correct responses, which rose post-test by 22.5%. This particular aspect of knowledge was described and enacted by the learners in the workshop. It was achieved through an activity that involved wiring up a button and an LED and using this code to make it behave like a light switch.

Table 7.3 Question 2 KU test

Q2	UNDERSTANDING			

	`int btn = igitalRead(buttonPin);`
	`if (btn == HIGH)`
	`  light on`
	`else`
	`  light off`
	The code above will make what happen.
		pre	**post**	**diff**

a	A light will come on when the button is pressed and stay on.	17.5	7.5	-10.0
b	A light will come on when the button is pressed and go off when it is released.	60.0	82.5	22.5
c	A light will flash when a button is pressed.	10.0	5.0	-5.0
d	Not sure.	12.5	5.0	-7.5

Question 3 tests knowledge of reading analogue sensors in Arduino (Table 7.4). There was a drop in learner uncertainty, with a quarter more of learners becoming confident to present an answer. The number of participants selecting the correct answer increased by 20%. This knowledge was described and enacted throughout the workshop. Participants used analogue sensors in a number of the examples they built and tested throughout the morning session.

Table 7.4: Question 3 KU test

Q3	`int val = analogyRead(0);`	KNOWLEDGE		
	What will val contain now?	**pre**	**post**	**diff**
a	Nothing.	2.5	10.0	7.5
b	The value of the analogue pin 0.	47.5	67.5	20.0
c	**0**	12.5	7.5	-5.0
d	Not sure.	37.5	15.0	-22.5

Question 4 tested the learner's knowledge of user-defined functions (Table 7.5). Functions are used in many programming languages so learners are likely to have experienced them if they have previously programmed. For this reason, previous programming experience is likely to result in a good pre-test score. Performance on this question was good. There was a 22.5% increase in

learners selecting the correct answer. This was described and enacted by learners in the workshop, but not to the same level of detail as other elements.

Table 7.5: Question 4 KU test

Q4	What is the purpose of a user defined function?	KNOWLEDGE		
		pre	**post**	**diff**
a	It contains a chunk of code you may wish to reuse several times.	37.5	60	22.5
b	It makes the program function correctly.	22.5	20	-2.5
c	It is a built in feature of the language we need to use.	15	7.5	-7.5
d	Not sure.	25	12.5	-15.5

Question 5 tested knowledge of syntax for the branching or decision-making (Table 7.6). Prior programming experience would assist with this element of knowledge. Decision-making is one of the core programming competencies that was delivered and enacted by the learners at various points in the workshop. Learners demonstrated a 35% improvement here.

Table 7.6: Question 5 KU test

Q5	**Suppose we want to branch or make a decision in our program we would use the following statement:**	KNOWLEDGE		
		pre	**post**	**diff**
a	if	45.0	80.0	35.0
b	when	10.0	5.0	-5.0
c	branch	27.5	5.0	-22.5
d	not sure	17.5	10.0	-7.5

Question 6 tested the learner's understanding of fixed loops (Table 7.7). This concept is common in programming languages and, as a result, prior programming experience would be an advantage. This concept was delivered and enacted by learners as part of the workshop. There was a decrease in learner uncertainty; however, the learners showed no change in identifying the correct response.

Table 7.7: Question 6 KU test

Q6		UNDERSTANDING		

	`for(int i=0; i <10; i++){`
	`light on`
	`delay`
	`light off`
	`delay`
	`}`
	The code above will make what happen?
		pre	**post**	**diff**

a	A light will flash.	17.5	17.5	0.0
b	A light will flash 10 times.	45.0	45.0	0.0
c	A light will flash 9 times.	20.0	27.5	7.5
d	Not sure.	17.5	10.0	-7.5

Question 7 tested the extent to which learners understood some of the motives of making use of user defined functions to improve the structure of their code (Table 7.8). There was an increase of 27.5% in the number of learners identifying the correct answer. There was also a decrease in the number of learners incorrectly selecting answer (a).

Table 7.8: Question 7 KU test

Q7	**Why is it a good idea to use user defined functions?**	UNDERSTANDING		
		pre	**post**	**diff**
a	It lets you write more code.	20.0	5.0	-15.0
b	It makes your main code easier to read.	37.5	65.0	27.5
c	It will reduce the chance of wiring errors.	17.5	15.0	-2.5
d	Not sure.	22.5	15.0	-7.5

Question 8 tested learner's understanding of using a conditional loop to halt the execution of an Arduino program (Table 7.8). All Arduino programs are centred on an infinite loop of execution; this is useful as most programs are control response based. It is useful to have a technique to halt this. The question deliberately offers two correct answers: (a) and (b). Answer (b) is intuitively correct, as the behaviour observed by the learner will be execution of the program stopping.

However, the learner with a deeper understanding will correctly identify that this occurs because of an infinite loop. There was a small change, with 7.5% of learners shifting their selection to the correct answer. The level of learner uncertainty decreased by 17.5%, although this was distributed across all answers.

Table 7.9: Question 7 KU test

Q8	`while(TRUE){}` **The code above will make what happen.**	UNDERSTANDING **pre**	**post**	**diff**
a	The code will loop infinitely.	40.0	47.5	7.5
b	The program will stop running.	2.5	7.5	5.0
c	The program will pause for a fixed period of time.	10.0	15.0	5.0
d	Not sure	47.5	30.0	-17.5

In summary, there is good evidence that the workshop resulted in changes in learner knowledge and understanding. The details of assignment and fixed loops remain problematic. In particular, questions 2, 4, 5 and 7 saw the participants all moving towards the correct answers, with a negative change across learner uncertainty and incorrect answers. The decision questions received the two highest proportions of correct answers (82% and 80%). Learners appear to have grasped this concept well by building a light switch and thereby rendering physical the concept of branching.

Question 2 was supported in the learning experience by giving learners hands-on experience of developing user-defined functions to make lights operate based on the use of a switch. Question 4 related to a high-level understanding of some advantages of user-defined functions to structure code. This was supported in the workshop by the development of an RGB LED colour mixer. As the learners worked through this activity, there was a natural desire to compartmentalise and reuse the chunk of code responsible of setting the colour of the LED. Question 5 relates to a learners' understanding of decision-making in programs. This topic was covered in several parts of the workshop, including creating a light switch and setting a threshold for a light sensor. Q7 probes again for learners' high-level understanding of user-defined functions. This was supported in the

study and the good performance in the question indicates a high-level skill that is not typically attributed to novice programmers.

Emotional Response

Emotional response was captured with the Reflective Emotional Inventory based on the EARL (Petta et al., 2011). As in study III, the responses have been grouped via ten high level headings. The first five are negative emotions and the second five are positive emotions. The responses have been collated and normalised to an intensity range from 0 (emotion not felt) to 2 (emotion felt intensely) (Figure 7.4). The most striking result is that positive emotions were reported as far more intensely experienced than negative emotions.

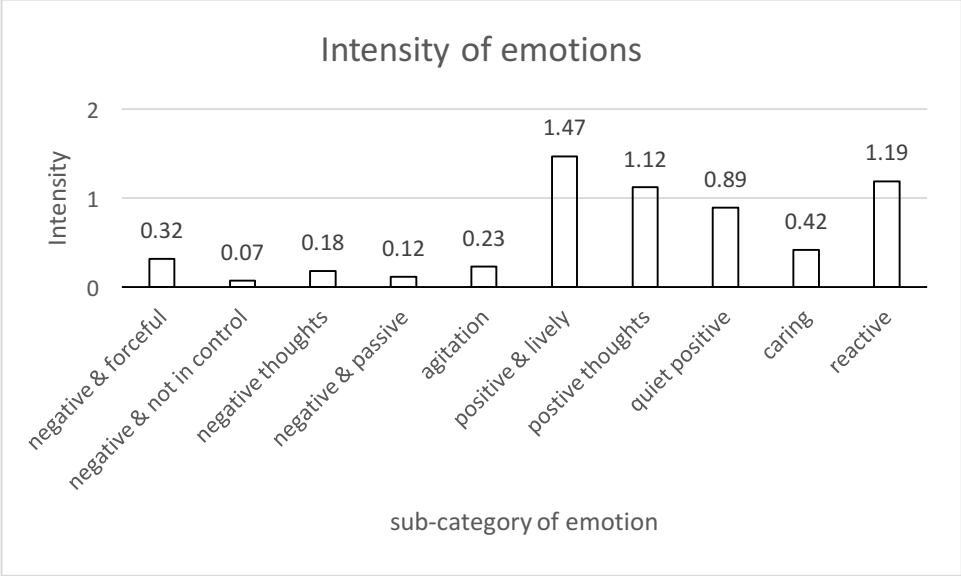


Figure 7.4: Strength of Emotional Responses: Digital Makers

Different sub-categories of emotional response are considered next, beginning with the negative set and concluding with the positive set.

Negative and forceful (0.32) emotions were not strongly reported. The most prevalent emotion in this category was annoyance, which was often associated with the "program not working". One

participant noted "missing a bracket", which is a very typical error with C-style syntax because the language relies heavily on brackets to group chunks of code for logical control structures and user-defined functions. *Negative and not in control* (0.07) was not strongly reported, with no participants providing any contextual comments, although a strong feeling of embarrassment was indicated by one participant. It is perhaps surprising this was not reported more widely given the various points in the workshop that required participants to share their work with the group, which is a process that people often dislike. *Negative thoughts* (0.18) were not reported strongly though the most prevalent were feelings of doubt (0.40) and frustration (0.35). The reason for feeling doubt was frequently cited as uncertainty about whether their physical app would function or whether they would manage to complete it on time, Kate: "not sure we would finish on time". *Negative and passive* emotions were not reported strongly (0.12), several participants note feeling boredom (0.25) for reasons including Alan: "we were finished first and had to wait" and having prior knowledge of Arduino, Jon: "know lots of this already".

Positive and lively (1.47) emotions were reported strongly. Reasons cited for feeling amusement (1.77) and delight (1.77) included: "doing something new", "programming the whole day" and "learning new stuff". *Positive thoughts* (1.12) were also reported strongly. The dominant emotion included in this sub-category was a feeling of pride (1.30). Many participants linked this feeling to the building of a physical app that worked. Example statements include Kate: "When it worked well", Jon: "proud of what we had produced". *Quiet positive* (0.89) emotions were consistently reported by learners. Learners reported feeling calm (1.10), contented (1.00) and relaxed (1.10). *Caring* (0.42) emotions were not strongly reported. The most reported emotion in this subcategory was friendliness (1.00). This was linked to "speaking to new people" and "towards my team mates". *Reactive* (1.19) emotions were moderately reported. The most strongly reported emotion in this sub-category was interest (1.53), with a range of reasons given including "Arduino go!", "I learned a lot" and "doing interesting stuff".

It is clear that the physical apps session evoked a rich emotional response from the participants. The contrast between the extent to which negative and positive emotions were experienced is strong. Negative emotions experienced do tie into many of the problems reported in the literature about novice programming. Interestingly, many of the error-prone features of coding do match with those of physical prototyping, with breadboarding being particularly error-prone. Nonetheless, the minor irritations of an error-prone medium are outweighed by the strength of the positive emotions reported by learners. Many positive emotions stem from a sense of overcoming challenges to produce something that works and it is important to note each individual, group or pair did succeed in producing something that they were able to present to the group. The strength with which learners expressed a sense of pride relates well to the cultural impact, giving evidence that participants were interested and proud of their work.

Review of the Physical App

Across the four sessions, 24 different physical apps were developed, responding to a wide variety of ideas. Each of the apps built was considered for the elements of idea, build and complexity, build and idea. The elements were judged using a structured framework that involved rank ordering the apps for each element and identifying different levels in each sorted set according to the characteristics of that category. Aspects of the elements are considered next, followed by summary descriptions of the different category levels within each element and examples of the apps produced.

Idea: the idea sort orders for the quality of the physical app idea and its purpose. The very best in this element will have demonstrated consideration of the context for the physical app: is it on a bedside cabinet, in a kitchen or in a train station? Evidence of considering a specific user and beginning to understand his/her needs is also an important factor in having a well-resolved idea. A strong idea requires the app to have a well-defined sequence of interaction. The weaker apps in

this sort are likely to be driven by technical expertise and are simply an assemblage of the learner's newly acquired technical skills, with little thought of purpose or for users.

Build: the build sort will be an order based on the non-technical physical components of the physical app and the effort given to construction. This element will consider the materials used, the aesthetics of the app, the structures and mechanisms created. The very best in this element will have encased the electronics of their physical app to give the appearance of a visually considered prototype and made appropriate choices about materials and construction methods. They will have demonstrated a good degree of thought and design in model constructions. Making good use of the available materials is also indicative of a strong build. The middle level apps of this element will have achieved either creation of an aesthetic model or demonstrated good use of materials or structures. The bottom level apps of this element will have failed to function or will have not augmented the electronics with any modelling at all.

Technical Complexity: the technical complexity sort refers to the complexity of the build with respect to the code and hardware components used (e.g. servos, LEDs and buttons). The best in this element will have a well-integrated combination of the taught examples. The very best will also contain an element of novelty or an extension of the taught concepts. The middle level of this element will have apps that have robustly reproduced more than one example in an interesting context. The bottom level of this element will include apps that are simply a duplication of a single example such as the servo sweep. There may also be a number of apps that are not completed and thus not working.

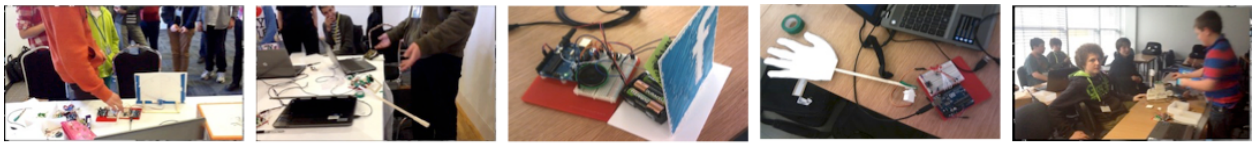
Idea

The idea sort categorises apps based on sense of clear purpose, solid sequence of interaction and identification of target user and context. The apps were arranged in a continuum that was iteratively reviewed with each app's idea attributes considered against its neighbours, and moves made where appropriate. Once the apps were ordered into six separate categories, a summary of

was produced of the characteristics that describe the apps in each category (Table 7.10). The apps in each of the six idea categories are now presented, beginning with Category 1 (Figure 7.5) which signifies the strongest apps, followed down to Category 6, which are the weakest.

Table 7.10: Idea Sort

Category	Characteristics
1	user identified, of clear purpose, solid sequence of interaction
2	user consideration, interaction sequence
3	user considered
4	idea weakly linked to brief, little consideration of user
5	weakly linked to brief, no consideration of user
6	no context or back story



(16) mmff (23) m (4) m (1) m (9) mmm

Figure 7.5: Idea Category 1

The top performers (Figure 7.5) demonstrated a robust acknowledgement of the brief: to build a physical app based on one of the ideas Post-Its from earlier. Inclusion in Category 1 was defined by evidence of being derived from a problem rather than a technical capability and by being a novel idea with strong links to specific users and/or context with a resolved sequence of interactions. App 16 was a homework progress monitor with a clear consideration of user and good resonance with the brief. This also solved a problem the learners personally identify with. App 23 was a physical alarm clock that starts with a small noise that gradually increases in intensity, then progresses to a physical arm that bashes the sleepy person until s/he wakes. This was a response to the challenge of getting out of bed for school after the holidays, put on a post-it in the morning.

App 4 provided a single purpose audio-visual notification of a Facebook notification. This is in keeping with what they had been asked to do and demonstrates the potential need for more eye-catching notification. App 1 provided a performance-rewarding app that quite literally gave you a pat on the back and also kept note of it with an LED bar graph. The final app in this category, App 9, was titled the FIFA notification station. It has a moving physical arm that draws attention to new information pertinent to a computer game that was popular among the group. The learners spent a great deal of time choosing specific items of information and mocking up the visual display with marker pens on the foam core board. The common feature across all the Category 1 apps is that they are driven by an idea rather than technology. The physical app tells a story of who it was for and why it was for them. In most cases the user was the learner, which meant they were building something that related to them whilst they learn about physical computing and what it entails.

Figure 7.6: Idea Category 2

The Category 2 apps (Figure 7.6) were defined as having a good idea, with consideration of some users and interaction sequence. App 19 was a spin the bottle game, which had a resolved interaction sequence, with a button press to initiate, a pause and audio alerts. This appeared to be built around a technical capability rather than being a response to a problem from the morning. App 17 was a multi-button tone generator intended as a keyboard game. This is a well-defined idea with resolved chain of interaction and user consideration. App 15 was a catapult game. This group had spent some time developing the narrative of the app and considering how users may interact and play with it. All Category 2 apps demonstrated resolved ideas that were relevant to

the learners but tended to be more driven by technology rather than solving a problem defined by the group.

Figure 7.7: Idea Category 3

Category 3 apps (Figure 7.7) were defined by having a good idea with some consideration of users. App 14 was a game that involved jumping over a moving stick; as time progresses, the speed of the stick increases. App 10 was an alarm clock that uses flashing lights to waken the user. App 24 was a catapult game. App 17 was a drawing machine based on the activity of an aunt's knitting. App 8 was a spin-the-bottle game. All apps in this category lacked detail, consideration of what the app was and the sequence of interaction. It took some work to encourage the learners to describe these aspects of their apps.

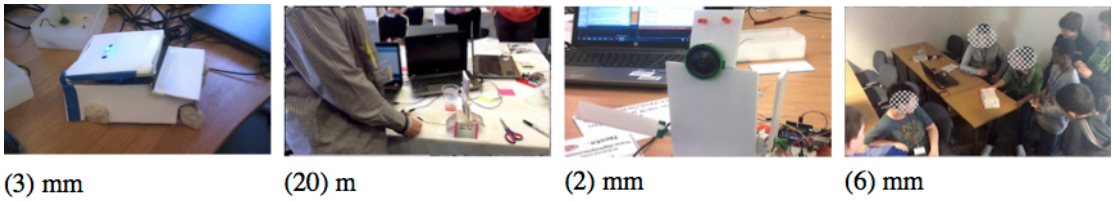
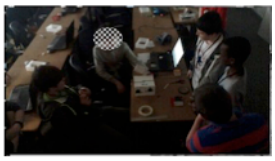


Figure 7.8: Idea Category 4

Category 4 apps (Figure 7.8) were defined by having an idea weakly attached to the brief, with a little consideration of the user. Apps 3 and 6 were toy police cars with flashing lights and sound. App 20 was a spinning sign that rotated and App 2 was a robot with moving arms, flashing LED eyes and audio tone output. All the apps in Category 4 were driven by technical capabilities, but there was little evidence of them fitting the needs of a specific user.

(5) mmm



(11) mmm

Figure 7.9: Idea Category 5

Category 5 apps (Figure 7.9) were defined as having a weak idea with little evidence of any user consideration or sequence of interaction. App 5 was a mood box that presented mood through a range of colours. App 11 was a speaker dock. These ideas did not show a novel, clear and concise purpose that fitted within the brief. The presentation of the apps offered little evidence of working towards a clear goal or solving the problems of a specific user.

(13) mm

(12) m

(18) m

(21) ff

(22) mff

Figure 7.10: Idea Category 6

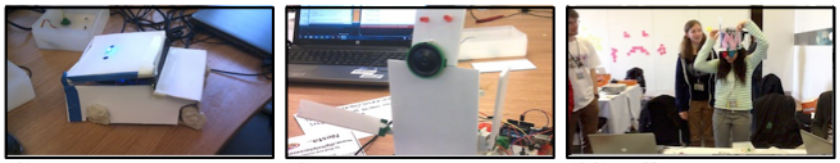
Category 6 apps (Figure 7.10) were defined as being without any idea, back-story or reference to the brief. App 13 was an attempt at making a four-legged walking platform. Apps 12 and 18 were performances of multiple sequenced tones, light effects and servo motion. App 21 was a model helicopter with rotating rotors (and an unsuccessful attempt to include RGB LED light effect). App 22 was a top hat with a light on it. All apps in this category were not on brief and failed to describe their main purpose, thus demonstrating no consideration of a potential target user or context. They were driven by an exploration of the technical capabilities of the Arduino equipment the learners had at their disposal and were unrelated to the ideas generated earlier.

Build

The build sort considers apps in terms of their physical construction. This takes into account use of materials, structures created and the aesthetics. Once the apps were ordered into six separate categories, a description was given to the characteristics of the apps in each category was created (Table 7.11). These apps are now presented with respect to the build criteria. The sequential category numbering indicates the quality: category 1 signifies the strongest apps; category 6 signifies the weakest.

Table 7.11: Build Sort

Category	Characteristics
1	Refined and complex build with good use of a range of materials and consideration of aesthetic
2	Complex build with good use of materials.
3	Refined yet simple build
4	completed simple build
5	basic or incomplete build
6	no attempt to make use of craft materials



(3) mm (2) mm (22) mff

Figure 7.11: Build Category 1

A Category 1 app (Figure 7.11) was defined as a refined and complex build that made good use of materials and had considered aesthetics. App 3 was a police car with light and sound. The learners had spent a great deal of time embedding the electronics in a model that offered a good representation of a police car. They made use of a range of appropriate building materials. App 2, the robot toy, was constructed with an array of components cleverly embedded in a convincing robot model. Where appropriate, such as with the servos, the hardware component was concealed.

Where it could add to the appearance of the build, such as the speaker mouth, it had been made visible. In addition, a novel construction approach was used to create an articulated abutment join with the foam-board. By inserting straightened paper clips, the learners were able to attach appendages with some flexibility.

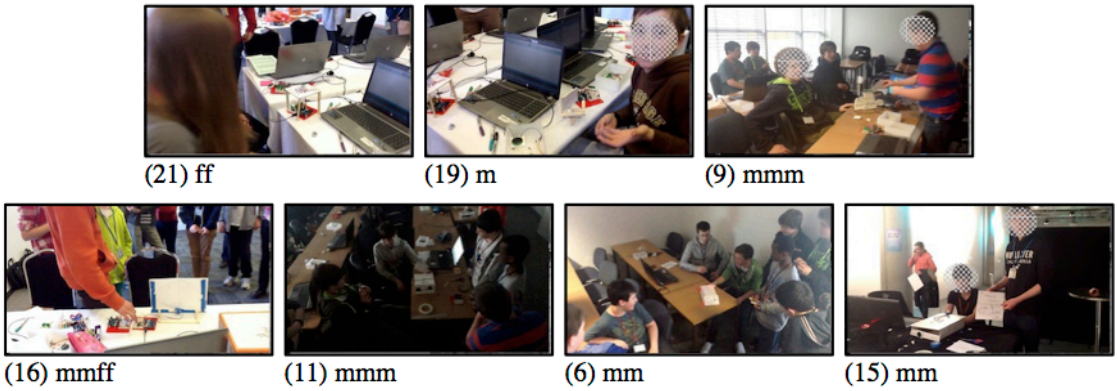


Figure 7.12: Build Category 2

A Category 2 app (Figure 7.12) was defined as a complex build making good use of available materials. App 21 was intended to be a model helicopter. Its form is not like a helicopter but the structure created demonstrated good competence, with hot glue and balsa wood resulting in a 'true' box to house the electronics. It also included model-building work performed to accommodate an RGB LED, and it was decorated. App 19 used a combination of capa board and balsa wood in the construction of a large spinning arrow attached to a servo underneath. App 9 made good use of capa board and marker pens to present a prototype of the way information would be presented to the user if a textual display were available. App 16 used a range of material to produce a visually striking analogue dial. App 11 used capa board to encase the electronics for the model in a box. App 6 used card to hint at the form of a toy police car. App 15 was a crafted capa board box that encased all the electronics for a catapult game. It was also decorated with diagrammatic instructions. All apps in Category 2 used a range of the available materials and indicated that a good attempt at turning the team's idea into a prototype form and embedding the electronics in the app within a model.

(20) m (14) mmff (4) m

Figure 7.13 Build Category 3

A Category 3 app (Figure 7.13) was defined as a simple yet refined build. App 22 was a spinning sign with a number of stickers adorning a well-crafted balsa wood frame. App 14, the jump stick game, had a relatively simple build but important problems had been resolved. The learners had identified that to have a large beam cantilevered, they required an equal force on the other side of the pivot. This was achieved with a large plastic counterbalance weight. App 4 is a good example of a simple yet complete prototype. The learner had accurately captured the Facebook logo and centred the build around this. The LEDs used in the notification were also embedded in the capa board using the card logo to defuse them, giving a very clean line. The apps in this category were all simple but complete and refined.

(1) m (7) m (8) mmmm

Figure 7.14: Build Category 4

A Category 4 app (Figure 7.14) was defined as a complete simple build. All apps in this category essentially revolved around attaching a balsa wood beam to a servo. App 7 had included some triangulation to reduce flexing and ensure the marker pen remained in contact with the card. The apps in this category were evidence that the learners had focused their efforts on aspects of the app other than its physical construction.

