

A Sense of Programming:

**An Exploration of Blind and Partially Sighted Learners’
Expressions of Their Sense of Programming Concepts**

Alexander Hadwen-Bennett

A thesis submitted in fulfilment of the requirements for the degree of Doctor of
Philosophy

King’s College London, 2021

Acknowledgements

This thesis would not have been possible without the support and input of many people. I owe a great debt of gratitude to my supervisors, Professor Lulu Healy, Dr Sue Sentance and Dr Cecily Morrison. I'm extremely grateful to Dr Sue Sentance for believing in me and giving me the opportunity to begin my career in academia and to Dr Cecily Morrison for giving me the amazing opportunity to work on the Code Jumper project with Microsoft. As my first supervisor, Lulu Healy has done an amazing job of guiding me through the PhD journey and her kind support has been instrumental in the creation of this thesis. I'm particularly grateful for the additional support she provided to ensure I stayed on track during the COVID-19 pandemic.

I would like to thank my husband, James, for his unwavering support and patience over the course of this four-year journey, which gave me the strength to keep going even through the more challenging times. To my brother and sister, Bryn and Eloise, thank you for always believing in me and supporting me.

Most of all I would like to thank my mum, because without her I would not have had the opportunity to achieve the things I have. She fought an education system that seemed determined to leave me behind and would not accept the judgement of people such as the head teacher at my primary school who believed that I would never be able to take GCSEs. I am so fortunate to have someone who saw my potential, fighting my corner and never giving up.

Abstract

Learners with visual impairments face particular challenges in learning to program, as many of the programming environments currently used in schools to introduce programming concepts are not easily accessible to them. Physical Programming languages have been suggested as a potential solution. The inherent tangibility of a physical language enables tactile explorations of code and its structure, which suggests they might impact on the ways in which learners appropriate the practice of programming. This thesis seeks to explore how forms of expressions of programming concepts are shaped, both by the programming environment, and by the embodied means through which students interact with it as they work on carefully crafted teaching activities. To interpret this process, Vygotsky's distinction between meaning and sense is employed, and teaching is envisaged as the support of students building their own sense, necessarily unique and context-dependent, of the socially shared meanings of the concepts in question.

Code Jumper was chosen as a suitable physical programming language for this research as it was specifically designed to be inclusive of learners with visual impairments. A pilot study was conducted to develop, evaluate, and refine teaching activities covering sequence, selection and repetition. The design of each activity drew upon Papert's constructionism, with each one conceptualised as a microworld which encompassed the tools employed and the pedagogical approach taken. The main study involved the delivery of a series of programming workshops to five learners with visual impairments. They had no previous programming experience and were aged between 11 and 15. The intervention was delivered over the course of one academic year and all sessions were video-recorded to capture the expressions of learners – including how they manipulated the tools, as well as their use of language and gestures.

The video recordings were analysed in an iterative process, drawing out the various ways the learners expressed their sense of programming concepts. All learners demonstrated the ability to successfully construct programs featuring sequence, selection, and repetition, however, they also expressed their sense of these concepts in other ways. Through gestures they expressed their understanding of sequence and repetition and demonstrated a strong link between the physical representation and

the development of their sense of these concepts. Additionally, they expressed their understanding of these concepts verbally, showing how they drew upon their experiences outside the computing domain in the development of their sense.

This study has demonstrated that physical representations can be a powerful tool for blind and partially sighted learners to learn how to program and that the representations they afford shape and become a part of the learner's developing sense of programming. As the students tackled the problems presented in the teaching intervention, the data suggests that learning involved the re-enactment and reprocessing of previous experiences, during which they could review their developing knowledge of programming concepts and modify their ideas. These results suggest that rather than conceiving these developing ideas as misconceptions that require correction, it would be more appropriate to refer to them as transitional theories that are gradually refined. Evidence of this learning process is manifested in a number of different ways, and not all of these forms of expression have been given sufficient recognition in past research where frequently much attention is directed to symbolic representations. This thesis has demonstrated the importance of valuing the embodied way in which programming can be accessed, engaged with and understood for the inclusion of blind and partially sighted learners. Additionally, it suggests that recognising the diversity of different embodiments of programming may be fundamental in the inclusion of other groups of learners that have traditionally experienced barriers to programming education.