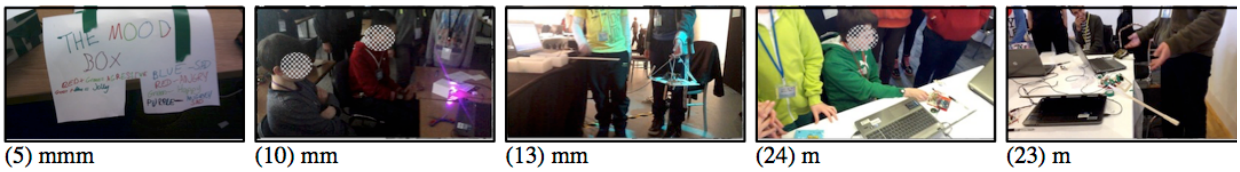


Figure 7.15: Build Category 5

A Category 5 app (Figure 7.15) was defined as a having basic and/or incomplete construction. Apps 5 and 10 comprised partially made boxes. App 13 consisted of four servos on a plate with poorly attached balsa wood legs. App 24 was a catapult but the arm was poorly constructed and failed when demonstrated. App 23 also comprised a balsa wood arm that had several structural flaws. Apps in category 5 offered little evidence of learners having a successful engagement with craft materials. However, all learners in this category had attempted to engage with craft materials.

Figure 7.16: Build Category 6

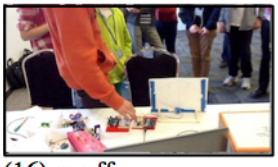
A Category 6 app (Figure 7.16) was defined as an app with no construction attempted. Apps 12 and 18 were created by a single male learner who was proficient with Arduino programming. The light, sound and motion app comprised an assembly of components with no apparent link to embed them in an object. App 17 was also constructed by an older single male learner who had a good degree of proficiency with Arduino. It is possible to argue that with the materials available, there was little scope to improve this app with a model, as embedding switches would have been incredibly challenging without soldering equipment.

Complexity

The complexity sort considers apps in terms of their technical sophistication: how advanced is the code and selected hardware? Once the apps were ordered into six separate emergent categories, a description was created of the characteristics that describe the apps in each category (Table 7.12). The apps in each of the six categories are now presented with respect to the complexity criteria. Category 1 signifies the strongest apps sequentially down to Category 6, which are the weakest.

Table 7.12: Complexity Sort

Category	characteristics
1	complex integration of more than three components
2	complex integration of more than two components
3	effectively integrate two components
4	uses single component from example with extension or adaptation
5	simple app using one of the examples
6	partially functioning example



(16) mmff

Figure 7.17: Complexity Category 1

A Category 1 app (Figure 7.17) was defined as an app that is a complex device integrating multiple components and techniques from the morning's teaching. In addition to this, there must be evidence of demonstrated knowledge beyond taught content. The only app in this category was App 16. This group identified that there were issues with using servos and pulse width modulation at the same time. To combat this, they made use of multiple Arduinos and used serial communication, a technique not taught, to allow the boards to communicate with each other. The app also included multiple components and good integration of a range of taught material all reliably functioning.

Figure 7.18: Complexity Category 2

A Category 2 app (Figure 7.18) was defined as a complex app that successfully integrates more than two of the concepts or hardware taught. App 17 is a keyboard that required the learner to wire up multiple buttons, disambiguate between them and create different tones on the speaker according to which button is pressed. The hardware and software was robustly put together. App 12 integrates all taught aspects of the day. It provides a light, sound and motion display. App 2 was a robot toy. Both arms are moved using two servos that required duplication of the servo object and associated code. The speaker was used to generate random blips and tones and the lights were programmed to flash. Pulse Width Modulation is a signal technique used to simulate an analogue output with a digital square wave. Varying the proportion of time-high to time-low over a fixed wavelength, digital components such as LEDs can be switched very quickly to give the appearance of dimming. This technique was taught to make an RGB LED generate a wide range of colours by varying the intensity of red, green and blue components.

This group successfully integrated all taught aspects. App 1, 'the pat on the back', was a technically complex app. It was button operated: when the button was pressed the arm and hand would swing to simulate a pat on the back. In addition to this, the software would keep count of the number of times the button had been pressed and indicate this via a line of LEDs. All the category two apps were technically sophisticated and required the learners to apply a range of the taught material. In addition, learners had to integrate different devices and demonstrate some ability to generalise the taught material to a novel context.

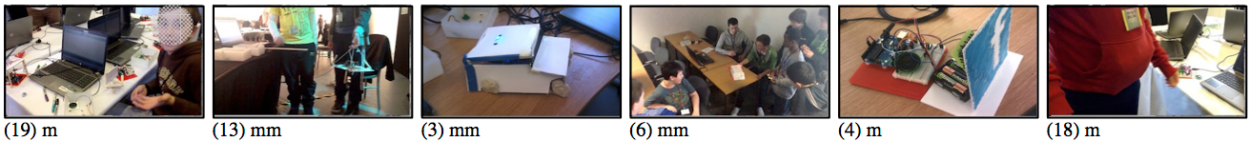


Figure 7.19: Complexity Category 3

A Category 3 app (Figure 7.19) was defined as a complex app that effectively integrates two aspects of the taught material. App 19 was a spin the bottle game, with a button interface. Once the button is pressed, a sequence of tones is played to alert the user; following this, a servo oscillated the arrow for a random period. This learner had demonstrated a good understanding of controlling sequence of execution and using `if` statements and `while` loops to enable pausing of the flow of execution. App 13 was an ambitious attempt to make a quadruped robot. This required the learners to control four servos, which requires four separate objects to be instantiated and used throughout the Arduino sketch. This demonstrates some understanding of object orientation, or at least the ability to form associations between software objects with hardware components. Apps 3 and 6 were toy police cars; both had flashing lights and sound. The learners demonstrated the ability to explore different tone sequences and encode the correct timing of lights and sound to get an authentic emergency vehicle's appearance. App 4 was a physical Facebook notification. It used a combination of lights and sound to alert the user to new status updates on their Facebook. The learner demonstrated the ability to control the flow of execution, waiting until a button is pressed. App 18 was a light and sound show; the learner demonstrated the ability successfully to integrate the sound and light material taught. All apps in Category 3 have demonstrated the ability to integrate and apply two of the taught aspects of the day.

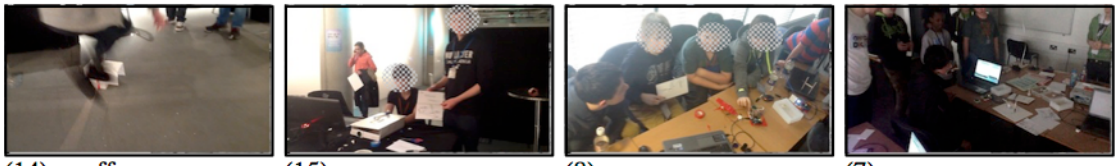


Figure 7.20: Complexity Category 4

A Category 4 app (Figure 7.20) was defined as a moderately complex app that was based on one example from the taught material with some adaptation and extension. App 14 was a jump stick game based on the servo sweep example in the Arduino selection of examples. The learners did extend this example by making the servo swing speed increase as time went on. This demonstrated an understanding of using variables as counters to track the number of iterations of the main loop. App 15 was a catapult game that was based on the 'servo knob' example in which the servo position is related to a potentiometer. There was an attempt also to integrate some sound effects. App 8 was a spin the bottle game that involved a slight extension to the servo sweep example. By embedding the example code within an `if` statement, the pointer would only oscillate if the button was pressed. App 7 was a drawing machine, adapted from the servo sweep example to move to a random position, thus simulating accelerometer data obtained from knitting needles in use. The output was realised by a marker pen on the end of a rod. All apps in category 4 are heavily based on a single example from the taught materials with some tweaks.

A Category 5 app (Figure 7.21) was defined as a simple app that is lifted from a single example, with little adaptation. App 5, the mood box, and App 10, the alarm clock, are based on the RGB LED function example. App 9, the FIFA notification station, is built around the servo sweep example with no additional interaction or extension. App 22 has an illuminating LED with no interaction or adaption. Apps 11, 21 and 20 are built around the servo knob example. All apps in this class are built around one of the class examples with no demonstration of deep understanding or ability to extend or apply learned material to a new context in novel circumstances.

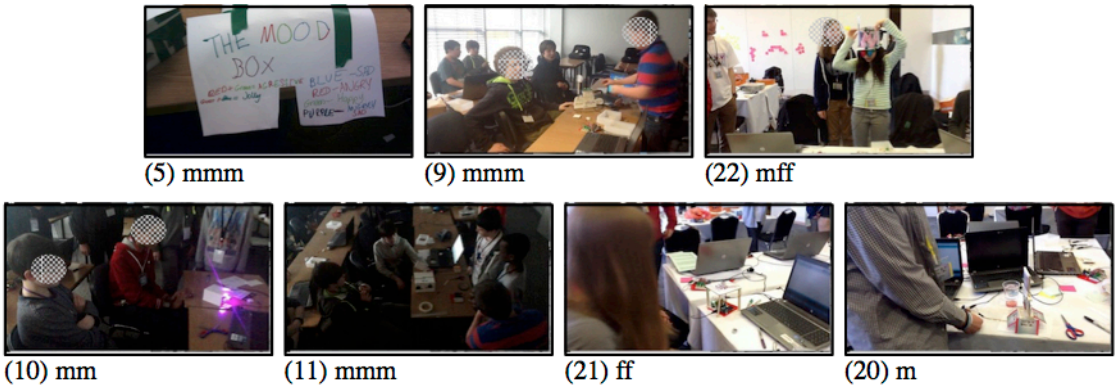


Figure 7.21: Complexity Category 5

A Category 6 app (Figure 7.22) was defined as partially functioning. The learners constructing Apps 23 and 24 were unable to complete their app successfully. With App 24, the learners had identified an interesting idea but did not have the technical ability to construct it. In the case of App 24, the learner did not decide on what to build until late in the session and as a result ran out of time. It is interesting to note, though perhaps not surprising, that the only apps that were not functioning were attempted by individuals.

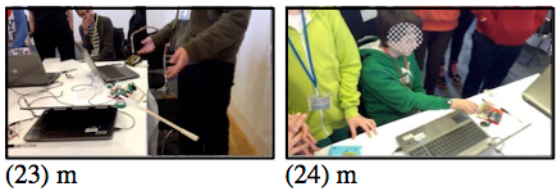


Figure 7.22: Complexity Category 6

Group composition

For the morning taught component of the workshop, learners worked as individuals, with a 1:1 ratio of equipment to learner. In the afternoon, when learners had the opportunity to build a physical app of their own design, they were left to form groups, pairs or work as individuals as they desired. As a result, there was a rich mix of grouping. There were eight female learners and forty male learners across the four sessions. Of those 48, ten learners opted to work as individuals (all male), seven formed pairs (one female and six male), four formed groups of three (one mixed and three male) and three formed groups of four (two mixed and one male). This section considers the group composition with respect to results overall, and separately for **idea**, **build** and **complexity,** to explore whether group composition may have affected any of the card sorts. The groupings found in top half and bottom half of the card sorts are discussed to ascertain whether group size has an impact on each of these elements of app creation.

Figure 7.23 gives the result of considering the group compositions and the sum of the ranked positions for the three card sorts. The leftmost number is the final rank of the physical app, the subscript giving the sum of the app's three positions (low number represents high ranking). Each app is annotated with the group number, the individual rank for each element (idea, complexity and build) and a note of the group's composition (size and gender).

Figure 7.23: Apps Ordered by Summed Card Sorts

Group composition: overall ranking of the physical apps

To be in the top 20% of apps, the learners had to demonstrate excellence across the range of skills and notably all had identified excellent ideas. Of the four in this category, three were individual male learners with the top place going to a mixed group of four. All apps in this band were robustly demonstrable.

The second 20% was dominated by male pairs. In this group, learners had performed well in two out of three skills related to app creation, with one skill tending to be excellent. All apps in this band were complete and demonstrable.

The third 20% comprised a good mix of groupings. Most learners in this band performed well across the range of skills, with one exception. For App 17, the learner had identified an excellent idea and demonstrated a good level of technical competence but had chosen not to embed their build in a suitable model. All apps in this band were demonstrable to the groups.

The fourth 20% comprised a mix of individuals, pairs and groups of three. Notably the developers of apps 22, 23 and 21 excelled in one skill associated with app creation and performed poorly in the other two. In contrast, apps 11 and 20 had similarly reasonable scores across all skills. In this band, most apps were functioning and demonstrable.

The fifth 20% comprised a mix of individuals, pairs and groups of three. Apps 10 and 5 scored poorly across all skills associated with app creation. Apps 12, 13, 24 and 18 have two poor scores and one reasonable score.

With the exception of the top-placed group, small teams and individuals performed best in the physical app workshop overall. Further research with a greater number of learners is required to explore this finding further.

Group composition is considered next in relation to the individual elements of app sophistication.

Group composition: elements of sophistication

Figures 7.24, 7.25 and 7.26 depict, for each size of group size, the proportion of apps that were in the upper and lower halves of the respective card sorts.

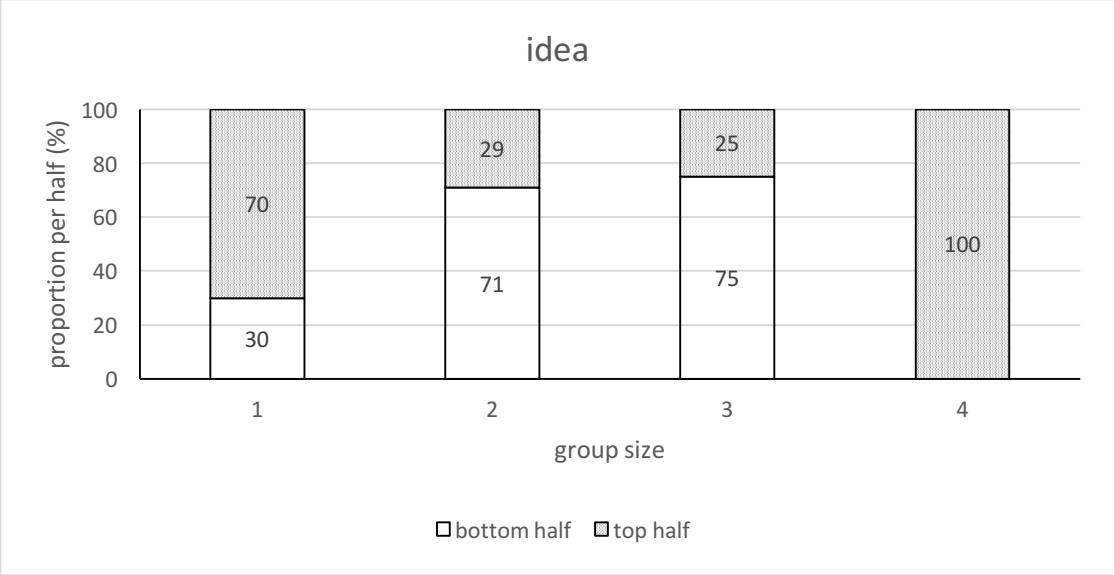


Figure 7.24: Proportion of Groups in Top and Bottom Half: Idea

Figure 7.25: Proportion of Groups in Top and Bottom Half: Build

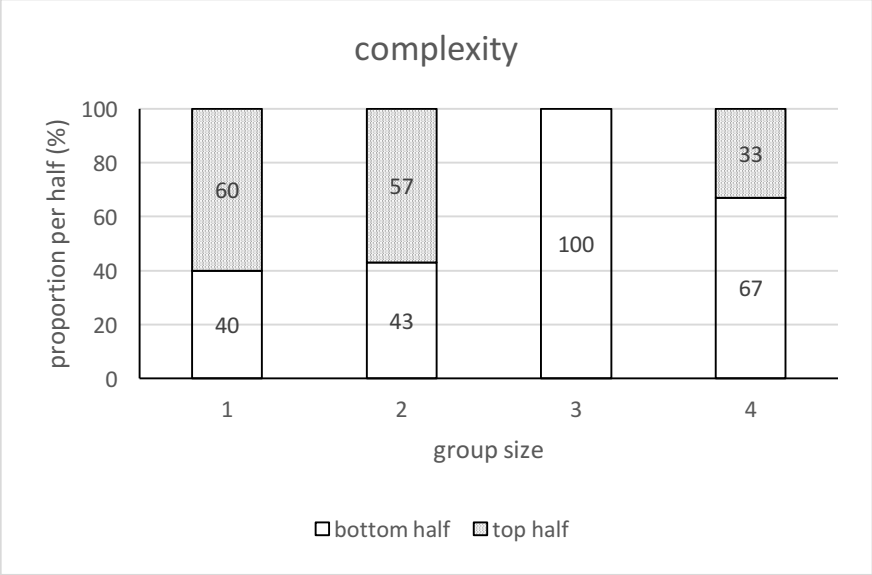


Figure 7.26: Proportion of Groups in Top and Bottom Half: Complexity

Discussion

Knowledge and Understanding

In addition to the high degree of engagement and the emotional response described, there was an observable learning effect. The Digital Makers study resulted in a 35% increase in knowledge and understanding amongst the learners that participated. This goes some way to confirm that the additional task given to learner to flesh out the programming did not obstruct the intended aim of supporting engaging programming learning.

Emotional Response

The emotional response from learners was similar to that of study III, Whack a Mole. Learners reported negative emotions such as frustration related to bugs in code and wiring. As in Whack a Mole, the strength of the positive emotions was notable greater than that of negative emotions. It is proposed that the error-prone nature of programming will always result in frustrations. The skill in designing a learning experience is to *support* this rather than to attempt to remove it. Facilitating

a learner through the resolution of a software or hardware bug sets them up with some skills of value beyond programming. It is also important to ensure learners do not reach a state in which they have invested significant effort without feedback, as this may result in wasted effort and potential damaging frustration. An example would be wiring up five identical components before confirming that the first one is wired up correctly. This is a tricky balance to achieve.

Review of the Physical App

Idea

There was a mix of outcomes regarding how learners engaged with placing their app in context and relating it to a real world problem. The majority of learners made a good attempt to contextualise their creation, many found resonance with the brief to a greater or lesser degree. One of the most interesting findings is that of the 24 apps only Apps 12 and 18 were devoid of any idea, consideration of users or a sense of place in the world. This demonstrates that for the majority of learners the rich real world framing of the task was valuable.

The two apps created with no context were produced by capable programmers who were demonstrating that their interest lay in the technology and they did not desire the distraction of a wider sense of place in the world for their work. Whether this is a good or bad quality in a programmer is a topic for debate elsewhere. There is evidence that the majority of learners engaged well in programming within the bounds of a real world application.

Build

Most of the learners took great pleasure in engaging in the opportunity to use various craft materials as part of the physical app challenge. The top two-thirds of the sort had learners demonstrate the ability to create mechanically intriguing and aesthetic solutions to give their physical app function and form.

Of the bottom third, many learners attempted to engage with the craft materials but lacked the necessary skills to successfully construct the structures they desired. It is important to note that craft skills were not explicitly taught in the workshop, although assistance and advice were provided in response to requests. Only three participants did not attempt to engage with the building materials. All were technically competent programmers who most likely viewed this as a distraction from their primary interests. To return to the central premise of this thesis, the build aspect is to stimulate the learner to take up the challenge of programming. In the case of the three apps where learners chose not to engage with the build aspect of the task, there is little doubt that they avoided the craft as a direct result of their hunger to pursue the challenge of programming with which they had proficiency.

Complexity

Complexity measures the technical competence demonstrated by the learners as they respond to the challenge of developing a physical app. In many respects, this has most in common with the traditional measures of success such as change in knowledge and understanding and the ability to demonstrate what was learned. The top two-thirds of learners were able to demonstrate the ability to apply the morning's tuition in novel circumstance to meet the needs of a self-directed challenge, which is quite an achievement. They mixed and adapted examples to solve their own problems. As for the bottom third, seven out of the nine were able to reproduce examples from the morning and attempt to frame them. Only two learners were unable to produce a working physical app by the end of the workshop. In one case, this was a straight example of an overly ambitious idea, or inability to measure a challenge against one's abilities. The second case is more difficult to comment upon. In a busy teaching environment, it is a challenge to ensure all learners are receiving the attention they need. In the case of App 24, it is possible that this learner was not sufficiently confident to make his voice heard and as a result was unable to receive the support he required.

Group composition

For **idea** generation, individuals performed well with around 70% of apps created by individual learners appearing in the top half of the idea card sort. The larger groups with four participants were also in the top half of the idea card sort. Groups of two and three performed less well in the idea generation stage. Individuals performing well in an idea-generation process perhaps may be unexpected. This may be explained by the large group idea-generation activity that took place before the build. This allowed the whole group to consider and externalise possible app ideas. Individuals may have found it simpler to select one of these ideas, whereas the medium-sized groups spending time negotiating.

For the **build** sort this was flipped, with individuals performing less well. For build, around 70% individual learners were in the bottom half of the build sort. The pairs, trios and quartets all perform well in the build card sort, which is perhaps unexpected. It is likely that a well-functioning group with more members offers a good opportunity to delegate tasks. Building the physical model is also an activity that can take place in parallel to code generation, thus favouring larger groups.

For **complexity,** pairs and individuals perform best with 60% of individuals in the top half of the sort and over 50% of pairs in the top half of the complexity sort. None of the trios and only one quartet made it into the top half of the sort for complexity. This is unsurprising, as many of the more complex apps were produced by technically capable individuals who were focused on exploring the hardware and software. These individuals were less engaged in group work or in placing their hardware in a specific context.

Limitations

There is always a need to apply pragmatism when designing a study involving education. In this study, the single observer issue could be described as a limitation, as in study II. A further

limitation is the possibly untruthful or poorly remembered self-reporting of emotions experienced. However, there is no evidence of any learners reporting their emotions other than as best they could. A stronger candidate for consideration as a limitation is the balance between sample size and sample bias. This study has a good sample size, with a rich set of data from 48 learners across Scotland. However, it does suffer a sample bias as a result of being piggy-backed on to a national public engagement event. It is therefore important to reflect on this when considering the results. The sample was taken from across Scotland for a bookable event. Participants who attended were likely to have a good awareness of what they were attending and therefore likely to engage fully and perform well. This has been evidenced from the various measures taken throughout the study where the majority of learners have performed well. These methods will need to be piloted across mainstream education to test if the results described generalise to the broader population of young learners.

Conclusion

The Digital Makers study used ownership, personalisation and purpose to create a highly engaging learning experience that resulted in a significant learning effect and strong positive emotional responses from learners. The main findings of this study can be summarised by referring to the research question posed:

(Q4) "How is introductory programming learning affected by designing activities that enable personalisation, ownership and purpose?"

The qualitative evaluation of the physical apps gives the best insight into the effect of the rich context that was built around the intended learning. Reflecting on a learner's app, ideas, builds and complexity offers a good indication of the extent to which that learner has embraced this approach to learning. Almost all learners engaged very well with this approach to learning, showing commitment to solving the problems they had defined. The ideas for physical apps

reflected the culture the learners belong to, such as with notifications for FIFA and Facebook. In some cases, they personalised their product by creating toys for younger siblings (police car with lights and sound). Almost all learners attempted to construct some kind of physical app using the craft materials provided. The complexity of the builds varied from relatively simple extensions of the demonstrations to complex compositions with multiple sensors and actuators. The only learners in study IV not to engage fully with the rich context were learners that had come to the session with an existing knowledge of Arduino and had premeditated plans for what they wanted to explore. For these learners, the rich context was a distraction to their intentions.

Built upon the findings of the four complementary studies I-IV, the next chapters consolidate the findings, relating them to the literature and offering advice on applying these findings in a set of Learning Dimensions. The Learning Dimensions will capture the key design features of a learning experience that has a high degree of engagement and ultimately results in successful learning. They are proposed as a first set of important decision areas for the creating of engaging computer programming learning experiences.

Chapter 8: Learning Dimensions Part I. Design and Delivery

Introduction

The Learning Dimensions (LDs) bring together findings from the literature and new empirical work that can guide the creation of engaging learning experiences for computer programming. The aim is to provide a resource for computer science educators that can be used either in the design of new learning experiences or as a reflective toolkit for the review and improvement of existing learning experiences. The Cognitive Dimensions framework by Green and Petre (2006) has served as a very successful nucleus for a great deal of research relating to notations of many forms including code, sketching, algorithm visualisation and musical staff notation. Cognitive Dimensions provided a much-needed common vocabulary that enabled researches to share and discuss insights. It is hoped that the LDs fulfil a similar role for educators in the design and evaluation of learning experiences. As a resource, the LDs are intended to be lightweight, accessible and easy to use. The intention of the LDs is not to present a new pedagogy or theory that tackles all or even most of the aspects of the creation of computer science learning experiences. Instead, the LDs are a set of insights and knowledge from which educators can select to add value and to make informed decisions about their practice. The eight LDs address three high-level aspects: (a) **design and delivery** of learning task, (b) **rhythm or tempo** of the learning experience and (c) **practicalities**. Five elements are considered for of each the eight learning dimensions: (i) a detailed description; (ii) links to relevant literature; (iii) a summary of its rationale; (iv) examples from fieldwork (v) how it can be applied.

The Learning Dimensions evolved from earlier work called the Model of Programming Experience (MPEx), which in the LDs has been refined to be simpler and more robustly rooted in fieldwork. The LDs that are presented were created by reviewing the main findings of each of the four studies presented in Chapters 4-7 and synthesising them with the background literature. Where appropriate additional literature has been drawn in.

Design and Delivery: Overview

The first area, **design and delivery** of learning task, describes learning dimensions that relate to the design of activities or tasks that make up a learning experience. It consists of four dimensions: *Closed versus Open*, *Cultural Relevance*, *Recognition* and *Space to Play*. *Closed versus Open* describes the relative merits of designing learning tasks with or without a lot of detail and structure. *Cultural Relevance* describes the affordances presented by locating learning tasks within the learner's culture. *Recognition* describes opportunities that arise from enabling learners to share their work. *Space to Play* describes the impact of designing learning tasks that encourage iterative experimentation, for example with peers, and self-directed discovery of knowledge and skills.

Closed versus Open

Description

Learning experiences may be designed to contain a number of tasks or activities. The *Closed versus Open* dimension encapsulates the extent to which these activities have a well-defined structure, route and end point. A good example of a closed problem is programming a robot to follow a line. The task defines the answer: there is little scope for the learner to take ownership. Towards the open end of the dimension would be a free choice activity where learners are able to demonstrate competency in a given skill through the creation of a piece of work that is not constrained by the educator. An example is creating a robot dance.

It is possible to consider an entire learning experience as open or closed. However, in the Learning Dimensions the scope is restricted to the individual learning activities that constitute the experience. This gives a finer-grained picture and enables the design of a more sophisticated learning experience. In many cases, it will be helpful to think about parts of learning experience independently. For example, an introductory session using Processing, the flexible software

sketchbook and language (Processing, 2010), might comprise three activities. The first activity requires drawing some primitive shapes in specific locations. This is a closed task since the definition of the task dictates all aspects of the activity. This offers a high degree of support or guidance for the learner. The stakes are low and success is likely, since the task is well defined and does not require the learner to make significant independent steps. A follow-on task may be to draw a more complex form, such as a house, using multiple primitive shapes. This task is more open since the route to the end point is not specified and is left open to the learner. There is a defined end point, however, and the facilitator is able to prepare and explore a number of possible solutions. If the shape to be drawn is a house, the facilitator can experiment beforehand and be in a good position to support the learner. The final task may be very open and require the learners to draw something of their own choice. This raises some interesting new challenges: the learner is in control of the complexity of their drawing. The role of the facilitator grows to manage the expectations of the learner as well as offer to practical skill-based advice. This task is potentially more risky as the quality of the product is subjective and inexact, in that there is no clearly defined specification of what warrants a good house. Through these three activities, there is a clear gradient from closed task to open task with a shift in responsibility towards the learner, which in turn helps to drive independence and autonomy.

Related Literature

Resnick et al. give a good conceptualisation of the open or closed nature of a problem via their model for creative thinking tools (Resnick et al., 2009). The model refers to the characteristics of creative tools such as educational programming environments: creative tools are recommended to have a low floor (easy to get started with), a high ceiling (to support sophisticated projects) and wide walls (to support a number of different projects). A low floor for a tool can be a metaphor for a closed problem, in that it offers a high degree of support to novices. The high ceiling can be a metaphor for the openness of the task. An open problem supports a high degree of creativity and learner freedom.

The RoboCup Junior competition (2015) uses three different challenges, each of which is deliberately open. A first example is Robot football, in which the aim is to get the ball into the opponent's net. This is a compound task requiring learners to identify several complex actions to achieve that simple aim. Each action may have a number of variables. Although the aim is simple and there are well-defined rules governing the task, the rules require the identification of, and navigation through, many open problems.

Search and Rescue is another challenge in RoboCup. It requires the design and construction of a robot that can identify a lit candle and extinguish it. Again, the task defines the outcome but it comprises a number of open challenges, such as obstacle avoidance, location of the candle and a mechanism to extinguish it. The Robot Dance task was based on the dance challenge in RoboCup Junior, as it offers an open task. There is no expectation set in the defining of the problem. With the exception of failure to start, it is very hard to describe a dance in black and white terms of done or not done, in the way that one can easily tell if a line has been followed or not, or a candle extinguished or not. This means different learners can reach different conclusions and take different routes to reach their conclusion. From a practical point, this also offers a high degree of flexibility: a robot dance can be created in 15 minutes or an hour and a half, and by a wide range of participants. As the task has no defined end point, learners can extend and refine their work with any additional time.

Rationale

The choice of open or closed problems could have a large effect on the type of learning experience created and the responses obtained from different learners with different levels of ability.

Closed problems have an affordance for control: they offer a "quick win" and can be powerful in establishing the learner's interest. Since closed problems give control over the route the learner takes, this means that very specific skills can be covered or learned. A situation with a clear outcome from the outset may be regarded as less creative and more mechanical. All learners will

travel on a similar path to the same end point or level of competency. Crucially, the closed nature of the problem offers a high degree of support for the learners, which is particularly desirable for the novice.

Open problems give learners a direction rather than a path and given freedom to explore. An example of more open problem is a brief to build a robot that will autonomously navigate a given space is. The outcome may be defined but the strategies used to meet this requirement are far more open. This degree of freedom or lack of structure can be seen as intimidating by a novice, however. The advantage of this approach is it offers scope to develop independence and the ability to apply skills rather than reproduce them in controlled circumstances. Open problems foster creativity and are likely to lead to self-directed learning.

There is a sweet spot to be sought, however: if a task is too closed, learners will become disengaged; if a task is too nebulous and open, they will also become disengaged. It is likely that a learning experience will require a mix of open and closed tasks. These could offer a gradient that matches the learner's skill and knowledge level.

Example from Fieldwork

Robot Dance was designed to have a gradient from closed to open problems over three tasks. The initial task was to make the robot move forward. Given that the learners had most likely never programmed before, far less programmed using C, this task was walked through with them, in detail. The second task required learners to make the robot move and return to the same place. This required discovery and experimentation with a new instruction to make the robot rotate. Learners were given less support for this activity. The final task was to create a 20-second dance that had no constraints. The facilitator role had to shift to helping with bug fixes and offering direction for how learners could translate their ideas into code.

Whack a Mole is an example of an activity towards the closed end of the dimension. The task undertaken by learners was well defined from the beginning of the learning experience, with structured progression to a final state. The implementation details were defined by the supporting video tutorials to a large degree, although viewing the video was not enforced.

Digital Makers had a gradient that started with very small closed tasks, such as make a light flash, and then progressed to more open tasks. This gradient was designed to match the skill mastery of the learners, a strategy that was observed to retain the learners' engagement throughout the daylong activity. As confidence and ability in the learners grew, a greater degree of freedom was given by the facilitator. This was a more extended version of the strategy employed on Robot Dance. The educator role has to switch between teacher and facilitator as the task type changes. The educator switches between delivering knowledge of syntax, helping with the practicalities of getting code to run on an Arduino, and facilitating learners creating their ideas. This approach resulted in a substantial learning effect.

Application

When creating a learning experience, it is common for a selection of tasks to be used to help make more concrete a learner's understanding of whatever is being taught. When designing these activities, reflect on what the learner's skill mastery is and what the benefits and pitfalls are of choosing an open or closed task. Does the session require creative freedom or a more structured approach?

When skills and topics are new, it is important to ensure learners have sufficient structure and support in place to enable progress. This can be achieved with well-defined learning tasks that give learners a clear path to a desired outcome and a quick win. An example would be a learning task to make an LED blink with an Arduino. As the skill mastery of the learner increases, the openness of the learning task can also be increased to match. For example, a task to design your own code to communicate your name is quite open and requires the learners to self-direct to find

additional information and apply it. To go even further, you could propose a task for the learner to build a device to communicate their own code. This provides fertile ground to explore a number of interesting programming challenges, including light sensing, encoding and decoding of strings, network protocols and much more. In this example, the key feature is that the initial task is somewhat closed and so the approach to teaching this would be to offer a high degree of support. As the problem is made more open, the learners get more control and have responsibility for identifying the required skills to explore the problem at hand.

This dimension encourages educators to reflect on when to give learners freedom and when to close the scope of the task. In the studies conducted, good results have been achieved with a gradient tending from closed to open. Studying the effect of flipping this round or cycling through open and closed task presents an interesting area for further study.

Cultural Relevance

Description

Often part of a learning experience involves creating a product of some kind, such as code or a sketch. The *Cultural Relevance* dimension considers where this product sits within the learner's culture. It prompts consideration of whether or not the tasks they are asked to perform are authentic and relevant to their daily life experience. Ownership, personalisation and purpose are key aspects of creating a learning experience that will have high *Cultural Relevance* for the learner. If the learning experience is divorced from the world the learner inhabits, the *Cultural Relevance* will be low.

Ownership is achieved by giving the learner decision-making power in the learning experience. This helps to establish a relationship between the learning content and the learner. Personalisation involves tailoring the learning experience to an individual or group of individuals. Purpose is a crucial third dimension of learning activities. Addressing purpose is answering the 'who cares?"

question for the learner. This is what Edelson and Joseph (2004) describe as 'establishing relevance' in their Interest Driven Learning Design Framework: the learning needs to be linked to the real world to make it relevant.

An example of a learning experience with low *Cultural Relevance* would be if a learner were asked to demonstrate understanding of web authoring by creating an arbitrary learning product dictated by the educator. For example, write a database-driven website for shop X. Moving to a higher degree of ownership, the learner would be given some element of control over the learning product, for example, being given freedom to choose to design their solution for a particular shop in which they have an interest. Further still, they could be given a brief that requires them to demonstrate the skills and techniques they have learned by creating a website about any topic of their choosing. This creates a deeper relationship between the learner and the material with which they are engaging. *Cultural Relevance* requires that the learning experience is related to the learner.

Related Literature

Computer science education has been described as being knowledge-driven, which can result in it feeling clinical and devoid of context (Robins et al., 2003). Programming is complex and multifaceted: one approach to teaching it is to decompose it to its core constructs. This is natural and to a degree necessary: it suggests that a learner takes one step at a time. The drawback is that the investment required by learners can be quite extensive before any interesting payoff is received. It is possible that undergraduate learners could progress through a significant proportion of a programming course without building anything that relates to the world they live in. Considering the cultural impact of the learning experience has proved valuable in the studies conducted, and has been key to engaging and motivating learners. One approach to embedding learning in a culturally significant context is to use narrative based programming tools as categorised by Powers et al. (2006). Tools such as Alice (Cooper, 2010) can be used to develop a

learning experience that can embed learning within an interesting and culturally relevant context. An example could be using Alice to create a game related to the soccer club that a person supports.

Cordova and Lepper (1996) demonstrated that personalisation and choice are powerful motivators in a learning setting. The concept of learning being supported by the creation of a valued artefact is an idea rooted in constructivist epistemology (Papert and Harel, 1991) and drawn upon by Good and Robertson (2006b) in the context of learners creating high quality video games in learning experiences. The product or game in this case was something the learners valued and took pride in being able to create. In that experience, it was important that the game the learners were creating be of a similar standard to the games that they were experiencing in their social lives. The learners had a high degree of control over the game they were creating, which further reinforced their sense of ownership and value.

Rationale

Cultural Relevance is desirable in a learning experience as it establishes a deep connection between the learner and the tasks in which they must invest significant effort. Providing opportunities to offer choice and thus enable learners to take ownership and direction over of their work will increase the likelihood of high effort investment. Helping to form connections between the learning tasks and the learners' world can be achieved by designing opportunities for personalisation. It is also important for learners to see how the knowledge and skills they are acquiring relate to the real world. When carefully facilitated, this will have a positive effect on their learning. Placing computing concepts in a real world setting adds context for the new knowledge and skills and is likely to increase the learner's ability to apply the new skills in a novel setting (Boaler, 1998).

Example from Fieldwork

In all the studies conducted, an attempt was made to ensure there was some degree of *Cultural Relevance*. Robot Dance was selected as an initial study as there is a widespread literature reporting on the use of robots to support programming education. It is not difficult to extrapolate from the modest beginnings of making a simple robot move around an environment, to talk about much more sophisticated applications, such as self-driving cars. In Whack a Mole, the intention was to place learning in the context of a playful interaction or game-like product. Both of these studies place computer programming learning in a context that was easy to draw purpose from and authentically relate to the learners' world. They were designed to be of interest to learners and in practice, they were. However, they did lack ownership and personalisation.

In Digital Makers, there was a desire to offer a richer learning experience and attempt to build in ownership, choice and personalisation for the learners. This was achieved by introducing some design activities into which the programming and electronics could be wrapped.

The first activity was designed to generate an open list of situations to which learners had an emotional attachment. By collaboratively generating a 'post-it cloud' of situations that make one happy, irritated and excited, the learners were able to produce a rich set of possible problems to solve that related to them. The language for exploring and defining these solutions should not be limited to code: here the "language" used was the post-it notes. The element of choice was introduced in an activity in which each learner had to pick three of the Post-Its from the cloud and then storyboard how the technology they had been learning about could be applied to solve or assist the problem. These were shared with their group and the final large task for the learners was to prototype one of the ideas that they had described in their storyboard. This final task consolidated the programming and electronics skills that had been learned in the morning; it placed those skills in a context that the learners had ownership over and were able to personalise. Physical apps such as the 'FIFA notification station' or 'homework monitor' demonstrate well a

context for learning that would be hard to realise without a co-creation approach. Co-creation here describes the process in which a facilitator supports the initial learning, and the application of the learner's imagination results in a personalised product of a quality that the facilitator alone could not have created.

From simply trying to place learning in a context that is of interest to learners, to efforts taken in Digital Makers to co-create a product that is of value to learners, it has become clear that designing for a high degree of *Cultural Relevance* can enhance engagement and motivation.

Application

Considering *Cultural Relevance* when designing a learning experience can be as simple as identifying a real world context for the application of the knowledge or skill. Two examples are using physical computing to design a physical representation of a Facebook notification or a medication reminder for an elderly relative.

Ownership is achieved by giving learners control and the power to make choices related to their learning experience, such as "Build an interface for this simple game by choosing only two the following sensors..." The task is no longer generic or arbitrary and the learner has opportunity to influence what they are doing.

For a learning experience to be considered personalised, it needs to be flexible enough to wrap around a key characteristic that is unique to the learner. For example, the learner may have an elderly relative who has difficulty remembering when to take their medication. Many fascinating possible ideas and solutions that could be explored using basic Arduino components and programming. This helps to establish a rich emotional attachment between the learner and the learning, and it will foster a high degree of engagement and motivation. *Cultural Relevance* seeks to relate and embed the learning in the learner's world.

Recognition

Description

It is typical for a learning experience to result in the generation of a product. It may be a program, a sketch or a concept. The *Recognition* dimension considers the potential for the learner to share the product of their learning. As early as nursery school, learners seek recognition from their teachers, peers and parents. A good example of this is pleasure gained from the displaying of work on the walls of the learning environment for all to see. In the Learning Dimensions context, a model of *Recognition* has three parts: (a) the mode of the interaction, (b) the audience size, and (c) the response or result. Each of these, when considered together, will have an effect on the learner's engagement and motivation.

(a) Mode of interaction

Different modes of interaction proposed for *Recognition* are Seen, Presented and Discussed. Each of these will be described next.

Seen

This represents a *Recognition* situation where the product of the learning is visible to others but there is no contact between the learner and the audience. Malone and Lepper (1987) give the example of an exhibition as a mode for learners to share their work. Malone and Lepper go on to describe the difference between sharing a product and sharing a process, where product may be a painted picture and process is the act of painting it. An exhibition where art works are hung in a gallery offers an opportunity to share the product but not the process.

Presented

This mode denotes a *Recognition* situation where the product of the learning is presented by the learner to the audience, but no dialogue with the audience is permitted. Examples could be a demonstration or a presentation. When presenting work or demonstrating work, the learner is able to articulate a lot of their thinking, which will give the audience a rich insight into the process that has led to the product.

Discussed

With a discussion, a conversation about the product of the learning can take place between the learner and the audience. An example would be an oral exam in which work is discussed. Discussion is considered the richest of the three modes of interaction. Learners gaining recognition through discussion are not just exposing a product, presenting an idea or artefact but they are also engaging in rich discourse about the artefact and process. This should ensure the audience and the learner reach a shared understanding of the idea or knowledge being presented. This resonates well with Vygotsky's theory of socially constructed knowledge (Vygotsky, 1980).

Each of the modes of interaction share two common factors that will modulate the degree of motivation a learner experiences as a result of the recognition: audience size and depth of interaction.

(b) Audience Size

Audience size has an impact on the significance of the interaction on the learner. In an educational context, it is possible for work by learners to be shared at various levels, such as learning group, class, year, school and district. Whilst historically there were physical limitations upon the range of audiences that a learner's work was likely to reach, now technology has broadened this significantly. Many modern programming educational packages leverage web community to support education through recognition from peers (Brennan et al, 2010). This may result in a piece

of work being viewed by thousands of people. As an example, the author has shared sketches on Open Processing that have been viewed over two thousand times in two years (http://www.openprocessing.org/sketch/106385)

(c) Depth of Interaction

The third aspect of *Recognition* considered is the depth of interaction response between a learner and audience. This is an important component contributing to the motivational effect of *Recognition*. Four possible responses are for the work to be (1) viewed, (2) commented upon, (3) detailed critique and (4) used or built upon. These symbolise the amount of time, effort and interest the observer has invested: viewed represents the shallowest interaction and built upon represents the deepest. A piece of work may be simply looked at, but an observer may choose to engage further, selecting the degree to which he or she desires to comment upon the work. Further still, the observer may offer a detailed critique of the work, demonstrating a deep understanding of it. Finally, in a code shareable project, the work may be built upon such as in OpenProcessing (http://www.openprocessing.org/) or other online communities such as Scratch (Resnick et al., 2009). The *Recognition* dimension builds upon Malone and Lepper's (1987) definition of recognition as an agent for intrinsic motivation in learners. By drawing together the features described, the degree of opportunity for learner recognition offered by a given learning experience can be shaped.

Related Literature

Sharing work has potential benefits from the pride found in the sharing and learning with a learner's peers, although learners may have to overcome any embarrassment they may feel when put into a presentation situation. *Recognition* is identified in Malone and Lepper's taxonomy of intrinsic motivation in learning (Malone and Lepper, 1987). Malone and Lepper describe recognition in terms of visibility of the learning. This may relate to the visibility of the process, for example when demonstrating a skill in which a learner has developed proficiency. It may also

refer to the visibility of the product, for example, when an item of artwork is exhibited. It has been shown (Blank and Kumar, 2010) that competition has been shown to motivate a few, but demonstration motivates a far wider audience, which underscores the value of *Recognition*. One of the key benefits of web 2.0 and the enablement of user-generated content has been the ability for learners to share their work amongst their peers. Scratch (Resnick et al., 2009), Greenfoot (Kölling, 2010) and Alice (Cooper, 2010) all have sizeable online communities that enable the sharing of products of learning. There is evidence to suggest that this is an integral component of the success of Scratch (Resnick et al., 2009). It is also likely to be key component of other tools.

Rationale

Recognition has been shown to be a positive task extrinsic motivator for learners (Malone and Lepper, 1987). People value sharing their work and gaining validation from others. In addition, learning benefits may be obtained by requiring learners to formalise their understanding to a point they can communicate it to others, as with self-explaining (Chi et al., 1984). Finally, where open tasks have been set, sharing each learner's approach and the outcome of their work exposes learners to alternative approaches and opportunities.

Example from Fieldwork

Robot Dance had the explicit need for the learners to share their work in the performance put on at the end of the session. This can be regarded as a high value, small audience sharing opportunity: the audience comprised a small set of people that have undergone the same learning experience. They were eager and had invested time in observing what their peers have created.

In Digital Makers, there were three opportunities for recognition from peers and the facilitator. Early on in the session, an idea generation activity was carried out to begin to elicit potential project ideas. Learners were asked to note down and pin on the wall descriptions of situations that evoke different emotions. In turn, situations that evoke happiness, irritation and excitement were

asked for. The idea of placing things on walls to generate 'information radiators' was first observed by Robinson et al. (2007) in the context of agile development. An 'information radiator' is simply a collection of information placed in a position that is visible to all that have an interest in it. For example, in agile software development, physical scrum boards are information radiators that offer quick access to key information about the sprint. When the information radiators are formed, learners are able to view what other learners had shared. In terms of the *Recognition*, this is an example of seeing. This was relatively light sharing with a low degree of nervousness for the learners.

The second opportunity for *Recognition* took place after several hours of practical work on programming and wiring Arduino circuits. This session built on the ideas-generation activity. After an introduction to storyboarding, individual learners were asked to pick three Post-Its from the wall. For each of the situations, they were asked to storyboard a potential physical app to assist or support the situation described on the post-it. For example, one learner had noted that remembering to do all their homework was a source of irritation. The solution was a homework counter physical app. At random, finished storyboards were selected and presented by the learner that had created them. This is an example of presenting to a small interested audience that were also involved in the activity. This was more nerve-wracking for the learners, since presenting to a group may be intimidating. The learning benefit is further development of the concept or design idea that has been supported by the storyboard.

The final opportunity for *Recognition* came at the end of the session. After the main build activity, each pair, group or individual had a short amount of time to demonstrate their physical app and pitch to the groups why they had made it, what problem it solved and for whom. This is an example of symmetric *Recognition* of moderate depth with an invested audience. This activity generated a fantastic atmosphere, with learners eagerly tweaking and refining their work as the anticipation of demonstrating their work increased. The fact that this was a demonstration of a quite rapidly prototyped piece of work appeared to motivate learners. There was an observable

sense of pride in having the opportunity to share work they literally just learned how to make. This served as a driver to work hard on what they were creating. Another advantage came from sharing the different learning that had taken place, as the learners had all diverged quite broadly in their approaches. The demonstration at the end was a great opportunity to learn how others had used the technology.

Application

Identifying opportunities for learners to gain *Recognition* in a learning experience is a valuable way to consolidate the session or activity. It gives learners time to reflect and formalise their understanding by presenting it to others. As with all the learning dimensions, the *Recognition* dimension provides some structure and underpinning to how *Recognition* can be used to enhance a learning experience. There is a range of opportunities, from displaying work to discussing work; each has different advantages and potential risks. It is important that when learners are sharing work they have a solid understanding of what is expected and the group has agreed on a set of rules, to ensure all learners have a positive experience. Presentations can be a source of anxiety but this can be reduced by ensuring learners are aware they will all be taking part and by emphasising the importance of being courteous to their peers.

As cited in the literature and confirmed in studies described in this thesis, sharing work to gain recognition can be a powerful motivator for learners. It can also distribute and consolidate knowledge and understanding amongst the group. It is important to be aware that for many people, presenting work is intimidating, particularly where the item being presented is new. This can result in people feeling quite vulnerable. It is therefore very important that steps are taken to create a safe environment for sharing work. This can be achieved quite simply, for example by generating a set of rules within the group so that the presenter is treated with respect. Typically, consensus will form around one person speaking at a time, with questions being selected by the presenter or the teacher.

Space to Play

Description

The *Space to Play* dimension seeks to break down the traditional view of a teacher-learner relationship. It encapsulates the extent to which a learning experience offers and encourages learners to explore independently, experiment and iterate over aspects of the learning experience. This idea is rooted in constructivist learning theory (Papert and Harel, 1991): learning takes place best when learners engage in project work that results in an artefact that is relevant to the learners, as described in the *Cultural Relevance* learning dimension. *Space to Play*, however, addresses the fact that space and independence may be intimidating for certain learners. Furthermore, it acknowledges the tension between the learner as an individual and a need to cover a particular amount of content with a group of learners. Where space can be intimidating, direction, constraint and facilitation can be catalysts to creativity and learning.

The *Space to Play* dimension suggests there should be a flexible structure to learning experiences, with frequent opportunities for learners to iterate over a concept that has just been taught. This empowers individual learners to approach exploration on their own terms and take ownership of the learning experience. As a simple example, a section teaching about setting a colour for an LED could be followed by time for learners to make concrete this new knowledge by discovering the correct proportion of red, green and blue to make several other colours. As the learner's skill and competence grows, the size and complexity of this *Space to Play* can grow to meet their abilities. More complex and multi-faceted projects could then be attempted.

This approach to session design should support a varied range of learner abilities. The primary desire is for all learners to achieve a core level of competence in whatever is being taught. By providing flexible time gaps with space to experiment, learners of differing abilities can work at their own pace throughout these sections. *Space to Play* is a valuable strategy when using this approach to session design. *Space to Play* is a time management strategy where a fixed amount

of time is given to an activity. Two focal points are included at either end of a time box. The focal launch leads in to the time-boxed activity. This point draws together the group and ensures they all have a clear and common understanding of what they are expected to do for the time-boxed period. This is also an important opportunity for answers to any questions or points of clarity to be shared among the group. At the conclusion of the activity, there is a focal landing point. This serves as an opportunity for learners to share their findings with the whole group. It is also an opportunity for the facilitator to ensure every learner has sufficient understanding to progress to the next learning point. *Space to Play* can be designed by defining these three steps: (1) focal launch, (2) time boxed activity and (3) focal landing.

Related Literature

Space to Play aims to design a learning experience that empowers learning by iterating over opportunities for experimentation and practical application of new knowledge and skills. Several approaches in the literature are relevant to this.

Contributing Student Pedagogy, introduced by Hamer et al. (2008), was devised to motivate and engage learners by blurring the lines of teacher and learner. This is achieved by encouraging and supporting learners to assume roles and perform tasks traditionally associated with teachers such as assessment and contented delivery. In both *Space to Play* and Contributing Student Pedagogy (Hamer et al., 2008), the teacher assumes the role of a facilitator rather than a gatekeeper to knowledge. This begins to erode the expectation that the teachers possess knowledge and deliver this to learners. Instead, it sets up a much more progressive and subtle learning experience in which learners have a degree of influence on the direction of their learning.