Contents

Acknowledgements	2
Abstract	3
Chapter 1. Introduction.....	15
1.1 Background	15
1.2 Research Questions	16
1.3 Methodological Approach.....	17
1.4 Structure of the Thesis.....	17
Chapter 2. Theoretical Framing	19
2.1 Interpretations.....	19
2.2 Perezhivanie	20
2.3 Tools and Mediation.....	24
2.4 Forms of Speech.....	25
2.5 Sense and Meaning.....	27
2.6 Emotions.....	30
2.7 Summary	30
Chapter 3. Programming Education and Constructionism	32
3.1 The Contemporary Landscape.....	32
3.1.1 Identifying the Challenges and Strategies to Address Them	32
3.1.2 Pedagogical Approaches	38
3.1.3 Misconceptions.....	39
3.2 Back to Constructionism's Roots	42
3.2.1 Does Constructionism Advocate Undirected Discovery?.....	42
3.2.2 Abstract vs Concrete	43
3.2.3 Misconception or Transitional Theory?	44
3.2.4 Pedagogical Approaches from a Constructionist Perspective	46
3.3 Summary	48
Chapter 4. Accessibility of Programming Education for Blind and Partially Sighted Learners.....	50
4.1 Making Programming Accessible to Learners with Visual Impairments	50
4.1.1 Overview of Literature.....	50
4.1.2 Making Text-Based Languages Accessible	51
4.1.3 Making Block-Based Languages Accessible	55

4.1.4	Physical Artefacts.....	57
4.1.5	Auditory and Haptic Feedback.....	60
4.2	Discussion	61
4.3	Conclusion.....	62
Chapter 5.	Research Methods	63
5.1	Introduction	63
5.2	Epistemology.....	63
5.3	Methodology	67
5.3.1	Design-Based Research.....	67
5.3.2	Formative Experiments and Design Experiments.....	70
5.3.3	Unit of Analysis	74
5.3.4	Reflexivity.....	75
5.4	Data Collection.....	76
5.4.1	Expressing Sense.....	76
5.4.2	Video Recording	83
5.5	Ethical Considerations.....	84
5.6	Pilot Study	85
5.6.1	Learning Ecology.....	85
5.6.2	Data Analysis	91
5.7	Main Study	93
5.7.1	Learning Ecology	93
5.7.2	Data Analysis	98
5.8	Summary	102
Chapter 6.	Data Analysis: Sequence and Threading	104
6.1	Introduction	104
6.2	Steven’s Sense of Sequence and Threading	106
6.2.1	Exploration and Contour Following.....	107
6.2.2	Designing and Building Sequences.....	108
6.2.3	Sequence Assessment Activity.....	110
6.2.4	Threading	111
6.2.5	One Event, Multiple Actions.....	111
6.2.6	Summary	111
6.3	Adam’s Sense of Sequence and Threading	113
6.3.1	Exploration and Contour Following.....	113
6.3.2	Designing and Building Sequences.....	114

6.3.3	Sequence Assessment Activity.....	115
6.3.4	Threading	115
6.3.5	One Event, Multiple Actions.....	115
6.3.6	Summary	116
6.4	David’s Sense of Sequence and Threading	117
6.4.1	Exploration and Contour Following.....	117
6.4.2	Designing and Building Sequences.....	118
6.4.3	Sequence Assessment Activity.....	119
6.4.4	Threading	120
6.4.5	Summary	120
6.5	Sarah’s Sense of Sequence and Threading.....	121
6.5.1	Exploration and Contour Following.....	121
6.5.2	Designing and Building Sequences.....	123
6.5.3	Sequence Assessment Activity.....	123
6.5.4	Threading	124
6.5.5	One Event, Multiple Actions.....	124
6.5.6	Summary	124
6.6	Gregg’s Sense of Sequence and Threading	125
6.6.1	Exploration and Contour Following.....	125
6.6.2	Designing and Building Sequences.....	126
6.6.3	Sequence Assessment Activity.....	127
6.6.4	Threading	127
6.6.5	Summary	127
6.7	Discussion	128
6.7.1	A Sense of Sequence and Threading.....	128
6.7.2	Affect.....	130
6.7.3	One Event, Multiple Actions.....	130
6.7.4	Changes in Expression of Sense.....	131
Chapter 7.	Data Analysis: Repetition.....	132
7.1	Introduction	132
7.2	Steven’s Sense of Repetition	134
7.2.1	Exploration and Contour Following.....	136
7.2.2	Designing and Building Loops.....	137
7.2.3	Expressing a Sense of Repetition	138
7.2.4	Relationship Between Sense and the Physical Tool.....	138
7.2.5	Relationship to Non-Domain Specific Concepts	138

7.2.6	Repetition Assessment Activity	139
7.2.7	Transitional Theories.....	140
7.2.8	Summary	141
7.3	Adam’s Sense of Repetition	141
7.3.1	Exploration and Contour Following.....	143
7.3.2	Designing and Building Loops.....	143
7.3.3	Expressing a Sense of Repetition	144
7.3.4	Relationship to Non-Domain Specific Concepts	145
7.3.5	Repetition Assessment Activity	145
7.3.6	Transitional Theories.....	146
7.3.7	Summary	146
7.4	David’s Sense of Repetition	147
7.4.1	Exploration and Contour Following.....	149
7.4.2	Designing and Building Loops.....	149
7.4.3	Expressing a Sense of Repetition	149
7.4.4	Relationship to Non-Domain Specific Concepts	150
7.4.5	Transitional Theories.....	151
7.4.6	Summary	152
7.5	Sarah’s Sense of Repetition.....	152
7.5.1	Exploration and Contour Following.....	154
7.5.2	Designing and Building Loops.....	154
7.5.3	Relationship Between Sense and the Physical Tool.....	155
7.5.4	Expressing a Sense of Repetition	155
7.5.5	Relationship to Non-Domain Specific Concepts	155
7.5.6	Repetition Assessment Activity	156
7.5.7	Transitional Theories.....	156
7.5.8	Summary	156
7.6	Gregg’s Sense of Repetition.....	157
7.6.1	Exploration and Contour Following.....	159
7.6.2	Designing and Building Loops.....	159
7.6.3	Expressing a Sense of Repetition	159
7.6.4	Relationship to Non-Domain Specific Concepts	160
7.6.5	Transitional Theories.....	161
7.6.6	Summary	161
7.7	Discussion	162
7.7.1	Evolution in a Sense of Repetition	162
7.7.2	Relationship Between Sense and the Physical Representation	164