Playful learning has been a common thread in much of the work conducted by Resnick (e.g. 1996, 2007, 2009). A central part of his implementation of constructivism is that a learning experience is greater than the sum of the indicative content. Learning is a social activity that centres on imagining, creating, sharing and reflecting. *Space to Play* aims to support these principles and

create an experience that nurtures independent learning by following the three simple steps. *Space to Play* has similarities to 'tinkering':

> *The tinkering approach is characterized by a playful, experimental, iterative style of engagement, in which makers are continually reassessing their goals, exploring new paths, and imagining new possibilities.* *Resnick and Rosenbaum (2013).*

Tinkering is a modern implementation of Papert's constructivist learning theory. Scratch (Resnick et al., 2009) and Programmable Bricks (Resnick et al., 1996) have created a medium to support exploratory project-based play. Resnick's group have designed these tools to support a lightweight and provisional approach to the creation of ideas. If learners are expected to self-direct, it is inevitable that errors will occur. It was important that these tools allow graceful recovery and agility to cope with shifting directions and new knowledge. Using a term from Cognitive Dimensions (Green and Petre, 1996), these tools require a low viscosity. *Space to Play* is a technology-independent design principle to create a learning experience that embodies similar qualities and results in a low viscosity.

Rationale

Building *Space to Play* into a learning experience balances delivery of indicative content against learner personalisation and ownership. This gives learners space to consolidate their newly acquired knowledge at a pace that suits them. The space learners are given then allows for a degree of flexibility that will serve learners of differing ability well. The simple structure of focal launch, time-boxed activity and focal landing provide a simple structure to be applied to a range of learning tasks.

Examples from Fieldwork

Throughout all of the initial studies and follow-up evaluations, this approach has matured and grown to be a key characteristic of how learning experiences are designed. Robot Dance was built

around a series of content delivery and independent consolidation activities that embody *Space to Play*. For example, the second activity in Robot Dance involved making the robot move forward for a specific distance and then getting the robot to return to the original point. The second challenge was deliberately left open. The launching focal point made sure all participants understood the challenge. This activity was time-boxed to five minutes and learners were offered support if needed by the facilitator responding to learners' questions with increasingly focused questions in return. For example:

learner: "How do I get back to where I started?"

facilitator: "What instruction from your list do you think might help?"

learner: "we used forward last time, maybe spin?"

facilitator: "Why don't you try that and see how it works?"

The dialogue is quite subtle; the aim of the facilitation is to guide the learners in the correct direction, giving as little input as possible. It would be simple to respond to the first question with guidance to perform a spin then wait for 1.5 seconds then move forward again. The aim is always to enable the learner's discovery over knowledge delivery. In the previous activity, learners discovered that the distance the robot travelled was controlled by the wait time between the move forward and the stop instructions. For this challenge, the same relationship needed to be discovered between the spin instruction and the following instruction. The focal landing point drew the learners back together. They were then able to share how they had achieved the task. By moving round the groups to see how long it took their robot to perform a half turn, it was possible for the facilitator to discuss with the learners how all the robots were different, and explore reasons for this, such as battery charge level and differing levels of friction in the drive trains of the robot. This is an example of *Space to Play*: the task has a well-defined focal launch, a time-boxed activity that requires learners to experiment and learn through trial and error, and a focal landing point that consolidates the group's understanding.

The Whack a Mole study was constructed around three guided activities with a final more sophisticated activity that rested upon the previous learning. It explored a different approach to *Space to Play*. Each activity was facilitated by a video tutorial via which the learner was walked through the activity in detail. The learner was encouraged to pause the video when they wanted, to allow them to code along with the video. The launching focal points were learner-driven in this case. Some learners may watch one instruction and then attempt it; others may be comfortable with watching three steps and performing them all together. As each video tutorial encapsulates a single required skill, at the end of the video there is a good opportunity for learners to experiment with what they had achieved and refine it. For instance, at the end of the first example, they had made a light flash. By means of this task, they have become familiar with the infinite loop in which the Arduino code is wrapped. This loop contains all of the function calls that will control the LED and how quickly it will flash. There is a good opportunity to experiment with different flash rates.

The main advantage of using video in this way is that learners can progress at their own pace and have control over when they take a pause to try a new skill and iterate over it. There is also the opportunity to re-watch the video if, for example, a particular piece of instruction is challenging. Consistency is another potential advantage, though it could be argued that removes the opportunity for spontaneous learning conversations. The disadvantage of this application of *Space to Play* is that there is no group launch and landing focal points. This removes the opportunity for learners to share and learn from each other. Removing the *Space to Play* could result in learners spending too much time on task. This is a contentious point, as balance has to be drawn between letting learners move at their pace and stopping them from stalling or over-polishing a skill. The law of diminishing returns comes into play. There is also a risk that learners may focus on easier tasks, when it may be better to tackle new activities that are more challenging.

In the Digital Maker study, *Space to Play* was integral to the design of the learning experience. The majority of the morning was spent learning about Arduino programing and electronics.

Programming is a high precision error-prone activity; electronics prototyping has similar characteristics. To support this, the session was designed around frequent short *Spaces to Play*. A piece of programing and electronics was demonstrated and learners were given a time-boxed time to try the task for themselves and experiment. The components used by the learners were a speaker, LED, servo, RGB LED, potentiometer and button. Each learning activity focused on the key characteristics of the components and how to program these. At the focal launch point for the speaker, the learners had been shown how to make the speaker start and stop playing a tone. In the time box, they had to make their speaker play a tone, then experiment with different tones and then use a series of tones to play a little tune. The focal landing point for this activity involved going round the groups and listening to what they had programmed.

The rhythm of introduce, demonstrate, handle and reflect proved effective at engaging learners throughout the studies conducted in this research.

Application

It is possible to apply *Space to Play* to many learning experiences: simply give design time, experimentation time and build time. This is achieved by defining three things. Firstly, define your focal launch point. If your learners are to engage in a period of independent work, it is important they have all of the knowledge, materials and instruction required to achieve this. Independence and space need to be supported. This can be as simple as getting the groups to reiterate the key aims of the activity they are about to engage in. The simple act of *Space to Play* the activity gives a strong indication of how much effort or depth is expected. A five-minute activity will be approached quite differently to a one-hour activity. Secondly, facilitate your time box so that you can give support to learners where they need it. When responding to questions, aim to respond with direction rather than answers and encourage a trial-and-reflection approach rather than an approach that seeks knowledge from the facilitator. The third and final task is to prepare your focal landing point. Given space, learners will diverge to some degree and this is

one of the reasons for taking this approach. If the aim is to get all learners to a specific point at the end of the session, it important to pull learners back together before moving on. The focal landing point should give learners opportunities to share what they have discovered and learn from the finding of the other groups. Is there consensus or is there not? This offers a great opportunity to be responsive to the group and enable spontaneous learning conversations about things that are not key to the session but interesting to the learners. *Space to Play* is setting up the required circumstance for learners to learn how to systematically explore and apply new knowledge and skills.

Chapter 9: Learning Dimensions Part II. Rhythm or Tempo

Rhythm or Tempo: Overview

The second area that the Learning Dimensions address is the **rhythm and tempo** of the learning experience. This chapter describes features of the learning experience that deal with the flow of the experience through time. It has two dimensions: *Driver Shifting*, and *Risk Reward*. *Driver Shifting* describes the affordances of transferring the role of driving the learning experience from the educator to the learner, vice-versa, or via a collaboration of both. *Risk Reward* describes how the duration of tasks and the frequency of feedback can be adjusted to suit different learning experience needs.

Driver Shift

Description

Driver Shift is a new concept that has emerged from the studies conducted. This dimension attempts to capture which actors in the learning experience are driving, i.e. taking control of the learning experience at a given point in time. It is likely that there will be transitions between learners and facilitators as drivers throughout a session. There can also include a mid-state in which collaboration between the learner and the teacher takes place. For example, a classic higher education style lecture where the lecturer projects content to the learners for a sustained period would be regarded to have a very low degree of *Driver Shift*. In contrast, a guided practical session with a tight cycle, in which learners are shown a brief example and then given space to try it, would be said to have a high degree of *Driver Shift*. This can be represented as a time series graph with the three different *Driver Shifts* depicted via the y-axis and time progressing via the x-axis (Figure 8.1).

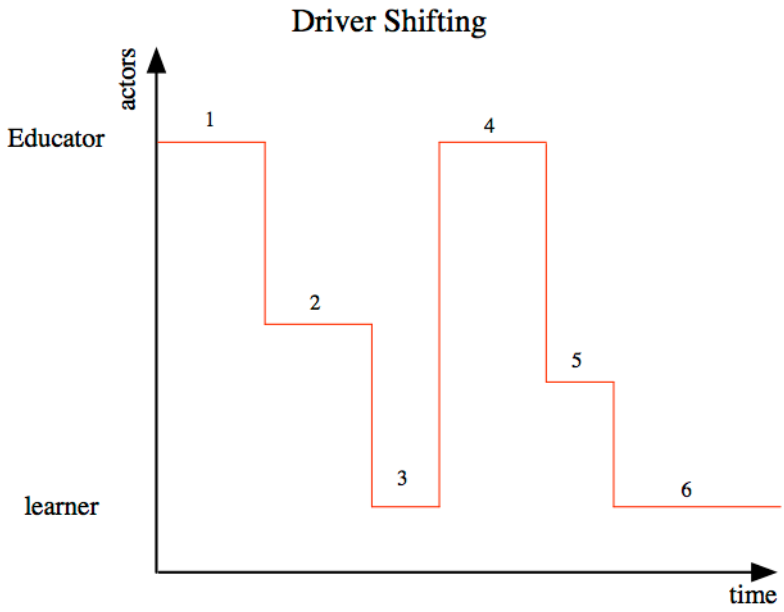


Figure 8.1: Driver Shifting Time Series Session Breakdown

Figure 8.1 shows an example of learning experience that has a high degree of *Driver Shift*. It begins with the facilitator as driver and ends with the learner as driver, and there are four other shifts before the end. To illustrate this concept each section will be described. (1) At the session introduction, the facilitator is telling learners what they will be doing. (2) During a balanced collaboration period, learners and the facilitator work through an example together. (3) For an independent consolidation, the learners work through a further example on their own. At point (4), a new concept is introduced by the facilitator. Point (5) illustrates a skewed collaboration between the facilitator and learners, where the learners are more dominant; the facilitator may be offering less guidance and interjecting only when needed. Point (6) is a further example of an independent consolidation period. This dimension is proposed as a broad-brush approach to assist in the creation of learning experiences that are engaging and encourage the learners to become active participants rather than passive recipients. The intention is not to become focused on the fine detail of exact timings and the exact proportion to which collaboration is skewed one way or the other. Measures can be considered relative to one another. *Driver Shift* can be considered at two levels: firstly, it can be considered at a high level, as a measure of the degree to which driving

the session is distributed between participants. Secondly, it can be considered at a lower level, in which specific interactions and durations are designed or reviewed.

Related Literature

Driver Shift has emerged from the empirical work carried out as part of this research as a valuable method to shape a learning experience. Before examples are given in the next section, its relationship to the literature is considered. It is widely accepted that undergraduates can focus on a traditional lecture for around 20 minutes (Johnstone and Percival, 1976). Advice to new lecturers encourages them to switch modality at intervals of this time to re-engage with learners. The flipped classroom approach (Tucker, 2012) takes this a step further and proposes that the content delivery is a waste of valuable contact time. In a flipped setting, learners can consume content in an appropriate modality and the lecture period can be used for discussion. This resonates with Perkins' states of knowledge in the novice (Perkins and Martin, 1986). Content delivery leaves learners with 'inert' knowledge, whereas the flipped classroom offers space to make concrete this knowledge. The *Driver Shift* dimension encourages educators to think carefully about how to allocate time to ensure learners are engaged and given the opportunity to apply concepts.

In Contributing Student Pedagogy (Hamer et al., 2008) discussed in the background that there was a strong desire to break down traditional views of the teacher-pupil relationship and shift the role of the teacher from gatekeeper to guide. This has resonance with a desire for a high degree of *Driver Shift*.

Dreyfus and Dreyfus (1986) propose it takes 10 years for a novice to progress to become an expert in computer programming. A side effect of this is the expert-novice match mismatch: whilst it seems intuitively correct that being taught by an expert is a desirable thing, it can presents some challenges. The expert will have difficulty in identifying the tacit knowledge that is taken for granted but is absent in the novice. *Driver Shifting* offers opportunities for feedback from learners,

which is invaluable to guide the facilitators about what learners are understanding and what they are still finding challenging.

When applied to programming in particular, driver switch offers an opportunity for learners to switch between comprehension and generation (Robins et al., 2003). As discussed in the background section, comprehension and creation of code are different yet intertwined competencies. In a traditional situation, programming concepts are often delivered in a lecture with low *Driver Shift*. This will lend itself to a code comprehension mode. That learning will then be supported by a lab session which is driven by the learner and which requires predominantly code creation skills. It is proposed that a session with a high degree of *Driver Shift* can bring together learning about both code comprehension and code creation.

Rationale

The *Driver Shift* dimension encourages educators to reflect on the impact that switching responsibility for driving the learning experience may offer. In certain circumstances, driver switch will present a desirable mode switch to reinvigorate engagement with learners. It can also offer the opportunity for learners quickly to try out the skills being taught. Where driver switch is not present, there is a risk that learners may become disengaged in the learning experience. An extremely high degree of driver switch is likely to become distracting and reduce the chance of a flow state (Csikzentmihaly, 1991). As with all the dimensions, there is not a single optimal state to fit all situations.

Example from Fieldwork

Driver Shift was applied in the Digital Makers study, though it can be applied retrospectively to the earlier studies as well. The model used in the Digital Makers study involved a progressive transfer of direction from educator to learner. This reflects the aim of many learning experiences

where new material is introduced. The desire is that at the end of the session the learners are comfortable and able independently to apply the material covered.

The first half of the Digital Makers day focused on giving the learners the skills required to work with Arduino boards. A high degree of *Driver Shifting* was used as textual programming and wiring of electrical components is fiddly and error-prone. A series of short narrowly scoped demonstrations were followed by *Space to Play* (see *Space to Play* LD) for the learners to try what had been demonstrated – this proved effective.

As the day progressed and competence with the newly acquired skills and knowledge grew, the length of the learner-driven blocks was increased and the scope of the task opened out. This offered more opportunity for creativity. Throughout the session, the role of driver oscillated between learner and facilitator with a gradual progression towards the learners working autonomously under their own direction, seeking advice rather than direction from the facilitator.

Reflecting on Robot Dance, *Driver Shift* was also evident. Robot Dance was designed to be delivered in under an hour, yet it still offered the opportunity for cycles of *Driver Shift*. As in Digital Makers, the ratio of facilitator time to learner time gradually crept up towards the end of the session at which point the learners were given the opportunity to apply all the skills they had acquired and to program their dance. In contrast to Digital Maker, *Driver Shifting* was effectively used to ensure that all groups reached the same milestone at the end of the session.

Application

The *Driver Shift* dimension encourages educators to identify opportunities to switch control of the learning experience between the facilitator-led and learner-led activities. This oscillation is one approach to ensure that learners remain actively engaged in the learning experience. If considered in finer detail, as in Figure 8.1, it is possible to model a learning experience in some detail and reflect on how the session will flow and if a progression towards autonomous learners

is possible. *Driver Shift* could encourage a gradual shift of direction and control towards the learner that tracks their skill and knowledge mastery.

Risk Reward Cycle

Description

The *Risk Reward* cycle considers the relationship between the investment of effort or risk that a learner undertakes and the reward when feedback is received. Investment of effort without confirmation that the correct actions are been taken by the learner is considered a risk. This is because it may result in wasted effort or even worse enforcing an incorrect understanding or application of a skill.

Feedback can take a number of forms, such as observation and direction from a teacher or the completion of a complete program that can be executed. For example, this could be the time taken to write a hello world program. For Java, the amount of effort investment required from the learner to get the payback or reward of some text being displayed is non-trivial, so high risk. In a language like Processing, the effort investment made by the learner before observable outcome is much shorter. In Processing and other scripting languages, it is possible to render output in one line of code, so lower risk. This programming effort is considered a risk to a learner, since independent work in a particular direction for a given period time without feedback has a chance that the learner has moved in the wrong direction. In any learning experience, there will be a cycle of learner effort investment and pay-off as the learner works through different tasks and receives feedback as they progress.

This dimension can be considered at two levels. At a coarse grain or high level, a learning experience will have risk reward that sits on a continuum from tight to loose. Tight describes a situation in which the learner effort investment is short and the feedback is frequent. At the other extreme, loose describes a situation where the learner effort investment is significant and

sustained but the feedback is infrequent. It is unlikely that a learning experience will consist of a single investment and reward. If we consider Risk Reward at the level of a correctly executing and functioning computer program, then there are few instances where a program is written, executed and functions correctly first time without any debugging and is thus complete.

The Risk Reward dimension can also be considered at a fine-grained level, using a time series graph depicting risk throughout the duration of the learning experience. The graphs in Figure 8.2 depict three example sessions. The y-axis represents increasing risk; the x-axis depicts progression of time. A sharp decline in risk represents a feedback event. It is important to emphasise that the use of time series graphs is purely for illustrative purposes: these are sketches to aid in the description of Risk Reward cycles. There is no intention to measure things or to attempt to gather precise metrics. These are estimates of time and risk. Paradoxically, the more accurate the estimate, the less valuable the graphs become, since the focus shifts too much to the detail. This is counter to the intention to use time series graphs to offer an overview of an entire session.

Figure 8.2 depicts the *Risk Reward* cycle of a classic lecture with homework. The *Risk Reward* cycle is very loose, as feedback is not received until the homework is completed and returned. This approach requires the learner to be able to receive, process and then act upon a great deal of information before feedback is received. Although the lecture remains a dominant traditional teaching tool used extensively in Higher Education, it is unlikely that the *Risk Reward* would be this stark, with no engagement from learners throughout the lecture. Many lectures will be punctuated with questions, quizzes and other mechanisms to engage learners and gauge their level of understanding.

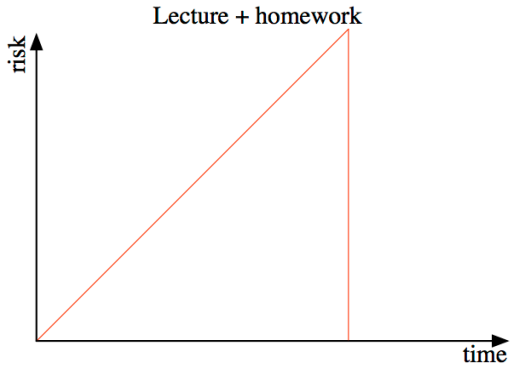


Figure 8.2: Example of Risk Over Time: Lecture

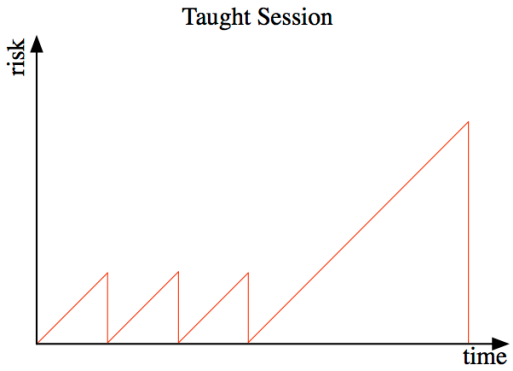


Figure 8.3: Example of Risk Over Time: Taught Session

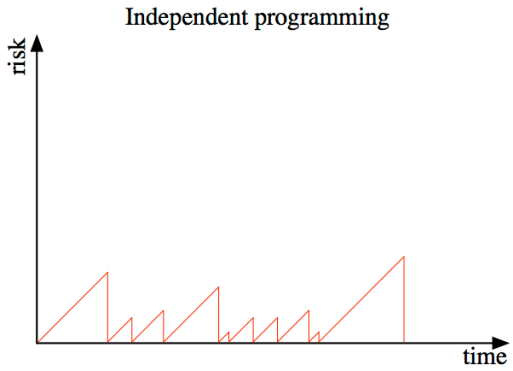


Figure 8.4: Example of Risk Over Time: Independent Programming

Figure 8.3 depicts a taught practical session. The session is structured around three small examples and a final larger and more independent exercise. At the conclusion of each exercise, a

well-defined outcome is achieved. One advantage of this approach is that there are several well-defined points at which the learners are brought together; this reduces the level of risk getting too high.

Figure 8.3 depicts independent programming. Each sharp drop is the successful execution of the program. The first large peak represents the initial investment of getting the program to an executable state; minor tweaks and extensions are performed, with the larger peaks representing extensions that are more complex or refactoring exercises. The amount of structure and support placed around a programming activity will have an impact on the shape of this graph, with greater structure and support reducing risk.

A special case in this dimension is the *time to first task*. This is the time it takes to get the learners to start the first task. This is a crucial first interaction between the teachers and learners and establishes a rapport with learners. This has emerged from the studies conducted as an important point for establishing a rapport with learners.

It is worth noting that the gradient of the rising line on the graphs represents the rate of acquisition of risk. This is potentially an interesting concept, with different activities acquiring risk at different rates. For example, each minute spent editing a sizable and complex program written by someone else is more risky that writing a new program, as the learner is simultaneously adding new content and having to understand what is already in place. Figure 8.5 depicts how two different activities can reach the same level of risk in vastly different periods. Where risk is acquired quickly, it is important to have more frequent feedback cycles such as when covering new material. This is necessary to avoid a situation where a learner has invested a substantial amount of effort but has been moving in the wrong direction.

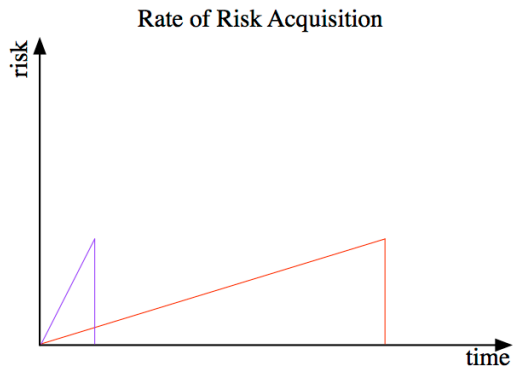


Figure 8.5: Activities with Different Levels of Risk.

The *Risk Reward* cycle, along with other Learning Dimensions, describes the tempo of a learning experience. The individual learner and the level of the learner are important considerations when reflecting on the shape of *Risk Reward* cycle that is being designed. As with all Learning Dimensions, there is no single optimal point it: rather, the dimension gives a reflection of a number of mitigating circumstances.

Related Literature

Identifying barriers to programming is understandably an important aspect of supporting the novice. One of the main drivers for languages specific to teaching programming is to address the fragile nature of computer programs. Like a house of cards, a small alteration to any part may have globally undesirable consequences that are difficult to track down. Computer code requires a very high degree of accuracy and as a result, programs are brittle. Small changes can break things in ways that are unclear to the novice and sometimes also to the expert. Visual programming languages attempt to address this by making 'jigsaw' pieces that only fit together in a fashion to form executable code (Powers, 2006). Green and Petre (1996) characterise this in the Error Prone cognitive dimension. A language like Python, for example, where even white space has a syntactic impact on the code, would be described as error prone, as there is great deal of potential to make mistakes. Lego RoboLab (RoboLab, 2015) is at the opposite end of the spectrum, with a limited set of constructs and a limited number of ways to assemble them. This

'design against errors' approach leverages a trade-off between errors and creative freedom. Visual programming languages like RoboLab and Scratch (Resnick et al., 2009) are designed to remove the possibility of syntax errors. These enable teachers to design a session with a tight Risk Reward cycle: the time between investment of effort by the learner and some observable outcome can be short.

One of the key design decisions for the creation of creative tools put forward by Resnick et al. (2009) was that they must possess a 'low floor' or enable a quick win for learners. The *Risk Reward* dimension seeks to take this a step further than purely identifying difficulties. It encourages thought around the relationship between challenging aspects of work and the reward learners receive.

Rationale

The *Risk Reward* dimension draws attention to an important relationship found in any learning experience. Learners will undertake activities that present risk, such as writing code. For the duration of this activity it will be unclear if the behaviour of the code is as desired. The reward, which mitigates the accumulated risk, is an executing program or feedback from a facilitator. By considering the temporal relationship between risk activities and feedback events, it should be possible avoid situations where learners are reaching out too far. It is also possible to create a tempo of sessions to suit different levels of learner and different material. High risk, error prone activities suit a high *Risk Reward* cycle and low risk activities require a looser risk reward cycle.

Example from Fieldwork

Throughout the delivery of Robot Dance, it became apparent the time to first task was an important consideration when establishing a relationship with a new group of learners. With younger learners (primary and secondary school pupils), the aim was to reduce this to less than ten minutes. In this time, the context of the session was set and the required knowledge to get the

robot to perform an action was conveyed before learners performed the action for themselves. Following further studies and the more established conceptualisation of *Risk Reward*, it is possible to reflect on the *Risk Reward* cycle for Robot Dance. There were three cycles in Robot Dance: the first lasted around 15 minutes and covered the introduction and a fully described example. The second cycle was also around 15 minutes and involved working on a less well-defined problem. The final cycle was 30 minutes, including the open-ended robot dance task and the final performances.

In Whack a Mole, the *Risk Reward* cycle extended what was found to be successful in Robot Dance. The learners in this study were undergraduate learners and so more material could be covered in a smaller amount of time. Three cycles were used to deliver the underpinning skills required to complete the final task of implementing a Whack a Mole game. Firstly, instructional videos were used to introduce the skills requires so each learner was able to spent a different amount of time on each of the task. On average, the task took between 10 and 15 minutes, with around 30 minutes being spent on the final challenge.