7.7.3	Drawing on Personal Experiences.....	164
7.7.4	Transitional Theories.....	165
Chapter 8.	Data Analysis: Selection and Variables	167
8.1	Introduction	167
8.2	Steven’s Sense of Selection and Variables.....	169
8.2.1	Exploration and Contour Following.....	170
8.2.2	Expressing a Sense of Selection.....	171
8.2.3	Expressing a Sense of Variables and Counters	171
8.2.4	Summary	172
8.3	Adam’s Sense of Selection and Variables.....	172
8.3.1	Exploration and Contour Following.....	173
8.3.2	Expressing a Sense of Selection.....	174
8.3.3	Expressing a Sense of Variables and Counters	174
8.3.4	Summary	175
8.4	David’s Sense of Selection and Variables.....	175
8.4.1	Exploration and Contour Following.....	176
8.4.2	Expressing a Sense of Selection.....	177
8.4.3	Expressing a Sense of Variables and Counters	177
8.4.4	Summary	178
8.5	Sarah’s Sense of Selection and Variables	178
8.5.1	Exploration and Contour Following.....	179
8.5.2	Expressing a Sense of Selection.....	180
8.5.3	Expressing and Sense of Variables and Counters	180
8.5.4	Summary	181
8.6	Gregg’s Sense of Selection and Variables.....	181
8.6.1	Exploration and Contour Following.....	182
8.6.2	Expressing a Sense of Selection.....	183
8.6.3	Expressing a Sense of Variables and Counters	183
8.6.4	Summary	184
8.7	Discussion	184
8.7.1	Expressing a Sense of Selection and Variables.....	184
8.7.2	Transitional Theories.....	185
8.7.3	Relationship Between Sense and the Physical Representation	186
Chapter 9.	Discussion: Developing a Sense of Programming	189
9.1	Introduction	189
9.2	The Role of Tools.....	189

9.2.1	Stimulus-Means.....	190
9.2.2	Stimulus-End.....	196
9.3	The Role of Transitional Theories.....	197
9.3.1	Revisiting Misconceptions	197
9.3.2	Relationship Between the Expert and the Novice.....	198
9.3.3	Role of Experiences Across Subject Domains.....	200
9.3.4	Representations of Programming.....	201
9.3.5	Transitional Theories.....	201
9.4	The Value of Perezhivanie	202
9.4.1	Introduction	202
9.4.2	A Personal Example.....	203
9.4.3	Reflecting on This Study.....	205
9.4.4	Implications for Pedagogy	206
9.4.5	Implications for Professional Development.....	207
9.4.6	Implications for Resource and Tool Development	208
Chapter 10.	Conclusion.....	209
10.1	Research Question 1	209
10.1.1	Sequence and Threading	209
10.1.2	Repetition	210
10.1.3	Selection and Variables.....	211
10.2	Research Question 2	212
10.2.1	Tools.....	213
10.2.2	Affect.....	214
10.2.3	Transitional Theories.....	214
10.3	Research Question 3	215
10.4	Contribution	216
10.5	Implications	218
10.5.1	Computing Pedagogy	218
10.5.2	Curriculum Designers	219
10.5.3	Tool Designers	220
10.6	Further Research.....	220
	References.....	221
Appendix 1	Pilot Study Ethical Approval	235
Appendix 2	Pilot Study Participant Information Sheet (Parent).....	237
Appendix 3	Pilot Study Participant Information Sheet (Child).....	240

Appendix 4	Pilot Study Consent Form (Parent)	243
Appendix 5	Pilot Study Consent Form (Child)	245
Appendix 6	Main Study Ethical Approval	247
Appendix 7	Main Study Participant Information Sheet (Parent)	248
Appendix 8	Main Study Participant Information Sheet (Child)	251
Appendix 9	Main Study Consent Form (Parent)	254
Appendix 10	Main Study Consent Form (Child)	256
Appendix 11	Main Study Curriculum	258
Appendix 12	Steven – Session 8 Timeline	267
Appendix 13	Steven – Coded Transcript for Session 8	273
Appendix 14	Steven – Narrative for Session 8	283
Appendix 15	Analysis of Steven’s Narrative	286

List of Figures

Figure 1: Vygotsky’s Speaking/Thinking System with Meaning at its Centre (Mahn, 2012)	22
Figure 2: Perezhivanie - Theoretical Model	23
Figure 3: Relationship Between Sense and Meaning	29
Figure 4: The Scratch Programming Environment	36
Figure 5: The Greenfoot Programming Environment	37
Figure 6: Example Programs to Illustrate Transitional Theory 8	46
Figure 7: Illustration of Epistemological Perspectives	64
Figure 8: Exploratory Procedures and Associated Movement Patterns (Lederman & Klatzky, 1987)	79
Figure 9: Illustration of the Types of Hand Movements that Come Under the Gesture Umbrella	83
Figure 10: Code Jumper in Use	87
Figure 11: Diagram of a Code Jumper Program	88
Figure 12: Design Board	95