The Digital Makers study employed a similar structure in which the first half of the day was spent in a very tight *Risk Reward* cycle of around 10 minutes or fewer per example. This was then followed by an open-ended 90-minute creative building task.

The overarching aim of many learning experience is empowering learners to transition towards independence. In all the studies conducted, the length of *Risk Reward* cycles was extended to a larger open-ended task that reflected the learners' confidence with skills being taught.

Application

This dimension does not present a new idea or additional feature to consider adding to a learning experience. Instead, it highlights an existing consequence of teaching. By noting feedback events in the timeline of a session plan and considering the apparent risk the activity requires, it can be

possible to tailor a session to the needs of the learner and indeed the material. Where learners are unfamiliar with material or the material is particularly error prone, a tight *Risk Reward* cycle is desirable. Where learners have growing confidence and ability, a greater degree of creative freedom afforded by a loose *Risk Reward* cycle is more suitable.

Chapter 10: Learning Dimensions Part III. Practicalities

Practicalities: Overview

Finally, **practicalities** describes some practical decisions to be made relating to the learning experience as a whole. This section presents two dimensions: *Grouping* and *Session Shape*. *Grouping* describes the possible arrangements of learners. *Session Shape* describes the affordance of the physical environment and how this may enhance or impede the learning experience.

Grouping

Description

The *Grouping* dimension draws attention to the different arrangements of learners that are possible. Throughout the studies conducted, three natural groupings of learners were noted: individuals, pairs, and groups of more than two people. In addition, there have been situations where there have been asymmetric groups in which learners worked with parents or with learners of different abilities. The *Grouping* dimension considers groups at the following four levels: Individual, Pair, Team and Asymmetric collaboration. Characteristic of each will be described next.

Individual

In the individual circumstance, learners are working independently. Their thoughts, ideas, and solutions are their own. There is no need to defend or communicate their thinking. There is also no potential to obtain peer feedback or inspiration from the ideas of other team members. There are always learners who desire to work alone (Martin and Hughes, 2013). This is unsurprising for at least three reasons. Firstly, the additional effort associated with being in a group can be viewed naively (or accurately) as wasted effort if the same work can be achieved solo. Secondly, group projects often have tensions between individuals who are not felt to be contributing to work and

those who are contributing well (e.g. Karn and Cowling, 2006). Finally, situations have been observed where groups explore a number of possible routes for a project but inevitably, many of these ideas must be rejected.

Pair

In the pair circumstance, the learner is working with a partner of similar background, for example, where two school pupils are working together. In this *Grouping*, solutions can be arrived at jointly, and communication and negotiation become important aspects of the learning experience. The ability for the pair to communicate effectively and work as one will influence the learning experience. Pair programming has become more common in an educational context (Simon and Hanks, 2008; McDowell et al., 2002), as it has been increasingly adopted in industry through implementation of the Agile development process (Begel and Nagappan, 2007).

Group

In the group circumstance, each learner is working together with at least two more two learners. Solutions are arrived at through negotiations within the team. The final product should result from the contribution of each group member. The size of the project that can be produced is larger than with solo projects, since there are multiple people working on the problem. There is an increasing need for good task management and it may be valuable to have clearly defined roles. Topping and Ehly (1998) describe distinctions between a collaborative group where all participants are working together and a cooperative group that implies a division of labour with each group member working on a different aspect of the project. In all studies, groups have been cooperative with learners generally taking responsibility for a particular aspect of the activity.

Asymmetric-collaboration

In this circumstance, the group has a cross section of learners of differing abilities. It is typical for older learners to support and work with younger learners. The situation described in Driver

Shifting where the learner and the educator are working together is an example of an asymmetric collaboration. In Robot Dance in the Community (Chapter 5), the emergent group of parents and children are also an example of an asymmetric collaboration. Asymmetric collaborations may be in pairs or groups, and will require good communication skills and an understanding of the value of a broad range of experiences.

Working alone, interpreting the ideas of others, and presenting and defending your own thinking, will all have an impact on what is learned. Therefore, different group compositions undoubtedly can have an impact on learning for the individual, and ultimately they achieve different things. Working within one discipline will likely be quite different to asymmetric-collaborative and working as part of a cross-functional team. It is proposed that this may bring a greater sense of worth in the product but possibly at the cost of additional communication challenges. Throughout a given learning experience, it is possible that a *grouping* will be transient and can be adjusted to best fit the task in hand (Martin and Hughes, 2013).

Related Literature

There is a substantial literature exploring various approaches to group learning, including collaborative learning (Dillenbourg, 1999; Bagley and Chou, 2007), team based learning (Lasserre and Szostak, 2011; Michaelsen et al., 2002), cooperative learning (Beck and Chizhik, 2013) and Peer Learning (Topping, 1998). Topping (1998) offers an excellent overview and discussion of the various approaches to learning with the support of others.

Rationale

Grouping is an important aspect of learning experience design as it offers the ability to provide learners with support. It also mimics the real world setting they are being prepared for. The same task approached by an individual, pair, group or asymmetric group will be tackled in quite

different ways and reflection around this is important. A balance must be struck between individual confidence and the potential learning benefits of a group setting.

Example from Fieldwork

In Robot Dance, learners were placed in groups of three or four. This was for the pragmatic reason of sharing laptop and robotic equipment. One resultant advantage of the group-working configuration was that learners were able to support each other: it is more difficult to become stuck in a group than if working solo. More than one person has seen the demonstration and each group member is able to contribute suggestions to any problems that arise. The second iteration of Robot Dance took place in the community in a public space, where groups were not forced; it was observed that self-selecting groups had a high degree of communication and collaborative reasoning. In the community setting where learners were not encouraged to form groups, it was interesting that different learners preferred to work in pairs or individually as well as in larger groups, suggesting that this is not just a function of the material but also of the individual learner.

In Whack a Mole, learners were also grouped. This was partly related to access to resources but was also informed by the observations made in Robot Dance. In Whack a Mole, the learners were older and the task was more complex. There were two test conditions, one in which learners worked with physical computing to make their game and a second one in which they used a screen-based simulation. In the physical setting, there was good scope for a division of labour between physical construction and development of the computer program required. This was a noticeable difference between the physical group and the screen-based group as the physical group had the additional tasks of constructing the game. The small groups worked well with this challenge to the extent that a range of skills across the team resulted in a successful outcome. It was noted that a successful completion of the task for the group does not necessarily mean that all learners in the group had competency over all the skills covered in the session.

In Digital Makers, the duration of the session was extended to a full day and a different approach was taken to *Grouping*. There were two distinct phases: in the first half, learners were acquiring new skills and experimenting with new technology. In the second part of the session, they were given space to experiment with these newly found skills and set a relatively open challenge. In the first section, learners were working as individuals, each with their own set of equipment. The rooms used were set up with islands of desks so chatting to one's neighbour was possible. This was to address the risk of learners missing important aspects of the teaching as other group members took control. The session was very interactive with lots of opportunity for learners to ask for clarification on any points that were not clear. In addition, as described for the *Driver Shift* dimension, there were many points that brought the groups of learners to a common level of understanding.

Application

When designing a learning experience it is importance to consider the potential for group work. The simplest and possibly most compelling reason for this is that very little programming work 'in the real world' is done in isolation. Irrespective of what the learners know about development, all of the nuances, complexity and untidiness of human interaction will have a substantial impact on how they transform their ideas into reality. Many of the motives for working in teams in work do transfer to a learning setting. As examples, problem solving, exploring ideas and critiquing solutions are difficult to do in isolation. It is wise for learners to experience the social complexity that working in a group brings through a learning experience. This needs to be balanced against the desire to support individual focus on a particular learning point. Switching groups can be a good way to reach a compromise, as achieved in Digital Makers. The duration of the session is an important consideration as switching groups is potentially disruptive, which can be either useful or harmful. As with all the LDs, they highlight and provoke reflection around the merits and shortcomings of a particular design decision.

Session Shape

Description

The physical environment encapsulates all elements of the space that learning takes place, including aspects such as the arrangement of tables and location of supporting visuals such as white boards or projectors. The physical environment is an important aspect of a learning experience (Brown and Long, 2006). Currently, as learners progress through nursery school, primary school, secondary school and on to Further and Higher Education, there is a notable shift in learning space design, from a flexible open activity specific space to the increasingly closed transmission-centred lecture theatre design. This correlates with a trend of increasing learner to teacher ratio and a perceived increased 'efficiency'. As learners mature and their attention span increases, the ability to consume and assimilate lectures increases (Wilson and Korn, 2007). An early but interesting study from chemistry education (Johnstone and Frederick, 1976) suggests that lecturer style has a relationship with attention span, though 15-18 minutes is typical. It is typical for educators to sight 20 minutes as the attention span in a lecture, this has however been scrutinise to show that it is more complex than simply lecture style and individual learner characterises play an important role (Wilson and Korn, 2007). However there is evidence that active learning is a powerful tool to engage learners and the use and design of learning space must reflect this (e.g. Hoellwarth and Moelter (2011), McConnell (1996), Prince (2004)). The physical environments involved in the studies and therefore reflected upon here are **classroom**, **public space**, **computing lab** and **informal learning space**.

Classroom: the classroom is a place in which learners will have a concrete association with engaging in the act of learning. The classroom will most likely have a physical layout to support learning with a defined point for the teacher (Betoret and Artiga, 2004). This focal point will likely be supported by a black, white or smart board. A layout with a natural focal point is one in which educators are more likely to gain and retain the attention of learners. Classrooms also can

offer a degree of flexibility so that furniture can be arranged in an appropriate fashion, for example to provide “islands” for collaborative group work.

Public space: public spaces are often used for outreach and public engagement activities. The rationale is simple: to get the learning to the widest possible audience, take it out of the school or university and take it to shopping centres, community centres, art galleries, science centres and coffee shops. This can affect the learning experience in a number of ways. For example, the location may be noisy and have many distractions. Simple strategies like facing learners away from sources of distraction can have a big impact the learners’ focus.

Computing Lab: a traditional desktop computer lab is a space set out with a number of computers, designed for learners to be working on computer-based tasks. This is likely to be a fixed environment as the computers will have static power and network needs. The layout of computer labs often is optimised for individual work, with rows of computers ideally suited to allocating one learner per computer. Engaging in group work in that circumstance can be difficult: communication with anyone other than the person immediately next to you can be problematic. More innovative approaches to computer spaces are emerging with mobile technology that is better suited to collaborative working, but which also can be more expensive.

Informal learning space: conference and science centres often have a range of flexible spaces for learning. They may have a dedicated classroom that is closed and similar to the classrooms found in schools. They may also have flexible open plan areas. When working in an open space with movable seating and tables, it possible to configure the space to meet the need of the session, such as if a focal point is needed for projection or if collaboration is required. One shortcoming of an open plan flexible working space is the potential for distraction and the cost of re-configuration.

Related Literature

Learning Space (Oblinger, 2006) is a compilation of a number of relevant articles describing and discussing the importance of space in Higher Education. A strong theme throughout the book is the relationship between the physical environment and its affordance for better educational practices. A given space in itself is inert and meaningless: it is how it is used or inhabited that will affect learning. Another theme that emerges is the importance of flexibility. This is tightly coupled to the shift from a knowledge transmission model of education to a more active mode of knowledge co-construction and active learning (Bonwell and Eison, 1991). A traditional lecture theatre is optimised for the projection of knowledge from the lecturer to a large audience of passive recipients. Learning Space documents a shift towards more flexible smaller spaces.

Rationale

Session Shape encourages reflection on the affordances and limitations offered by the type of learning space that will be used. Simple adjustments to the layout of room can have a notable effect on how the learners engage with the session. Unlike some of the other learning dimensions, this will be a feature of any learning experience, whether consciously considered or not. A holistic approach to learning requires consideration of all aspects of the experience.

Example from Fieldwork

The first set of Robot Dance studies were carried out on school visits. They took place in classrooms that generally had islands of desks that were appropriate to group work. In many situations the classroom was used by a single teacher and it was clear from detailed wall decorations and carefully laid out stations of equipment that a great deal of consideration and thought had gone into the layout of the classroom. In all sessions, the teacher of the class was present and this resulted in a very attentive well-behaved class. The layout of the class lent itself well to the high *Driver Shift* (see previous LD) mode of deliver that was utilised. Instructions

were given and the groups were required to carry out tasks independently. The delivery mirrored the type of learning that typically takes place and was therefore not disruptive to the expectations of the learners. The ratio of learners to facilitator was 24:1.

When Robot Dance in the Community took place, it was in a very different setting. The environment was a busy shopping centre in the days before Christmas: there were many eye catching decoration and shop windows. It was a 'drop in' event, which meant that learners joined and left the session when they wanted to and as equipment became available. Unlike the classroom activity, there was no collective start and end point. The learning space had to be adapted to meet the changing needs. Learners were given a quick introduction and a sample program to edit and play with, and then left to experiment. In this session, the ratio of facilitators to learners fluctuated between 1:1 and 1:4. With a smaller educator to learner ratio, it was easier to respond to a specific learner's questions. The physical environment for Robot Dance in the Community was very different to the classroom: two tables set up along one edge of three metre squared carpeted area. Learners began working on their program at a table but many opted to move their work down to the floor to reduce the time it took to test each new tweak to their program. The open space and small mobile netbooks allowed this flexibility.

Whack a Mole took place in a computer lab that had eight separate four-person islands of desktop computers. The groups were working in teams of three or four. The room was circular with no clear focal point. The layout was better than traditional computer labs that have rows upon rows of machines set up for individual work. The lack of a clear focal point did not present a problem in Whack a Mole, as the teaching was delivered by video tutorial. Although the tables were configured into "islands", collaboration was impeded by the presence of large desktop machines and widescreen monitors that obstructed communication across the table. The need to use the desktop computers removed any flexibility to use alternative breakout space with different seating that would have been better suited to collaboration.

In Digital Makers, different spaces were used in each of four difference cities (Dundee, Aberdeen, Glasgow and Edinburgh). In the Dundee session, the lab used in Whack a Mole was used yet again. The inflexibility of the seating did not present a problem since there were considerably reduced numbers for Digital Makers. Participants worked in pairs, which worked well on the desk clusters with computers and monitors dividing either side of the desks and serving to separate the pairs. In Aberdeen, a small lecture theatre was used. The seating was arranged in rows and although movable, the lack of space in the room left little scope for reconfiguration. This worked well for the taught part of the day where participants were coding along with a live demonstration. Good visibility of the two projectors was important. However, it did present a barrier to collaborative working for the more extended activity in the second part of the session. Many of the learners worked around this by turning chairs around.

The sessions in Glasgow and Edinburgh were hosted by the Glasgow Science Centre and Our Dynamic Earth respectively. Both provided very large open plan areas that were highly configurable. Islands of desks were placed in front of two projectors. There was sufficient space for learners to shift position depending on which part of the session they were engaged in. In the morning part, when visibility of projectors was necessary the learners shifted to one side of the tables. When the session switched to the creation of physical apps, which required more space for craft materials and collaboration, the learners spread out around the table. The projectors were no longer used so switching position made best use of the space.

Application

The *Session Shape* dimension serves as a placeholder to consider what constraints and affordances are offered by the space that you inhabit with your learners. Flexibility is the most desirable attribute for a learning space. In an ideal situation, a room will be have enough space to allow movement of learners as the session requires, as was seen in Digital Makers in Glasgow and Edinburgh. The ability to move furniture to suit the needs your session needs to be balanced

against the time this takes. Where the learning space is inflexible, it is often possible to improvise, for example turning seats where tables are fixed or using a larger room than needed to allow better location of learners. Adopting a holistic approach to learning, conscious decisions can be made about the available space and how to use that space to influence the learning experience. This enables teaching strategies to be matched up with the environment in which they take place.

Using the Learning Dimensions to Author, Reflect and Share Insights

The Learning Dimensions are set out as eight important areas for consideration in the design of engaging programming learning experiences. The three high level areas considered are: (i) design and delivery; (ii) tempo and rhythm; (iii) practicalities. These reflect a holistic approach to session design that considers the learner, the material being taught and the context in which the learning takes place. Each dimension comprises several parts: description, summary of related literature, rationale, examples from fieldwork and advice about its application. This structure was used since it does not tie readers into a linear route through the LDs: all the relevant information for each can be found in one place. It also encourages readers to consider theoretical and empirical underpinning of the proposed design considerations. Another advantage is that it offers practical examples of how the LDs have affected other learning experiences.

In the design of any new learning experience, almost certainly there will be a number of constraints and areas that the facilitator has freedom to make decisions about. Therefore, a two-stage approach is encouraged for the application of the LDs. One pass should be made through the LDs with notes made about aspects of the learning experience that are inflexible. For example, if special equipment is used this may impose learners to be in groups of four. The second pass should consider the LDs that are not constrained. If the constraints have resulted in known impediments, the facilitator can then plan how these can be addressed by the remaining LDs.

The LDs are not limited to be only a generative tool; they can be used to structure reflection about an existing learning experience. By considering each of the LDs against the learning experience, the educator may identify a number of valuable insights. They may highlight aspects of the learning experience they had not made a conscious design decisions about, such as the effect of the room layout on the learners. They may also identify LDs that are not evident in their learning experience. Finally, by providing a framework within which a learning experience can be discussed, the educator can readily share their insights and findings of how the application of LDs affected their learners. In much the same way that the Cognitive Dimensions (Green and Petre, 1996) provide a common vocabulary for the discussion and advancement of notation design, the LDs can provide a common vocabulary for the discussion of learning experience design.

Conclusion

The Learning Dimensions bring together literature, empirical evidence and offer guidance on how to apply these insights in practice. The desire from the outset was to equip educators with a lightweight tool that can support them in the creation of engaging learning experiences for computer programing. The overarching themes that are addressed by the LDs are (1) encouraging the educator to role-shift between teacher and facilitator; (2) the provision of strategies to support and enable learner growth towards independence; (3) provide a holistic approach to the design of engaging learning experiences.

The LDs provide insights for the creation of active learning experiences. A significant part of this requires the educator to be increasingly dexterous and deft in their support of learners. The traditional role of teacher is not obsolete, nor is it replaced by that of a facilitator. Rather the educator must be able to make transitions between roles, as circumstances require. Computer programing is detailed, complex and densely packed with interdependency. LDs would argue there is a role for a content delivery in supporting the teaching of computer programing, albeit perhaps not in the traditional sense. In Driver Shift, the intent is to deliver content in small

actionable chunks that the learner can immediately work with. This requires a switch in the educator from delivery of content to facilitator of learner-driven experimentation and self-discovery. This approach does not see active learning as a replacement for more traditional methods, but rather it is an extension and re-framing of existing methods.

The desired outcome for any learning experience is the acquisition of skills and knowledge by the learner, to the point that they are able to apply the skills and knowledge independently. The LDs highlight several approaches to the design of a structured progression for learner growth towards independence and autonomy. Creating learning experiences with a gradient from closed to increasingly open problems is one mechanism for the gradual increase in learner independence. This process offers a high degree of support when it is required and the gradual removal of structure at a pace that matches the skill mastery of the learner. Similarly with the LD *Space to Play*, larger time boxes can offer greater learner freedom and more loose cycles of LD Driver Shifting and LD Risk Reward, which are also strategies to grow learner independence. The educator can use LDs with the particular material they are covering to design a learning experience with the most appropriate combination of strategies to support learner growth. There is no formula: it relies on the experience of the educator to understand the need and interpret the best strategy for the learner.

The LDs are deliberately broad in what they cover. The motivation for this is simply ecological validity. Learning takes place within a rich and varied set of contexts and the LDs attempt to reflect this. There are many factors involved and the approach here considers the entire learning experience rather that to drill deeply down into one factor. This avoids focusing on one factor and thereby obscuring its interaction with another. The eight dimensions give a holistic approach to the creation of a learning experience. They consider social aspects of learning, implications for learning task design, features of the delivery as well as practicality such as *grouping* and the physical environment. Each is covered in sufficient depth to offer valuable insights to educators, without presenting too much detail that could present a barrier. There is no claim that this is a

closed set of LDs or that all possible insights for each LD have been documented. In contrast, this is presented as an initial version that should be used, updated, modified and improved by educators and researchers alike. The LDs should serve as a starting point rather than a conclusion.

Chapter 11: Working with the Learning Dimensions

Learning Dimensions are intended to be a lightweight tool that can aid in the design and refinement of learning experiences in programming. A direct aim of this research is to transform insights from the fieldwork conducted and literature into a tool that can be used by educators. The LDs web application has been written to help educators use the Learning Dimensions. Earlier in this research, an alternative model was devised: the Model of Programming Experience (MPEx). The MPEx was used as a generative tool to create two workshops that were evaluated by learners. This chapter will introduce and describe the main features of the LDs web application, using it to apply the LDs retrospectively to the first workshop.

This chapter first introduces the LDs web application, describing the main features. This is followed by a description of two workshops: Code a Kilt and Wee Beasties. The results of the workshops' evaluations are then considered, followed by a description of how the LDs web application has been used to reflect on the design of one of the workshops. This chapter closes with a reflection about the Learning Dimensions and how they may be developed in the future.

Learning Dimensions Web Application

The LDs web application comprises two views: the first presents information about the Learning Dimensions; the second gives a mechanism for educators to make relevant notes.

Figure 9.1 presents the first view, which was designed to enable educators to make notes related to each Learning Dimension. At the top is a collapsible banner that offers a very high-level summary of what the Learning Dimensions are. The buttons down the right serve as a menu to navigate the different Learning Dimensions. In addition to the Learning Dimensions, there is a button for 'details' that gives educators a space to save any supporting details relating the experience being designed. To provide a balance between flexibility and support, a link allows

the educator to add a text block to the text area using the following headings: title, class, learning aim and duration. This can be edited easily if desired.

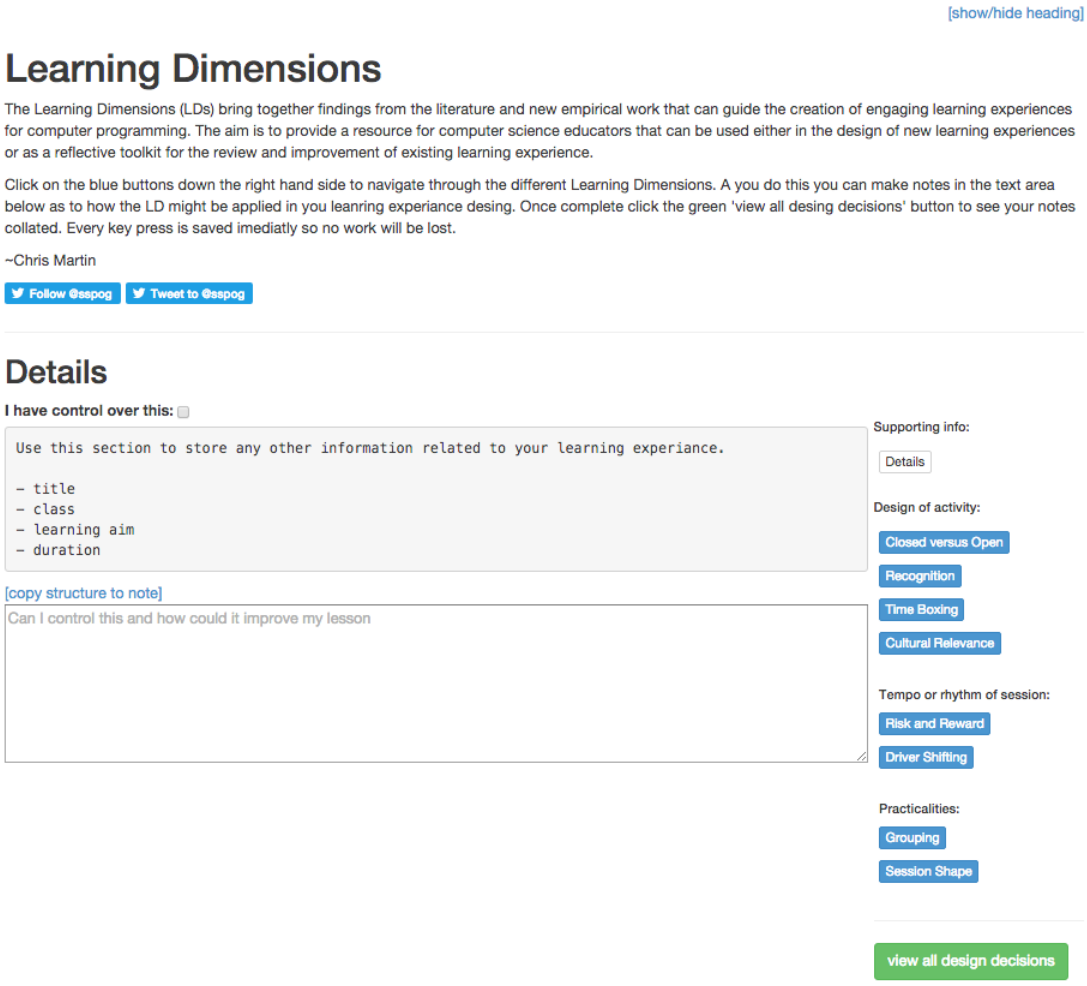


Figure 9.1: Learning Dimension web application: Edit View

This view screen of the application was designed to focus upon each individual Learning Dimension. The interface is minimal – no functionality is concealed – yet it brings together the items the user needs to address each Learning Dimension in turn. The page reads as the user would expect from top left to bottom right, with colour and sub-headings used to signpost the buttons to switch between Learning Dimensions. Figure 9.2 gives the collated notes edit view.

When a Learning Dimension button is clicked, the title and a brief description of the Learning Dimension are loaded. In addition, there is a check box to indicate whether the educator has control over this aspect of the learning experience. This implementation detail allows the educator quickly to review the Learning Dimensions and note areas over which they have control, as described in the 'Authoring, Reflecting and Sharing insights' section of Chapter 10. Below the description is a set of bullet points that describe different affordances of the Learning Dimensions and how they may affect the learning experience being designed. This is intended to aid the educator to reflect about how they might apply the Learning Dimensions to their learning experience. Underneath this description box is a text area where notes can be made; this can be expanded, if detailed notes are desired. Finally, there is a green button at the bottom right of the screen: this enables navigation to a view of all the Learning Dimensions notes together, as presented in Figure 9.3. This alternative view screen shifts the focus from the description of individual Learning Dimensions to the educator's notes and those Learning Dimensions over which they have control. This view allows one to 'take a step back' to get a sense of the shape of the entire learning experience and begin to think about how decisions relate to each other.

The web application was written in Angular JS (Angular, 2016) and Bootstrap (Bootstrap, 2016). It uses browser local storage that is cookie-based. Since there is no dependency on an external database, user data can persist throughout both browser refreshes and browser closing/re-opening. The text panel keypress event-handler was bound to a handler method, which means that every keypress is stored as it is typed. Coupled with the use of local storage, this means that there is no risk of the user losing any input, as every keypress will persist automatically without any direct action by the user.

The LDs web application is a first attempt at making the Learning Dimensions accessible to educators. Completing a large questionnaire can seem daunting. The LDs app uses current web technology to assist educators in working through their design decisions. By creating focus, providing contextual descriptions and dividing the supporting information, the findings of this

thesis should be more accessible to those who can benefit from them. Following a brief description of the two workshops and evaluations, the next steps for the web application and the Learning Dimensions will be discussed.

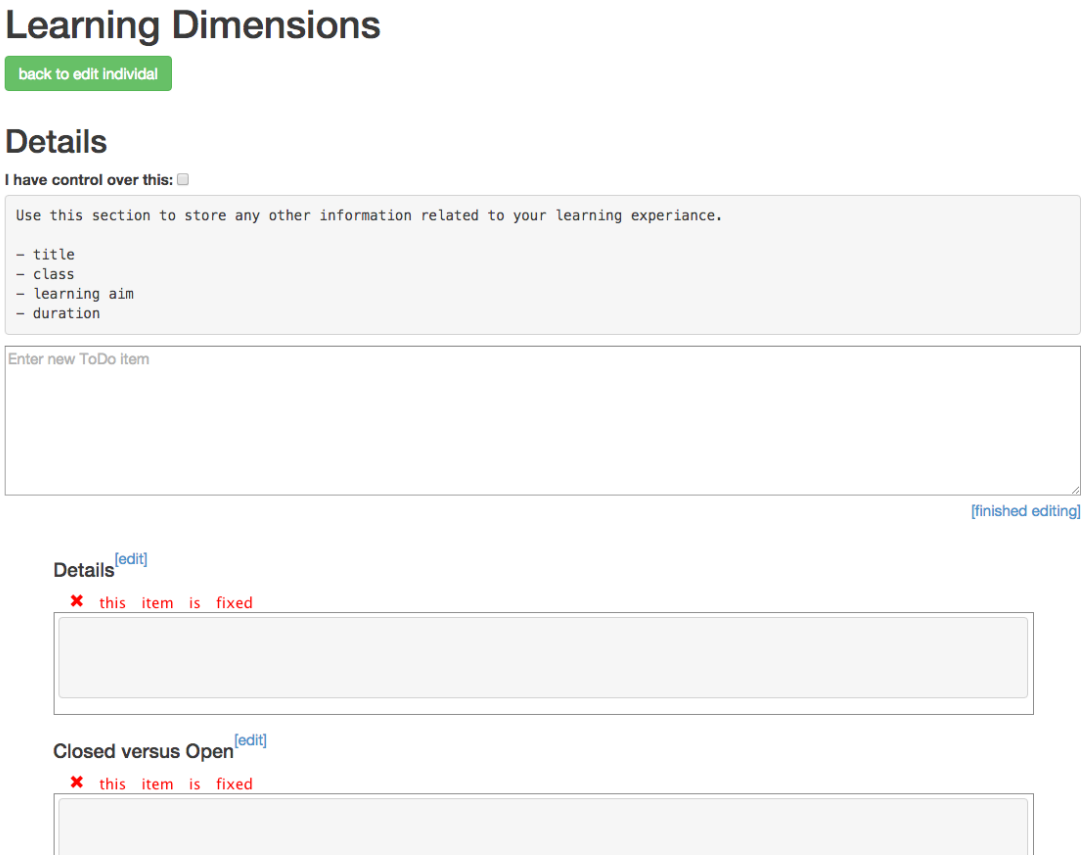


Figure 9.2: All LD Notes Collated Pop-up Edit View

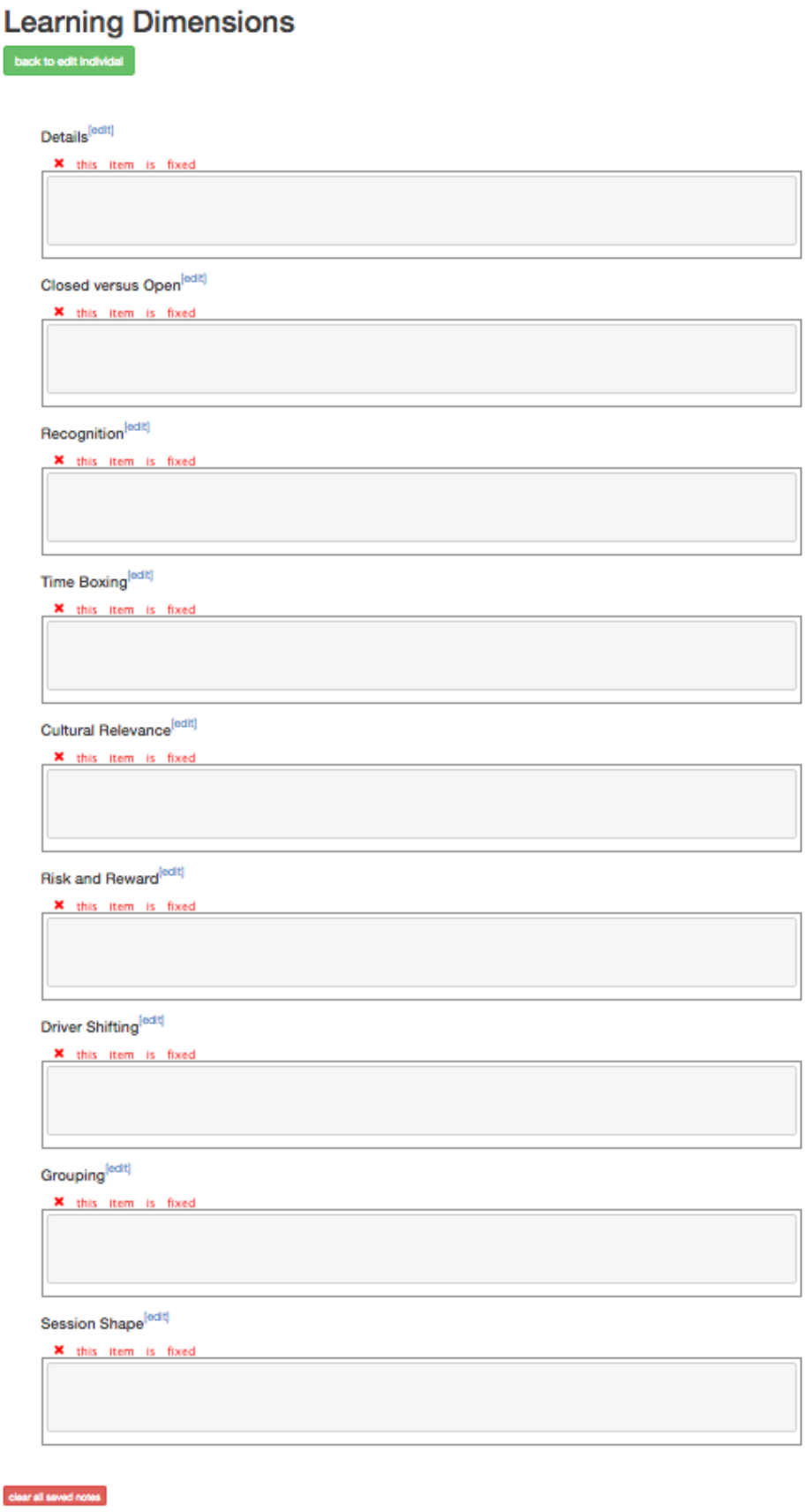


Figure 9.3: All LD Notes: Collated View

Code a Kilt Description

Code a Kilt was a computational tartan workshop. Learners were given an example program that uses functions and loops to create a tartan-like pattern. Code a Kilt walks learners through some basic computer programming competencies, framed by creating a computational tartan. It was designed as a tiered learning experience (Powers et al., 2006). With this design, there are layers of complexity that can be exposed to different learner. This enables the same experience to be delivered to a wide range of learners. Depending on a learner's ability and the depth they are able to go into, they will each take something slightly different from the learning experience. All learners start with a program that will draw a tartan when it is run. The workshop has been designed as a drop-in learning experience. As such, there is no predetermined structure to the event: learners are given a brief introduction and left to self-direct. This was the reason for giving learners a functioning program to experiment with, rather than instruction on how to create one.

The design of the learning experience was such that, for the most novice of learners, the primary learning objective is to gain an understanding of the sequence or stack of commands and the effect of this on the output tartan. The second learning aim is to demonstrate the ability to select colours using the colour picker that creates the function call required to set the colour in the program.

The basic program introduces three concepts: 1) setting the background colour, 2) setting the colour of the lines drawn and 3) a function to draw a grid of diagonal lines. To aid with colour selection, learners are given a colour picker that provides a number of colour swatches across the top and a colour space underneath, graduating the brightness and saturation. With a left click on a particular colour, the code required to set the line (or stroke) colour is copied to the clipboard and the learner can paste this into the programming environment. With a right click, the code to set the background colour is copied to the clipboard. This gives learners early access to syntactically correct code that has a known outcome, i.e. they select the colour and are given the parameters needed to reproduce it in their program. Following the selection of a line or

background colour, they are able to edit the colour by changing the parameters to make it brighter or more intense. This offers a good compromise between reducing syntax errors and retaining flexibility to code.

The ordering of the commands in the programming environment is also important, since it dictates the order in which items are drawn. As an example, if a learner were to move the background command to the end of the program, they would be disappointed when any overlaid grids were obscured by a solid background colour rendered on top of them, as the last instruction is drawn on top of all previous ones. Ordering of commends can be used positively as a feature in the workshop to layer up different grids and observe various visual effects.

For more advanced learners, the learning aims are extended to include editing of parameters passed to the grid function. The grid function takes four parameters: the `x` and `y` position at which the rows and columns begin, the spacing of the grid and the thickness of the grid lines. By adjusting these parameters, the learners experiment and receive visual feedback for different settings. Where learners have gained an understanding of the program structure (for example, pick a colour and draw a grid), they will likely extend the two-grid base program to have multiple grids.

For the most advanced learners, the learning aims extend further to understanding associations between function calls and function declarations as abstraction devices. The more advanced learner can begin to understand how the tartan is drawn further by interrogating the various commands used. By examining the `grid` command, for example, the learner can begin to understand how a tartan is constructed and grasp the hierarchy of commands. Two further commands become exposed: the `row` and `col` command. Each of these draws a row or column comprised of 45-degree diagonal lines, to form a hatched row or column. When a series of these are drawn across and down the screen, a uniform grid is formed.

Once the learner has sufficient time to produce a tartan they are happy with, it is then printed out as an A6 postcard. In addition to getting their tartan on a postcard to take away, a second postcard is produced and stuck up on the wall. As the day progresses, this collection of tartan postcards grows and provides an opportunity for learners to inspire each other.

Wee Beasties Description

Wee Beasties is a paper electronics workshop that allows learners to gain an understanding of some elementary computing concepts in a tactile learning experience. Using Bare Conductive Electric Paint (Bare Conductive, 2014), circuits can be painted onto paper or other surfaces. This workshop was designed to create a tiered learning experience (Powers et al., 2006) for a wide range of learners. It blends required competencies with more open-ended elements. A simple circuit was designed that uses two LEDs, a battery and a set of patch cables points. It was screen-printed with Bare Conductive paint. The circuit was printed on the back of a postcard-sized piece of white card (Figure 9.4: bottom). This formed a paper equivalent of a silicone printed circuit board found in most modern electrical devices. The reverse side of the card was left blank for learners to illustrate their Wee Beasties around the LED “eyes”. The LEDs were positioned in such a fashion that learners could choose to create a portrait or landscape Wee Beastie.

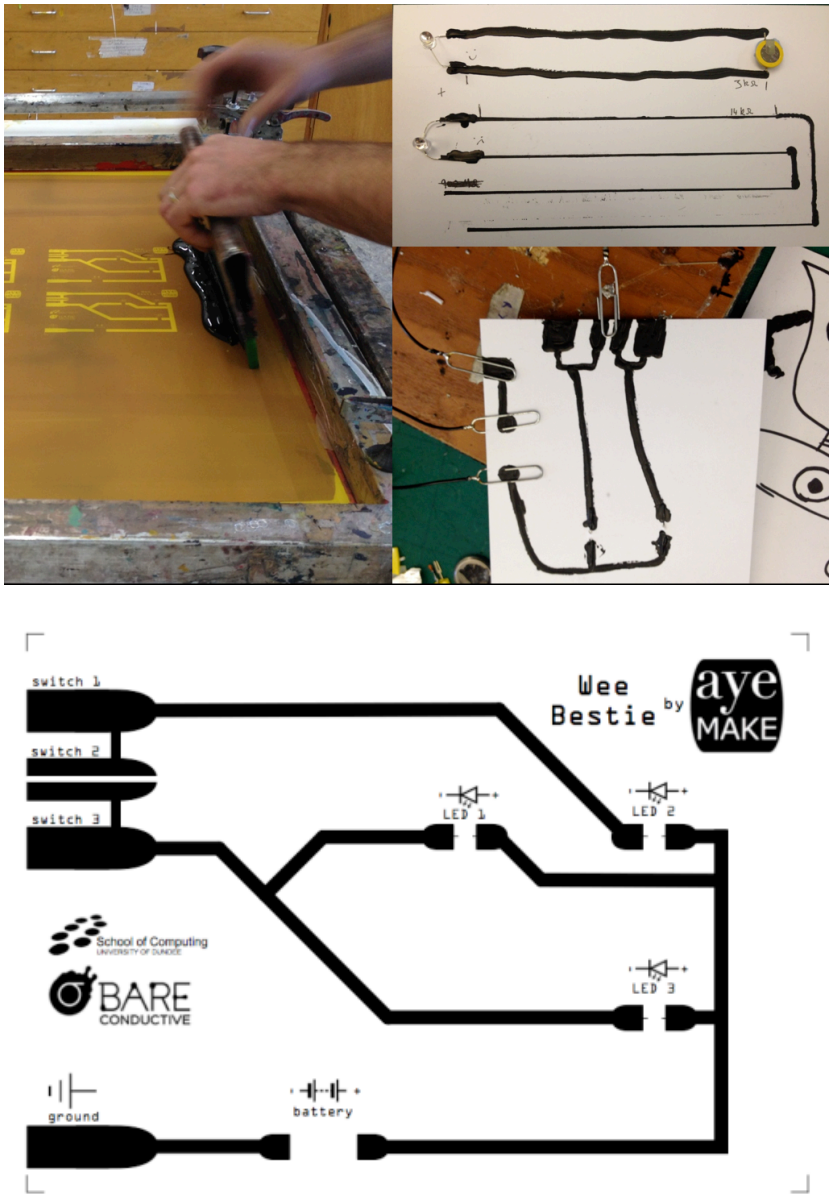


Figure 9.4: Wee Beastie Printed Circuit Development

The card circuit requires the learner to pierce holes into which the LEDs can be inserted (Figure 9.4: top). The learner must then insert the LEDs and position the battery, paying attention to the polarity of each component. When all components are in place, a Bare Conductive pen can be used to provide a small amount of conductive paint to act as a cold solder joint. The final step is to use the paper clip jumper cable to experiment with different connections.

For the youngest of learners, the primary learning aim is a physical construction task. The fine motor control to pierce holes in the card and locate the LEDs and battery will present a challenge. It is probable that younger learners will require assistance identifying the correct components and particularly the polarity of components. Once the build has been completed, the learners are able to use the paper clip jumper cable. By sliding it to different contact points, different configurations of LEDs may be turned on. This is effectively a very simple `if` statement. The output is modified by placing the electrical contacts in different configurations.

For older learners, the learning aim extends to component identification and understanding of electrical symbols and orientation. The syntax involved in component identification is similar to that of textual or visual programming. Electrical circuit construction requires all the accuracy of computer programming. Learners were also required to trace the circuit to enable them to choose how to produce a landscape or portrait beastie.

Evaluations

The workshops were run in parallel on consecutive weekends, firstly in the Mini Maker Faire (Maker Faire, 2014) (MMF) and then in the Overgate shopping centre in Dundee city centre (OG). In both sessions, there were four stations set up for each activity, with similar levels of support for each. The MMF was open to the public from 10:00 to 16:00; approximately 100 people attended. The OG stall was open from 08:00 to 18:00. The OG event had a good level of attendance by 10:00. It was notable that there were very few children in the OG accompanied by parents until after lunch. The flow of the workshops worked well throughout both days, with no need for adjustment or amendment from one event to the other.

Evaluations were completed by 50 learners across two weekends, with approximately the same number of responses for each workshop: Code a Kilt (26 learners) and Wee Beasties (24 learners). Informed consent was obtained from parents or guardians where participants were under 16. Table

9.1 provides full detail. Both workshops attracted a good mix of male and female learners. There was a wide range of different ages, but with different distributions. The most prevalent age category in Code a Kilt was over 18s (42.3%), closely followed by six to ten year olds (38.5%). In contrast, the best represented age range for Wee Beasties was the six to ten year olds (33.3%), closely followed by under-fives (29.2%). Evaluations were completed by 72% of the 26 Code a Kilt participants and around 40% of the 24 Wee Beastie participants.

Table 9.1: Descriptive Statistics for Code a Kilt and Wee Beasties

	n	**gender**		**age**			
		male	**female**	**>5**	**6-10**	**11-18**	**>18**
Code a Kilt	26	50	50	3.8	38.5	15.4	42.3
Wee Beastie	24	45.8	54.2	29.2	33.3	16.7	20.8
combined	50	48	52	16	36	16	32

The evaluation sheets for Code a Kilt (Appendix VI) and Wee Beasties (Appendix VII) asked participants to mark, with a vertical line on a scale, the extent to which they agreed with the given statements (Table 9.2). Seven of the nine evaluations with statements about the learning experience were positive; two control statements were neither positive nor negative. Control statements were included to identify if participants were reading the statements and considering them individually.

The questions were chosen to obtain a measure of how the learners felt about participation in the workshop and some of the design decisions made. Code a Kilt helped participants to generate a personalised tartan via computer programming, and the evaluation responses to this were very strong. Learners felt they had learned something new. They responded strongly to questions related to ownership and being able to take away the product of their learning. The visual output from the textual code was also reported as helpful: seeing the image change in relation to the code change was valuable. The two control items were noticeably different to the other items on the

survey, which indicates participants were reading and reflecting on their responses. The only item that was reported less favourably was the extent to which they think friends may find what they had made interesting. This may indicate there was more value in the creation and control over the tartan than in the finished product. It does indicate that the personalisation element was important.

The results for Code a Kilt show that there was a high degree of agreement across the visual analogue scale items (Aitken, 1969). The scale ranges from zero, which indicates the highest degree of agreement, up to 50, which indicates the highest degree of disagreement. Table 9.2 gives the questions that were used in the evaluation and the mean visual analogy scale responses from participants.

Table 9.2: Code a Kilt Evaluation

	Question	Score
1	I feel I have learned some new things at this workshop	5
2#	Being in a busy environment with lots going on helped me work.	24
3#	I find it easier to do this sort of thing in the morning rather than the afternoon	21
4	I enjoyed designing and programming my own tartan.	2
5	I'm pleased I can take away the thing I have made.	1
6	I liked that the computer program I was making produced something visual.	4
7	Seeing the tartan change as I edited the code helped me understand the program.	3
8	I think my friends will be interested in seeing what I have made.	13
9	I would recommend this workshop to a friend.	9

visual analogue scale range 0 = agree 50 = disagree; # indicates control

Two control items (questions 2 and 3, marked #) were included in the evaluation to determine to what degree the participants were giving similar positive answers without reading the questions.

The Wee Beasties exit survey produced similar results. As in the previous survey, two control items were included and there is a difference between the controls and the other items in the

survey. All other items were reported with strong agreement. Learners felt they had learned something. There was agreement that the open aspect of designing the face or picture was desirable. Participants also enjoyed assembling their Wee Beastie. It is interesting that there was no difference between a desire to share the artefact and recommend the workshop to a friend. Perhaps the unique nature of the artefact made it more interesting to share and less coupled to the act of personalisation and the process of making. In Wee Beasties workshop, participants were also happy they had a tangible artefact to take home.

Table 9.3: Wee Beasties Evaluation

	Question	Score
1	I feel I have learned some new things at this workshop.	3
2#	Being in a busy environment with lots going on helped me work.	20
3#	I find it easier to do this sort of thing in the morning rather than the afternoon	15
4	I enjoyed creating and decorating the face of my Wee Beastie	2
5	I'm pleased I can take away the thing I have made	3
6	Putting all the physical electronic parts in the correct place was fun	6
7	Building my wee beastie helped me understand how electronics work.	6
8	I think my friends will be interested in seeing what I have made.	7
9	I would recommend this workshop to a friend	5

score VAS range 0 = agree 50 = disagree; #indicates control

The studies in this chapter were designed before the Learning Dimensions had been created. In the next section, the Wee Beasties workshop was reviewed retrospectively using the web application. Illustrative extracts from the web application are given in italics. Firstly, the Learning Dimensions were considered to identify what the workshop designer was able to control and what was constrained, either by the nature of the task or the event of which it was a part. Secondly, each Learning Dimension was considered individually and notes made about how the workshop related to that specific Learning Dimension. Finally, a discussion of the result is given.

Web Application to Reflect on Wee Beasties

In the Wee Beasties workshop, three of the Learning Dimensions were constrained or limited by the nature of the task and the event: (i) *Recognition*, (ii) *Cultural Relevance* and (iii) *Grouping*.

(i) A combination of the nature of the task, making a physical artefact and the format of the larger event meant that the *Recognition* Learning Dimension was constrained.

> *The nature of the task limits the scope for recognition. Learners were creating physical artefacts that they were able to take away with them after they had made them. Digital stills were taken and printed to build a collage of the Wee Beasties as the day, but no recognition was offered beyond that.*

Some efforts were taken to gain some *Recognition* by sharing images on the day, but this was somewhat limited.

(ii) *Cultural Relevance* was also identified as limited. The short, variable time that participants were engaged in the workshop made it tricky to personalise:

> *As this is a drop in event with a short and variable amount of time to work with the learner the scope to give space personalise the activity was limited. Giving the learners choice over what design to draw offers some personalisation and ownership.*

(iii) *Grouping* was constrained by the drop-in nature of the event. As it was possible to give learners a Wee Beastie of their own, this was not a problem. The design was flexible to allow learners to work in many groupings:

> *There was no upfront control over this due to the nature of the event. We did observe learners working individually, one Wee Beastie per learner. There were also some asymmetric groups with parents collaborating with children.*

The remaining Learning Dimensions were not limited by the nature of the task or event and could be used to influence the design of the session. The workshop had elements that were both *open and closed*:

> *Open elements: part of this activity involves the design of a face or design that incorporates flashing LEDs. This is very open as learners can design anything they wish.*
>
> *Closed elements: the circuit the learners use was screen-printed and thus pre-defined. This constraint limits creativity but enables the workshop to be delivered to a wide range of learners as the challenge is understanding a 'thing that is' rather than creating something new which is a higher order task in terms of Bloom's taxonomy.*

Elements that were closed were chosen to support learners' lack of experience with the task. To offset the closed element, a 'softer' open element was included to allow learners to have some control over an aspect of the learning experience.

Space to Play was identified in the learning experience design with two notable time boxes:

> ***Space to Play launch**: introduce activity, choose which LED to include in design. **activity**: make holes for LEDs and draw and colour picture. **landing**: review and reinforce what has been created. **Space to Play launch**: describe how to use the conductive paint to form connection between the printed circuit and the components, activity: hook up components. **landing**: confirm it is working*

The workshop naturally split into two activities that learners could perform independently, firstly designing their artwork and secondly hooking up the electronics. As the session was not organised into groups with a defined start and end point, the full value of launching and landing was not experienced by all learners. This aspect of *Space to Play* is particularly valuable when encouraging a group of learners to diverge after the launch and then converge, sharing findings at the landing.

Risk Reward was described as of less importance in this workshop, primarily due to its short nature and the very low educator: learner ratio. In a workshop with a high ratio of educator to learner or with a greater duration, there is a greater risk of learners investing effort incorrectly.

> *The duration of this activity makes* Risk Reward *less important. There were two cycles of instruction giving with facilitation and assistance given to learners on demand. Small numbers enabled a rich degree of communication between the facilitator and the learners and there was very little scope for learners to acquire risk by making significant efforts that were incorrect.*

Driver Shifting was observed as described below:

> *This was variable across learners and in many ways affected by grouping also. Where learners were independent, there were two cycles of Driver Shifting with learners working independently following instruction. Several of the younger participants were assisted in an asymmetric collaboration with parents or older siblings so there was a transition of driving among the group and from facilitator to parent and child.*

The *Session Shape* had a good degree of control, since it was delivered in an environment that was created especially for the event.

> *There was lots of control over this as the event took place in an open hall and shopping centre. Tables were arranged in a single island to enable all learners working on the task to chat and share ideas easily. The facilitator was able to grab focuses as need and work 'over the shoulder' of learners. As the event was drop in, learners were not in cohorts and were few in number, there was no problem in grabbing focus of the whole group when needed.*

Discussion

The web application served as a useful tool with which to reflect upon the design of the Wee Beasties workshop. It has made the Learning Dimensions accessible and demonstrated how they can provide a vocabulary to expose some useful insights into how the successful workshop was

created. Its application to the Wee Beasties workshop helped to identify a number of further opportunities and ideas.

This application of the Learning Dimensions via the web application offers a concrete example of how they related to learning experience design. They provide a vocabulary and framework for the description and discussion of learning experiences. The web application provides an easy route to populate this framework. What this makes possible is the sharing of experience and practices without the expense and barrier of having to construct a taxonomy of learning experience features on a case-by-case basis. The first limitation is that it was only applied by its author, who has a detailed working knowledge of the Learning Dimensions: there was no verification by any other educator. The descriptions and in particular bullet points provided for each Learning Dimension served as a valuable prompt but it is not yet clear if additional or alternative material would be required to support educators who are unfamiliar with the Learning Dimensions. The next natural step and interesting piece of future work would be to conduct a study with educators, both those working on the design of new lesson plans and those planning to reflect on existing work.

For the LDs to have value to educators, they must be understandable and easily applied in practice. Educators need to be able to take ownership of the Learning Dimensions and drive their further development. This could provide motivation, in a way similar to that found in early programming being driven by sharing work on community forums. It is proposed that a future version of the Learning Dimensions web application could also be offered on community forums, to offer educators a greater degree of ownership over the Learning Dimensions.

Several exciting possibilities could result from the creation of a fully featured web application that supports multiple users and persistent centralised storage of Learning Dimensions. Educators could create a catalogue of learning experiences that could be shared with the community. The Learning Dimensions learning experience descriptions could support a feedback mechanism for

any session designed, shared, and then reflected upon after the session is delivered. This would support evidence-based practice and give educators an ability to track progress and understanding of how the Learning Dimensions can be applied 'in the wild'. With a community and a common vocabulary that affords a shared understanding, it becomes easy to share ideas and generate insights. Educators could receive feedback on the success of learning experience plans when applied in different circumstances.

With this type of system, there is value for both the individual and the community. A database of Learning Dimensions learning experience descriptions and feedback from multiple educators about the delivery of the sessions would make a very rich research resource, which in turn could feedback to advances and further understanding of the design of learning experiences.

Conclusion

The web application represents a first step towards moving the insights derived from this thesis into the hands of educators who can enhance the learning of computer programming. The evaluation is limited in its nature but offers support for the case that the Learning Dimensions represent a useful tool for the design and discussion of learning experiences. There is no claim the Learning Dimensions are a closed set; on the contrary, this exercise has raised a case for a learning dimension relating to session type, as many of the constraints observed in this evaluation were a direct result of the drop-in public nature of the event. It is hoped that future research in this area will add weight and understanding to how the original eight can be applied and pave the way for new learning dimensions.

Chapter 12: Conclusion

Introduction

This thesis presents empirically explored Learning Dimensions that highlight a number of key decision areas for the design of engaging learning experiences in programming. This chapter summarises the main contribution and findings of this research. The research questions in Chapters 4, 5, 6 and 7 are revisited and conclusions drawn about how best to enable engaging learning of computer programming. Finally, areas for future work are identified and described.

Main Thesis Findings

The main contributions of this thesis are best described as four parts. The **first** contribution is a critical review of related literature including: (1) novices and the challenge of learning to program, (2) tools to support learning to program and (3) motivating learning. This describes the current understanding of the challenges faced by learners of computer programming as well as current approaches to assist learners with these challenges.

The **second** contribution results from the four pieces of complementary fieldwork that describe and evaluate novel approaches to teaching programing in messy but ecologically valid settings. A combination of quantitative and qualitative methods provided insights into how to design engaging learning experiences in programming. Novel methods were devised to (i) make qualitative judgements about the work of learners creating physical apps and (ii) capture the emotional experiences of learners immediately after a programming experience. The former provided evidence of the very positive emotions experienced by learners developing physical apps. The latter further developed the HUMAINE project's system to allow identification of links between the emotions experienced and their origin. Robot Dance demonstrated a significant learning effect from a short robot programming experience. Robot Dance in the Community extended this to gather insights into how learners self-organised. Whack a Mole extended these

insights to discover that there is little learning effect difference between learners working with physical artefacts and those working with screen-based artefacts. Finally, Digital Makers leveraged everything that had been learned to create a highly engaging and successful learning experience that included a layer of *Cultural Relevance* to the learning tasks. These insights have been synthesised with the literature to create the third contribution.

The **third** contribution is the Learning Dimensions. The purpose of Studies I, II, III and IV was to design and evaluate novel learning experiences, and the finding of these studies have been located in the literature and used to inform the identification of the Learning Dimensions (described in detail in Chapters 8-10). In total, there are eight different areas for decision described in the LDs. The overarching themes they address are (1) encouraging the educator to role-shift between teacher and facilitator; (2) providing strategies to support and enable learner growth towards independence; (3) adopting a holistic approach to the design of engaging learning experiences that is independent of any language or environment. The LDs are not intended not as a formulaic tool to generate learning experience. They are intended to be a set of well-grounded, plainly described, accessible design guidelines for educators. By highlighting a small set of key decision areas and offering fieldwork-based examples as well as links to supporting literature, the LDs can guide educators to create engaging learning experience for a wide range of learners.

The **fourth** and final contribution of the thesis is the presentation and evaluation of a web application for the LDs (Chapter 9):-

- *Closed versus Open*
- *Cultural Relevance*
- *Recognition*
- *Space to Play*
- *Driver Shifting*
- *Risk Reward*
- *Grouping*
- *Session Shape*

In Chapter 9, the web-based tool was presented: it allows educators to make notes against each of the LDs. Each LD is first considered individually with supporting notes that describe the LD and

bullet points on how they may apply to a learning experience. Secondly, all educator notes can be viewed together to support more holistic shaping of the learning experience. The LDs web application has been retrospectively applied to Wee Beasties, a conductive paint workshop. This utilised the LDs as a framework to describe the workshop. Table 12.1 provides a matrix that cross-references the LDs with the studies in which they were observed.

Table 12.1: Learning Dimensions Cross-Referenced to Studies and Research Questions

		Closed Versus Open	Cultural Relevance	Recognition	Space to Play	Driver Shifting	Risk Reward Cycle	Grouping	Session Shape
Robot Dance	How does use of a physical robot in an activity support learning of introductory programming?	x		x	x		x	x	X
Robot dance in the community	Given freedom in a programming activity, how do learners organise themselves?	x			x		x	x	X
Whack-a-Mole	How does working with a physical or screen based artefact affect learning to computer program?	x			x		x		X
Digital Makers	How does personalisation, ownership and purpose in an activity affect introductory programming learning?	x	x	x	X	x	x	x	x

The research questions will be reviewed in the next section. The closing section identifies some interesting opportunities for future work.

Summary of Learning Dimensions

Closed versus Open

Learning experiences may be designed to contain a number of tasks or activities. The *Closed versus Open* dimension encapsulates the extent to which these activities have a well-defined structure, route and end point. A good example of a closed problem is programming a robot to follow a line. The task defines the answer: there is little scope for the learner to take ownership. Towards the open end of the dimension would be a free choice activity where learners are able to demonstrate competency in a given skill through the creation of a piece of work that is not constrained by the educator. An example is creating a robot dance.

Cultural Relevance

Often part of a learning experience involves creating a product of some kind, such as code or a sketch. The *Cultural Relevance* dimension considers where this product sits within the learner's culture. It prompts consideration of whether or not the tasks they are asked to perform are authentic and relevant to their daily life experience. Ownership, personalisation and purpose are key aspects of creating a learning experience that will have high cultural relevance for the learner. If the learning experience is divorced from the world the learner inhabits, the cultural relevance will be low.

Recognition

It is typical for a learning experience to result in the generation of a product. It may be a program, a sketch or a concept. The *recognition* dimension considers the potential for the learner to share the product of their learning. As early as nursery school, learners seek recognition from their

teachers, peers and parents. A good example of this is pleasure gained from the displaying of work on the walls of the learning environment for all to see. In the Learning Dimensions context, a model of *recognition* has three parts: (a) the mode of the interaction, (b) the audience size, and (c) the Depth of the interaction. Each of these, when considered together, will have an effect on the learner's engagement and motivation.

Space to Play

The *Space to Play* dimension seeks to break down the traditional view of a teacher-learner relationship. It encapsulates the extent to which a learning experience offers and encourages learners to explore independently, experiment and iterate over aspects of the learning experience. This idea is rooted in constructivist learning theory (Papert and Harel, 1991): learning takes place best when learners engage in project work that results in an artefact that is relevant to the learners, as described in the *Cultural Relevance* learning dimension. *Space to Play*, however, addresses the fact that space and independence may be intimidating for certain learners. Furthermore, it acknowledges the tension between the learner as an individual and a need to cover a particular amount of content with a group of learners. Where space can be intimidating, direction, constraint and facilitation can be catalysts to creativity and learning.

Driver Shift

Driver shift is a new concept that has emerged from the studies conducted. This dimension attempts to capture which actors in the learning experience are driving, i.e. taking control of the learning experience at a given point in time. It is likely that there will be transitions between learners and facilitators as drivers throughout a session. There can also be a mid-state in which collaboration between the learner and the teacher takes place. For example, a classic higher education style lecture where the lecturer projects content to the learners for a sustained period would be regarded to have very low degree of *driver shift*. In contrast, a guided practical session

with a tight cycle, in which learners are shown a brief example and then given space to try it, would be said to have a high degree of *driver shift*.

Risk Reward Cycle

The *risk reward* cycle considers the relationship between the investment of effort or risk that a learner undertakes and the reward when feedback is received. Investment of effort without confirmation that the correct actions are being taken by the learner is considered a risk. This is because they may result in wasted effort or even worse enforcing an incorrect understanding or application of a skill. Feedback can take a number of forms, such as observation and direction from a teacher or the completion of a complete program that can be executed. For example, this could be the time taken to write a hello world program. For Java, the amount of effort investment required from the learner to get the payback or reward of some text being displayed is non-trivial, so high risk. In a language like Processing, the effort investment made by the learner before observable outcome is much shorter. In Processing and other scripting languages, it is possible to render output in one line of code, so lower risk. This programming effort is considered a risk to a learner, since independent work in a particular direction for a given period time without feedback has a chance that the learner has moved in the wrong direction. In any learning experience, there will be a cycle of learner effort investment and pay-off as the learner works through different tasks and receives feedback as they progress.

Grouping

The *grouping* dimension draws attention to the different arrangements of learners that are possible. Throughout the studies conducted, three natural groupings of learners were noted: individuals, pairs, and groups of more than two people. In addition, there have been situations where there have been asymmetric groups in which learners worked with parents or with learners of different abilities. The *grouping* dimension considers groups at the following four levels: Individual, Pair, Team and Asymmetric collaboration.

Session Shape

The physical environment encapsulates all elements of the space that learning takes place, including aspects such as the arrangement of tables and location of supporting visuals such as white boards or projectors. The physical environment is an important aspect of a learning experience (Brown and Long, 2006). Currently, as learners progress through nursery school, primary school, secondary school and on to Further and Higher Education, there is a notable shift in learning space design, from a flexible open activity specific space to the increasingly closed transmission-centred lecture theatre design. This correlates with a trend of increasing learner to teacher ratio and a perceived increased 'efficiency'. As learners mature and their attention span increases, the ability to consume and assimilate lectures increases (Wilson and Korn, 2007). An early but interesting study from chemistry education (Johnstone and Frederick, 1976) suggests that lecturer style has a relationship with attention span, though 15-18 minutes is typical. It is typical for educators to cite 20 minutes as the attention span in a lecture, this has however been scrutinised to show that it is more complex than simply lecture style and individual learner characteristics play an important role too (Wilson and Korn, 2007). However there is evidence that active learning is a powerful tool to engage learners and the use and design of learning space must reflect this (e.g. Hoellwarth and Moelter (2011), McConnell (1996), Prince (2004)). The physical environments involved in the studies and therefore reflected upon here are **classroom**, **public space**, **computing lab** and **informal learning space**.

Research Questions Revisited

Study I, Robot Dance (Chapter 4) explored supporting introductory programming learning with programmable robots. It addressed the research question:

(Q1) How does use of a physical robot in an activity support learning of introductory programming?

A small learning effect was measured as a result of the workshop. Characteristics of Robot Dance that supported this are believed to be as follows. Learning in Robot dance was supported by the delivery of several small demonstrations followed by space for learners to experiment with the new skill. This ensured that new skills were always applied to avoid becoming inert (Perkins et al, 1986). Working towards a performance at the end of the session was a motivator for many of the learners. Robot Dance was first study of this work to recognise the importance of *Recognition*. One limitation was based around the arbitrary setting of learner groups, due mainly to equipment constraints. The next study explored the issue of grouping in more detail.

Study II Robot Dance in the Community (Chapter 5) observed learners working with no structure imposed on how they arranged themselves. It addressed the research question:

(Q2) Given freedom in a programming activity, how do learners organise themselves?

Learners were observed self-organising into a number of differently groupings. Learners were also observed using testing and debugging strategies and in some cases using external notations to support problem solving. One of the interesting findings in this study was evidence of emotional engagement in the task of programming the robot. The learners appeared to value the artefact they were working on and be highly motivated by taking control of this small object. . This finding fed into Study III.

Study III Whack a Mole (Chapter 6) was designed to explore the importance of working with a physical product as opposed to a screen-based simulation. It addressed the research question:

(Q3) How does working with a physical as opposed to screen-based artefact affect learning of computer programming?

Two similar groups of learners were exposed to identical learning tasks; the only difference was the product they were working with. One group produced a physical computing game, while the

other group programmed a screen-based equivalent. There was no advantage or disadvantage of either method that influenced a change in knowledge or understanding observed. However, there was a significantly different emotional response, with physical groups being more engaged with the task and experiencing a greater degree of positive affect. The next study pursued this result, creating a learning experience that was highly engaging for learners.

Study IV Digital Makers (Chapter 7) was designed to explore a more sustained learning experience that afforded learners greater control over the product of their learning. It also allowed learners a greater degree of freedom over the groups they worked in. It addressed the research question:

(Q4) How do personalisation, ownership and purpose in an activity affect introductory programming learning?

There was empirical evidence that the majority of learners engaged deeply in the rich context in which the learning was situated. This was coupled with a strong learning effect. The emotional profile reported by the learners was similar to that of study III, Whack a Mole. However, the difference between positive and negative emotions was more pronounced in favour of positive. This final study borrowed insights from the previous studies and reinforced their findings.

Future Work

Educator Evaluations

The main limitation of the evaluation carried out in Chapter 11 was that it was conducted by the author of the Learning Dimensions. This was a sensible first step in the evaluation of such a tool and confirmed its value. The logical next step would be to conduct a small evaluation of the Learning Dimensions web application with educators, giving them access to the tool over a period for them to use in the design of several learning experiences. Furthermore, it would be interesting

to see how educators could apply the tool to the evaluation of existing lesson plans. A diary-based study with exit interviews would generate valuable insights into how the Learning Dimensions have helped and where their limitations may be, as well as identifying any human-computer interaction flaws in the current design of the tool.

Publication of the Learning Dimensions

The primary motivation of developing the Learning Dimensions web application was to expose the findings of this thesis, sharing them with educators where they can have some impact. It is the intention of the author to publish and share the source code of the Learning Dimensions web application on GitHub to enable interested parties to access the Learning Dimensions and contribute to their evolution.

Learning Dimensions Web Application Part II

One of the most interesting insights to come from the evolution of the Learning Dimensions web application was the consideration of the value that could be generated by extending the LDs web application into a community-driven platform that supported storage, sharing and commenting on the descriptions of the Learning Dimensions. By extending the current tool to support multiple users and using centralised server based storage, a catalogue of Learning Dimension learning experience descriptions could be generated by a community of educators. This would result in a rich data set to evaluate refine and shape the future and existing Learning Dimensions.

Application and Advancement in Research

The Learning Dimensions were designed as distilled insights from existing literature and new research. The motivation was to support educators. The Learning Dimensions also present a useful framework for researchers who seek to study the learning to program. The Learning Dimensions are presented as version one of an open set of key decision areas. There is interesting future work

to be performed to provide additional research-driven detail to each of the Learning Dimensions and in proposing the inclusion of additional Learning Dimensions.

Conclusion

Creating an engaging learning experience is not a mechanical process that can be governed by a set of rules to be followed dutifully to guarantee consistent results. Learning experience design is a much more humanistic task. It requires reflection and consideration not just of what is to be learned but also of who is learning and how they can best succeed. Four studies of novel learning experiences described in this thesis have generated insights which are aligned with the literature and which have informed the creation of a set of Learning Dimensions. The Learning Dimensions should provoke thought about areas of important opportunity in the design of engaging learning experience: *Closed versus Open, Cultural Relevance, Recognition, Space to Play, Driver Shifting, Risk Reward, Grouping* and *Session Shape*. Finally, these Learning Dimensions have been made accessible to educators via a web application. This is a first attempt to help educators address Learning Dimensions to design and construct engaging learning experiences in programming.

References

AITKEN, R. C. 1969. Measurement of Feelings using Visual Analogue Scales. *Proceedings of the Royal Society of Medicine,* 62 (10), pp.989-993.

ALSMEYER, M., LUCKIN, R. and GOOD, J., 2008, April. Developing a novel interface for capturing self reports of affect. In *CHI'08 Extended Abstracts on Human Factors in Computing Systems* (pp. 2883-2888). ACM.

AMABILE, T. M., HILL, K. G., HENNESSEY, B. A. and TIGHE, E. M. 1994. The Work Preference Inventory: assessing intrinsic and extrinsic motivational orientations. *Journal of Personality and Social Psychology,* 66 (5), pp.950-967.

ANGULAR, 2016. [Online]. Available: https://angularjs.org/ [Accessed 2016].

ANDERSON, J. 1990. *Cognitive Psychology and its Implications,* W. H. Freeman, New York.

ANDERSON, R., MANOOGIAN, S. T. and REZNICK, J. S. 1976. The undermining and enhancing of intrinsic motivation in preschool children. *Journal of Personality and Social Psychology,* 34 (5), pp.915-922.

ARDUINO [Online]. Available: http://www.arduino.cc [Accessed 2014]

BAHRICK, H. P., FITTS, P. M. and RANKIN, R. E. 1952. Effect of incentives upon reactions to peripheral stimuli. *Journal of Experimental Psychology,* 44(6), p.400.

BALL, L. J. and ORMEROD, T. C. 2000. Putting ethnography to work: the case for a cognitive ethnography of design. *International Journal of Human-Computer Studies*, *53*(1), pp.147-168.

BANZI, M. 2012. How Arduino is open-sourcing imagination. [online] https://www.ted.com/talks/massimo_banzi_how_arduino_is_open_sourcing_imagination [Accessed 2014].

BARE CONDUCTIVE, 2014. *Creative electronic tools – Bare Conductive* [Online]. Available: http://www.bareconductive.com [Accessed 2014].

BECK, L. and CHIZHIK, A. 2013. Cooperative learning instructional methods for CS1: Design, implementation, and evaluation. *ACM Transactions on Computing Education (TOCE)*, *13*(3), 10.

BEGEL, A. and NAGAPPAN, N. 2007. Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. In *Empirical Software Engineering and Measurement, First International Symposium on*, pp.255-264.

BENNEDSEN, J. and CASPERSEN, M. E. 2007. Failure Rates in Introductory Programming. *ACM SIGCSE Bulletin,* 39(2), pp.32-36.

BETORET, F. and ARTIGA, A. 2004. Trainee teachers' conceptions of teaching and learning, classroom layout and exam design. *Educational Studies,* 30(4), pp.355-372.

BLANK, D.S. and KUMAR, D., 2010. Assessing the Impact of Using Robots in Education, Or: How We Learned to Stop Worrying and Love the Chaos. In *AAAI Spring Symposium: Educational Robotics and Beyond*.

BOALER, J., 1998. Open and closed mathematics: Student experiences and understandings. *Journal for research in mathematics education*, pp.41-62.

BONWELL, C.C. and EISON, J.A., 1991. *Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports*.

BOOTSTRAP, 2016. [Online]. Available: http://getbootstrap.com/ [Accessed 2016].

BOSCH, N. and D'MELLO, S., 2015. The Affective Experience of Novice Computer Programmers. *International Journal of Artificial Intelligence in Education*, pp.1-26.

BOSCH, N., D'MELLO, S. and MILLS, C., 2013, July. What emotions do novices experience during their first computer programming learning session? In *Artificial Intelligence in Education* (pp. 11-20). Springer Berlin Heidelberg.

BRAITENBERG, V. 1986. *Vehicles: Experiments in synthetic psychology,* The MIT Press.

BRAUN, V. and CLARKE, V. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology,* 3(2), pp.77-101.

BRENNAN, K., MONROY-HERNÁNDEZ, A. and RESNICK, M., 2010. Making projects, making friends: Online community as catalyst for interactive media creation. *New directions for youth development*, *2010*(128), pp.75-83.

BRESNAHAN, T. F. and TRAJTENBERG, M. 1995. General purpose technologies, 'Engines of growth'? *Journal of econometrics,* 65(1), pp.83-108.

BROOKS, R. 1977. Towards a theory of the cognitive processes in computer programming. *International Journal of Man-Machine Studies,* 9(6), pp.737-751.

BROOKS, R. 1983. Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies,* 18(6), pp.543-554.

BROWN, M. and LONG, P., 2006. Trends in learning space design. *Learning spaces*, pp.9-1.

CHI, M.T., BASSOK, M., LEWIS, M.W., REIMANN, P. and GLASER, R., 1989. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive science*, *13*(2), pp.145-182.

CHI, M.T., DE LEEUW, N., CHIU, M.H. and LAVANCHER, C., 1994. Eliciting self-explanations improves understanding. *Cognitive science*, *18*(3), pp.439-477.

COMPUTER HISTORY [Online]. Available http://www.computerhistory.org/timeline/computers/ [Accessed 2014]

CONDRY, J. 1977. Enemies of exploration: Self-initiated versus other-initiated learning. *Journal of Personality and Social Psychology,* 35(7), 459.

COOPER, S., 2010. The design of Alice. *ACM Transactions on Computing Education (TOCE)*, *10*(4), p.15.

CORDOVA, D. I. and LEPPER, M. R. 1996. Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *Journal of Educational Psychology,* 88(4), pp.715-730.

COUPER, M. P., TOURANGEAU, R., CONRAD, F. G. and SINGER, E. 2006. Evaluating the effectiveness of visual analogue scales: a web experiment. *Social Science Computer Review,* 24(2), pp.227-245.

CUMMINS, R. A. and GULLONE, E. 2000. Why we should not use 5-point Likert scales: The case for subjective quality of life measurement. In *Proceedings, second international conference on quality of life in cities,* pp.74-93.

CURRY, L. 1983. An Organization of Learning Styles Theory and Constructs. *Annual Meeting of the American Educational Research Association*, Montreal, Quebec.

CSIKSZENTMIHALYI, M. 1991. *Flow: The psychology of optimal experience* (Vol. 41). New York: Harper Perennial.

DALY, T. 2009. Using introductory programming tools to teach programming concepts: A literature review. *The Journal for Computing Teachers*.

DAVIES, S. P. 1993. Models and theories of programming strategy. *International Journal of Man-Machine Studies*, *39*(2), pp.237-267.

DECI, E. L., KOESTNER, R. and RYAN, R. M. 1999. A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psychological bulletin,* 125(6), p.627.

DETIENNE, F. 1990. Expert programming knowledge: a schema-based approach. *Psychology of programming,* pp.205-222.

DISCLOSURE SCOTLAND. 2007. *Disclosure Information.* [Online] Available at: https://www.disclosurescotland.co.uk/disclosureinformation/index.htm [Accessed 2010].

D'MELLO, S. 2013. A selective meta-analysis on the relative incidence of discrete affective states during learning with technology. *Journal of Educational Psychology*, pp 1082.

DREYFUS, H. L. and DREYFUS, S. E. 1986. *Mind Over Machine,* The Free Press.

Du BOULAY, B., 1986. Some difficulties of learning to program. *Journal of Educational Computing Research*, *2*(1), pp.57-73.

EDELSON, D. C. and JOSEPH, D. M. 2004. The interest-driven learning design framework: motivating learning through usefulness. In *Proceedings of the 6th International Conference on Learning Sciences* (pp. 166-173). International Society of the Learning Sciences.

EISENBERGER, R., PIERCE, W. D. and CAMERON, J. 1999. Effects of reward on intrinsic motivation – Negative, neutral, and positive: Comment on Deci, Koestner, and Ryan (1999).

ENTWISTLE, N. J. 1991. Approaches to learning and perceptions of the learning environment. *Higher Education* 22(3), pp. 201-204.

GERON, T. 2013. Bill Gates, Mark Zuckenberg, Chris Bosh Campaign For More Programmers. *Forbes [Online]*. Available: http://www.forbes.com/sites/tomiogeron/2013/02/26/bill-gates-celebrities-support-education-for-computer-programming/#3e080c07c618 [Accessed 03/02/2016]

FLEMING, N. and BAUME, D. 2006. Learning Styles Again: VARKing up the right tree! *Educational Developments,* 7(4), p.4.

FLEMING, N. D. and MILLS, C. 1992. Not another inventory, rather a catalyst for reflection. *To Improve the Academy, Vol.* 11.

GILMORE, D.J., 1990. Expert programming knowledge: a strategic approach. *Psychology of programming*, pp.223-234.

GLUCKSBERG, S., 1962. The influence of strength of drive on functional fixedness and perceptual recognition. *Journal of Experimental Psychology*, *63*(1), p.36.

GOOD, J. 2011. Learners at the wheel: Novice programming environments come of age. *International Journal of People-Oriented Programming (IJPOP),* 1(1), pp.1-24.

GOOD, J., HOWLAND, K. and NICHOLSON, K., 2010, September. Young people's descriptions of computational rules in role-playing games: an empirical study. In *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on* (pp. 67-74). IEEE.

GOOD, J., RIMMER, J., HARRIS, E. and BALAAM, M., 2011. Self-Reporting Emotional Experiences in Computing Lab Sessions: An Emotional Regulation Perspective. In *Proceedings of the 23rd Annual Psychology of Programming Interest Group Conference*.

GOOD, J. and ROBERTSON, J. 2006a. CARSS: A framework for learner-centred design with children. *International Journal of Artificial Intelligence in Education,* 16(4), pp.381-413.

GOOD, J. and ROBERTSON, J., 2006b. Learning and motivational affordances in narrative-based game authoring. In *The Proceedings of the 4th International Conference for Narrative and Interactive Learning Environments (NILE), Edinburgh* (pp. 37-51).

GREEN, T.R.G. and PETRE, M., 1996. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *Journal of Visual Languages & Computing*, *7*(2), pp.131-174.

GROSS, P. and KELLEHER, C., 2010. The Looking Glass IDE for learning computer programming through storytelling and history exploration: conference workshop. *Journal of Computing Sciences in Colleges*, *26*(1), pp.75-76.

HAMER, J., CUTTS, Q., JACKOVA, J., LUXTON-REILLY, A., MCCARTNEY, R., PURCHASE, H., RIEDESEL, C., SAELI, M., SANDERS, K. and SHEARD, J., 2008. Contributing Student Pedagogy. *ACM SIGCSE Bulletin*, *40*(4), pp.194-212.

HAMMERSLEY, M. and ATKINSON, P. 2007. *Ethnography: Principles in practice,* Routledge.

HIDI, S. 1990. Interest and its contribution as a mental resource for learning. *Review of Educational Research,* 60(4), pp.549-571.

HOELLWARTH, C. and MOELTER, M.J., 2011. The implications of a robust curriculum in introductory mechanics. *American Journal of Physics*, *79*(5), pp.540-545.

HONEY, P. & MUMFORD, A. (1982) Manual of Learning Styles. London: P Honey.

IPRE. 2010. *Institute for Personal Robots in Education* [Online]. Available: http://www.roboteducation.org/ [Accessed 2015].

JOHNSTONE, A.H. and PERCIVAL, F., 1976. Attention breaks in lectures. *Education in chemistry*, *13*(2), pp.49-50.

KAY, J.S., 2010, March. Robots as recruitment tools in computer science: The new frontier or simply bait and switch? In *AAAI Spring Symposium: Educational Robotics and Beyond*.

KARN, J. and COWLING, T., 2006, September. A follow up study of the effect of personality on the performance of software engineering teams. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering* (pp. 232-241). ACM.

KELLEHER, C. and PAUSCH, R., 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, *37*(2), pp.83-137.

KESSLER, C. M. and ANDERSON, J. R. 1986. Learning Flow of Control: Recursive and Iterative Procedures. *Human-Computer Interaction,* 2(2), pp.135 - 166.

KÖLLING, M. 1999a. The problem of teaching object-oriented programming Part I: Languages. Journal of Object-Oriented Programming, 11, 8-15.

KÖLLING, M. 1999b. The problem of teaching object-oriented programming Part II: Environments1. Journal of Object-Oriented Programming.

KÖLLING, M. 2008. Greenfoot: a highly graphical ide for learning object-oriented programming. *Proceedings of the ACM SIGCSE Bulletin*, *40*(3), pp.327-327.

KÖLLING, M., 2010. The Greenfoot programming environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), p.14.

KORCHNOY, E. and VERNER, I. M. 2010. Characteristics of learning computer-controlled mechanisms by teachers and students in a common laboratory environment. *International Journal of Technology and Design Education*, 20(2), pp. 217-237.

KUMAR, D. and MEEDEN, L., 1998, March. A robot laboratory for teaching artificial intelligence. In *ACM SIGCSE Bulletin* (Vol. 30, No. 1, pp. 341-344). ACM.

KURLAND, D. M., PEA, R. D., CLEMENT, C. and MAWBY, R. 1986. A Study of the Development of Programming Ability and Thinking Skills in High School Learners. *Journal of Educational Computing Research,* 2(4), pp.429-458.

LAVE, J. and WENGER, E. 1991. *Situated learning: Legitimate peripheral participation*, Cambridge university press.

LEGO. 2010. *LEGO Mindstorms* [Online]. Available: http://mindstorms.lego.com/en-us/default.aspx [Accessed 2010].

LINN, M.C. and DALBEY, J., 1989. Cognitive consequences of programming instruction. *Studying the novice programmer*, pp.57-81.

MCCONNELL, J.J., 1996. Active learning and its use in computer science. *ACM SIGCSE Bulletin*, *28*(SI), pp.52-54.

MCGRATH, C.H., GUERIN, B., HARTE, E., FREARSON, M. and MANVILLE, C., 2015. Learning gain in higher education. [Online]. Available: http://www.hefce.ac.uk/media/HEFCE,2014/Content/Pubs/Independentresearch/2015/Learning,gain,in,HE/Learning_gain.pdf [Accessed 23.10.16]

MCNERNEY, T.S., 2004. From turtles to Tangible Programming Bricks: explorations in physical language design. *Personal and Ubiquitous Computing,* 8(5), pp.326-337.

MAJOR, L., KYRIACOU, T. and BRERETON, O.P., 2012. Systematic literature review: Teaching novices programming using robots. *IET software*, *6*(6), pp.502-513.

MAKE IT DIGITAL [Online]. Available: http://www.bbc.co.uk/makeitdigital [Accessed 2015]

MAKER FAIRE [Online]. Available: http://makerfaire.com [Accessed 2014]

MALONE, T. W. and LEPPER, M. R. 1987. Making learning fun: A taxonomy of intrinsic motivations for learning. *Aptitude, learning, and instruction,* 3, pp.223-253.

MALONEY, J., RESNICK, M., RUSK, N., SILVERMAN, B. and EASTMOND, E. 2010. The Scratch Programming Language and Environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), p.16.

MARKHAM, S. 2000. Learning Styles measurement: a cause for concern. *Learning*, *p*. 9.

MARTÍ, J.A.T., BETORET, F.D., GARCÍA, M.O. and CIGES, A.S., 2006. Análisis de las variables mediadoras entre las concepciones educativas del profesor de secundaria y su conducta docente. *Revista de educación*, (340), pp.473-492.

MARTIN, C. and HUGHES, J., 2011. Robot dance: Edutainment or engaging learning. *Proceedings of the 23rd Psychology of Programming Interest Group PPIG 2011*.

MARTIN, C. and HUGHES, J. 2013. Learner-led assessment. *Proceedings of the International Conference on Enhancement and Innovation in Higher Education*. Glasgow.

MARTIN, R. C. 2008. *Clean Code: a Handbook of Agile Software Craftsmanship,* Pearson Education.

MEYER, D. K. and TURNER, J. C. 2002. Discovering emotion in classroom motivation research. *Educational psychologist*, 37(2), pp.107-114.

MIT Media Lab. *Life Long Kindergarten group* [Online]. Available: http://llk.media.mit.edu/ [Accessed 2010].

NESTA. 2014. *Nesta* [Online]. Available: http://www.nesta.org.uk [Accessed 2014].

OBLINGER, D. 2006. *Learning spaces*, EDUCAUSE Washington, DC.

OFCOM. 2015. *Ofcom* [Online]. Available: https://www.ofcom.org.uk/about-ofcom/latest/media/media-releases/2015/cmr-uk-2015 [Accessed 2016].

PANE, J.F. and MYERS, B. A. 1996. Usability Issues in the Design of Novice Programming Systems. *Human-Computer Interaction Institute Technical Report CMU-HCII-96-101*.

PAPERT, S. 1980. *Mindstorms: children, computers, and powerful ideas,* Basic Books, Inc.

PAPERT, S. and HAREL, I., 1991. Situating constructionism. *Constructionism*, *36*, pp.1-11.

PEKRUN, R., 1992. The impact of emotions on learning and achievement: Towards a theory of cognitive/motivational mediators. *Applied Psychology*, *41*(4), pp.359-376.

PEKRUN, R., GOETZ, T., TITZ, W. and PERRY, R.P., 2002. Academic Emotions in Students' Self-Regulated Learning and Achievement: A program of Qualitative and Quantitative Research. *Educational psychologist*, *37*(2), pp.91-105.

PERKINS, D.N., HANCOCK, C., HOBBS, R., MARTIN, F. and SIMMONS, R., 1986. Conditions of learning in novice programmers. *Journal of Educational Computing Research*, *2*(1), pp.37-55.

PERKINS, D.N. and MARTIN, F., 1986. Fragile knowledge and neglected strategies in novice programmers. In *first workshop on empirical studies of programmers on Empirical studies of programmers* (pp. 213-229).

PETRE, M. and PRICE, B., 2004. Using robotics to motivate 'back door' learning. *Education and Information Technologies*, *9*(2), pp.147-158.

PETTA, P., PELACHAUD, C. and COWIE, R., 2011. Emotion-Oriented Systems. *The Humaine Handbook, ISBN*, pp.978-3.

PIAGET, J., tr. COOK, M. 1952. The origins of intelligence in children. New York, NY: Internat. Univ. Press

POWERS, K., GROSS, P., COOPER, S., MCNALLY, M., GOLDMAN, K. J., PROULX, V. and CARLISLE, M. 2006. Tools for teaching introductory programming: what works? In *ACM SIGCSE Bulletin* (Vol. 38, No. 1, pp. 560-561). ACM.

PRINCE, M., 2004. Does active learning work? A review of the research. *JOURNAL OF ENGINEERING EDUCATION-WASHINGTON-*, *93*, pp.223-232.

PROCESSING. 2010 [Online] Available: https://processing.org/ [Accessed 2010].

PURCHASE, H. C. 2000. Learning about interface design through peer assessment. *Assessment & Evaluation in Higher Education*, 25(4), 341-352.

RCJ. 2010. *RoboCup Junior* [Online]. Available: http://www.robocup.org/robocup-junior/ [Accessed 2010].

RENNINGER, K.A. (2000). Individual interest and its implications for understanding intrinsic motivation. In C. Sansone and J. M. Harackiewicz (Eds.) *Intrinsic motivation: Controversies and new directions (pp. 373-404)*. San Diego, CA: Academic Press.

RESNICK, M. 2007. All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. *Proceedings of the 6th ACM SIGCHI conference on Creativity cognition*. (pp. 1-6). ACM.

RESNICK, M., MARTIN, F., SARGENT, R. and SILVERMAN, B., 1996. Programmable Bricks: Toys to think with. *IBM Systems journal*, *35*(3.4), pp.443-452.

RESNICK, M., MALONEY, J., MONROY-HERNÁNDEZ, A., RUSK, N., EASTMOND, E., BRENNAN, K., MILLNER, A., ROSENBAUM, E., SILVER, J., SILVERMAN, B. and KAFAI, Y., 2009. Scratch: programming for all. *Communications of the ACM*, *52*(11), pp.60-67.

RIST, R.S., 1995. Program structure and design. *Cognitive Science*, *19*(4), pp.507-561.

RIST, R.S., 1996. Teaching Eiffel as a first language. *Journal of Object-Oriented Programming*, *9*(1), pp.30-41.

ROBINS, A., ROUNTREE, J. and ROUNTREE, N., 2003. Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), pp.137-172.

ROBINSON, H., SEGAL, J. and SHARP, H., 2007. Ethnographically-informed empirical studies of software practice. *Information and Software Technology*, *49*(6), pp.540-551.

ROBOLAB-online. [Online] 2015. http://www.robolabonline.com [Accessed 2015].

ROBOTCUP JUNIOR 2015. [Online] http://rcj.robocup.org/ [Accessed 2015].

SCHIEFELE, U. 1991. Interest, learning, and motivation. *Educational psychologist, 26*(3-4), pp.299-323.

SHARP, H., ROBINSON, H. and PETRE, M. 2009. The role of physical artefacts in agile software development: Two complementary perspectives. *Interactive*. *Computing*, 21, 108-116.

SHARP, H., ROBINSON, H., SEGAL, J. and FURNISS, D. 2006. The Role of Story Cards and the Wall in XP teams: a distributed cognition perspective. In *Agile Conference, 2006* (pp. 11-pp). IEEE.

SIMPSON, J. A. and WEINER, E. S. 1989. *The Oxford English Dictionary*, Clarendon Press Oxford.

SKINNER, B. F. 1953. *Science and human behavior*, Simon and Schuster.

SMITH, M. K. 1997. *Participant observation. A guide for educators and social practitioners* [Online]. Available: http://infed.org/mobi/participant-observation-a-guide-for-educators-and-social-practitioners/ [Accessed 03/10/2013].

SOLOWAY, E., GUZDIAL, M. and HAY, K.E., 1994. Learner-centered design: The challenge for HCI in the 21st century. *Interactions*, *1*(2), pp.36-48.

SOLOWAY, E. and SPOHRER, J.C 1989. *Studying the Novice Programmer,* Hillsdale, NJ, USA, L. Erlbaum Associates Inc.

STERNBERG, R.J., 2003. The development of creativity as a decision-making process. *Creativity and development*, pp.91-138.

TANENBAUM, J.G., WILLIAMS, A.M., DESJARDINS, A. and TANENBAUM, K., 2013, April. Democratizing technology: pleasure, utility and expressiveness in DIY and maker practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2603-2612). ACM.

TOPPING, K., 1998. Peer assessment between learners in colleges and universities. *Review of educational Research*, *68*(3), pp.249-276.

TOPPING, K. and EHLY, S. eds., 1998. *Peer-assisted learning*. Routledge.

TUCKER, B., 2012. The flipped classroom. *Education Next*, *12*(1), pp.82-83.

VYGOTSKY, L.S., 1980. Mind in society: The development of higher psychological processes. Harvard university press.

WALL-E [online] 2016 http://www.disney.co.uk/wall-e/ [Accessed 2016].

WHITE, R.W., 1959. Motivation reconsidered: the concept of competence. *Psychological review*, *66*(5), p.297.

WIDOWSKI, D. and EYFERTH, K., 1986. *Comprehending and recalling computer programs of different structural and semantic complexity by experts and novices*. Techn. Univ.

WILSON, K. and KORN, J.H., 2007. Attention during lectures: Beyond ten minutes. *Teaching of Psychology*, *34*(2), pp.85-89.

WINSLOW, L.E., 1996. Programming pedagogy—a psychological overview. *ACM SIGCSE Bulletin*, *28*(3), pp.17-22.

Appendix I: Robot Dance Test

initial		male		female		[POST/PRE]

Please read the following statements and mark with a vertical line whether you think they are **true**, **false** or **not sure.**

	true	false	not sure
Programs only contain special instructions the computer understands.			
A variable is a place to store a value.			
Computer programs happen line by line.			
The instructions in a computer program happen in a random order.			
Computer programs start at top and finish at the bottom.			
A variable provides random numbers.			
The value of a variable can change when a program is running.			
Small changes in the text of a program will have no effect.			

Please read the following statements and mark with a vertical line how much you agree or disagree with them.

	strongly disagree	disagree	agree	strongly agree
Computer programming is **interesting**				
Computer programming is **science**				
Computer programming is **a challenge**				
Computer programming is **enjoyable**				
Computer programming is **fun**				
Computer programming is **art**				
Computer programming is **easy**				
Computer programming is **creative**				
Computer programming is **a useful skill**				

Please write any additional comments you would like to below:

date group id

Appendix II: Whack a Mole Pre-Test

initial		male		female	

[~~POST~~ / PRE]

	true	false	not sure	
An array is a group of items of the same type.				1
Arrays start at items 1 e.g. `array[1]`				2
An array is a contiguous collection of items of the same type.				3
New items cannot be added to an array when the program is running.				4

In your own words (and pictures) describe what an array is in the context of computer programming:	5

Given the following:

```
for(int i=0;i<10;i++){
  display(i);
}
```

What would you expect to be displayed? 6

Given the following:

```
int[] array = {0,6,9,3,2};
```

What is array[0] equal to: 7

Given the following:

```
int[] array2 = {3,4,5,2,3,5,6,7,9};

for(int i=0;i<8;i++){
  array2[i] = 0;
}
```

What would array2 contain now: ___ ___ ___ ___ ___ ___ ___ ___ ___ 8

Chris Martin
cjmartin@computing.dundee.ac.uk

Appendix III: Reflective Emotion Inventory

If you have experienced any of the following emotions throughout the session please circle them and give a brief description of when and why.					
Anger	0	1	2	3	
Annoyance	0	1	2	3	
Contempt	0	1	2	3	
Disgust	0	1	2	3	
Irritation	0	1	2	3	
Rage	0	1	2	3	
Anxiety	0	1	2	3	
Embarrassment	0	1	2	3	
Fear	0	1	2	3	
Worry	0	1	2	3	
Rage	0	1	2	3	
Doubt	0	1	2	3	
Envy	0	1	2	3	
Frustration	0	1	2	3	
Guilt	0	1	2	3	
Shame	0	1	2	3	
Boredom	0	1	2	3	
Despair	0	1	2	3	
Disappointment	0	1	2	3	
Hurt	0	1	2	3	
Sadness	0	1	2	3	
Amusement	0	1	2	3	
Delight	0	1	2	3	
Elation	0	1	2	3	

Chris Martin
cjmartin@computing.dundee.ac.uk

Please turn me over
there is more on the other side

Excitement	0	1	2	3	
Happiness	0	1	2	3	
Joy	0	1	2	3	
Pleasure	0	1	2	3	
Courage	0	1	2	3	
Hope	0	1	2	3	
Pride	0	1	2	3	
Satisfaction	0	1	2	3	
Trust	0	1	2	3	
Calm	0	1	2	3	
Content	0	1	2	3	
Relaxed	0	1	2	3	
Relieved	0	1	2	3	
Serene	0	1	2	3	
Stress	0	1	2	3	
Shock	0	1	2	3	
Tension	0	1	2	3	
Affection	0	1	2	3	
Empathy	0	1	2	3	
Friendliness	0	1	2	3	
Love	0	1	2	3	
Interest	0	1	2	3	
Politeness	0	1	2	3	
Surprised	0	1	2	3	

Chris Martin
cjmartin@computing.dundee.ac.uk

Appendix IV: Digital Makers Test

Name:		Age:		[post]

1		`int a = 0;`
		`a = a + 10;`
		`a-2;`

		The code above will make what happen?
	a	The variable a will equal 0.
	b	The variable a will equal 8.
	c	The variable a will equal 10.
	d	Not sure.

2		`int btn = digitalRead(buttonPin);`
		`if (btn == HIGH)`
		`  light on`
		`else`
		`  light off`

		The code above will make what happen?
	a	A light will come on when the button is pressed and stay on.
	b	A light will come on when the button is pressed and go off when it is released.
	c	A light will flash when a button is pressed.
	d	Not sure.
3		`int val = analogyRead(0);`
		What will val contain now?
	a	Nothing.
	b	The value of the analog pin 0.
	c	0
	d	Not sure.
4		What is the purpose of a user defined function?
	a	It contains a chunk of code you may wish to reuse several times.
	b	It makes the program function correctly.
	c	It is a built in feature of the language we need to use.
	d	Not sure.

Chris Martin
cjmartin@computing.dundee.ac.uk

Please turn me over
there is more on the other side

[post]

5		Suppose we want to branch or make a decision in out program we would use the following statement:
	a	`if`
	b	`when`
	c	`branch`
	d	`not sure`

6		`for(int i=0; i <10; i++){`
		`  light on`
		`  delay`
		`  light off`
		`  delay`
		`}`

		The code above will make what happen?
	a	A light will flash.
	b	A light will flash 10 times.
	c	A light will flash 9 times.
	d	Not sure.
7		Why is it a good idea to use user defined functions?
	a	It lets you write more code.
	b	It makes your main code easier to read.
	c	It will reduce the chance of wiring errors.
	d	Not sure.
8		`while(TRUE){}`
		The code above will make what happen?
	a	The code will loop infinitely.
	b	The program will stop running.
	c	The program will pause for a fixed period of time.
	d	Not sure

Chris Martin
cjmartin@computing.dundee.ac.uk

Appendix V: Whack a Mole Reflective Emotion Inventory

emotion	intensity 0 not at all 3 lots				During what activity and why did you feel this way? comment if you can.
Anger	0	1	2	3	
Annoyance	0	1	2	3	
Contempt	0	1	2	3	
Disgust	0	1	2	3	
Irritation	0	1	2	3	
Rage	0	1	2	3	
Anxiety	0	1	2	3	
Embarrassment	0	1	2	3	
Fear	0	1	2	3	
Worry	0	1	2	3	
Doubt	0	1	2	3	
Envy	0	1	2	3	

emotion	intensity 0 not at all 3 lots				During what activity and why did you feel this way? comment if you can.
Frustration	0	1	2	3	
Guilt	0	1	2	3	
Shame	0	1	2	3	
Boredom	0	1	2	3	
Despair	0	1	2	3	
Disappointment	0	1	2	3	
Hurt	0	1	2	3	
Sadness	0	1	2	3	
Amusement	0	1	2	3	
Delight	0	1	2	3	
Elation	0	1	2	3	
Excitement	0	1	2	3	

emotion	intensity 0 not at all 3 lots				During what activity and why did you feel this way? comment if you can.
Happiness	0	1	2	3	
Joy	0	1	2	3	
Pleasure	0	1	2	3	
Courage	0	1	2	3	
Hope	0	1	2	3	
Pride	0	1	2	3	
Satisfaction	0	1	2	3	
Trust	0	1	2	3	
Calm	0	1	2	3	
Content	0	1	2	3	
Relaxed	0	1	2	3	
Relieved	0	1	2	3	

emotion	intensity 0 not at all 3 lots				During what activity and why did you feel this way? comment if you can.
Serene	0	1	2	3	
Stress	0	1	2	3	
Shock	0	1	2	3	
Tension	0	1	2	3	
Affection	0	1	2	3	
Empathy	0	1	2	3	
Friendliness	0	1	2	3	
Love	0	1	2	3	
Interest	0	1	2	3	
Politeness	0	1	2	3	
Surprised	0	1	2	3	

Appendix VI: Code a Kilt Evaluation Sheet

Code a Kilt

aye MAKE

female ☐ male ☐ age >5 6-10 11-18 >18

Please read and consider the following statements. Mark with a vertical line on the accompanying scale the extent to which you agree or disagree with the statement .

I enjoyed designing and programming my own tartan.	agree	disagree
Being in a busy environment with lots going on helped me work.	agree	disagree
I feel I have learned some new things at this workshop	agree	disagree
I like that the computer program I was making produced something visual.	agree	disagree
I think my friends will be interested in seeing what I have made.	agree	disagree
I find it easier to do this kind of thing in the morning than the afternoon.	agree	disagree
I'm pleased I can take away the thing I have made.	agree	disagree
I would recommend this workshop to a friend.	agree	disagree
Seeing the tartan change as I edited my program helped me understand the program.	agree	disagree

If you have any further comments, ideas or recommendations, please feel free to note them down on the back of this. Thanks.

School of Computing
UNIVERSITY OF DUNDEE

Appendix VII: Wee Beasties Evaluation Sheet

Wee Beastie

aye MAKE

female ☐ male ☐ age >5 6-10 11-18 >18

Please read and consider the following statements. Mark with a vertical line on the accompanying scale the extent to which you agree or disagree with the statement.

I enjoyed creating and decorating the face of my Wee Beastie.	agree	disagree
Being in a busy environment with lots going on helped me work.	agree	disagree
I feel I have learned some new things at this workshop	agree	disagree
Putting all the electronic parts in the correct place was enjoyable.	agree	disagree
I think my friends will be interested in seeing what I have made.	agree	disagree
I find it easier to do this kind of thing in the morning than the afternoon.	agree	disagree
I'm pleased I can take away the thing I have made.	agree	disagree
I would recommend this workshop to a friend.	agree	disagree
Building my Wee Beastie helped me understand how electronics work.	agree	disagree

If you have any further comments, ideas or recommendations, please feel free to note them down on the back of this. Thanks.

School of Computing
UNIVERSITY OF DUNDEE

BARE CONDUCTIVE