Figure 13: Steven Contour Following in Activity 4	108
Figure 14: Steven Counting Pods in Activity 4	109
Figure 15: Sequence Assessment Activity Programs	111
Figure 16: Adam Contour Following in Activity 7	114
Figure 17: David counting the pods in activity 7	119
Figure 18: David exploring Program C in the Sequence Assessment Activity	119
Figure 19: Sarah producing a sequence gesture in activity 7	122
Figure 20: Gregg counting pods in activity 13	127
Figure 21: Code Jumper Loop Structure	136
Figure 22: Steven exploring a loop in reverse	137
Figure 23: Repetition Assessment Activity Programs	139
Figure 24: Adam's 'Loop' from Activity 6	144
Figure 25: Adam Making a Looping Gesture	145
Figure 26: David Making a Looping Gesture in Activity 21	150
Figure 27: Sarah Making a Looping Gesture in Activity 10	154
Figure 28: Gregg Making a Looping Gesture in Activity 20	160
Figure 29: Steven exploring the branches in Activity 27	171
Figure 30: Aspects of Sense	213

List of Tables

Table 1: List of Misconceptions - adapted from Sorva (2018) and Swidan et al. (2018) 40	
Table 2: Overview of Strategies for Making Text-Based Languages Accessible	55
Table 3: Overview of Strategies that Employ Physical Artefacts	60
Table 4: Theories Under the 'Constructivist' Umbrella	66

Table 5: Exploratory Procedures and Associated Properties (Adapted from Lederman & Klatzky, 1987)	79
Table 6: Updated List of Exploratory Procedures Incorporating Findings from Vinter et al. (2012)	82
Table 7: Pilot Study Participants	86
Table 8: Pilot Study Activities	90
Table 9: Pilot Study Groups	91
Table 10: Summary of Main Study Participants.....	94
Table 11: Main Study Activities	95
Table 12: Main Study Groups.....	97
Table 13: Steven’s Expressions of Sequence and Threading	107
Table 14: Adam’s Expressions of Sequence and Threading	113
Table 15: David’s Expressions of Sequence and Threading	117
Table 16: Sarah’s Expressions of Sequence and Threading	121
Table 17: Gregg’s Expressions of Sequence and Threading	125
Table 18: Steven’s Expressions of Repetition	135
Table 19: Adam’s Expressions of Repetition	142
Table 20: David’s Expressions of Repetition	148
Table 21: Sarah’s Expressions of Repetition	153
Table 22: Gregg’s Expressions of Repetition	158
Table 23: Steven’s Expressions of Selection and Variables	170
Table 24: Adam’s Expressions of Selection and Variables	173
Table 25: David’s Expressions of Selection and Variables	176
Table 26: Sarah’s Expressions of Selection and Variables	179
Table 27: Gregg’s Expressions of Selection and Variables	182
Table 28: Example Selection Programs 1	187
Table 29: Example Selection Programs 2	188

Chapter 1. **Introduction**

1.1 Background

The introduction of computing into the national curriculum for England in 2014 brought with it the requirement for primary school children to be taught the basic concepts of programming (Department for Education, 2014). Programming can be challenging to learn (Blackwell, 2002) and for blind and partially sighted learners there are numerous additional barriers to the learning process. Many modern programming environments are inaccessible to blind and partially sighted learners, being challenging or impossible to interface with using a screen reader (Baker, Milne, & Ladner, 2015; Stefik, Hundhausen, & Smith, 2011). Currently the most popular languages for introductory programming in primary schools are block-based (The Royal Society, 2017). Block-based languages such as Scratch (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010) enable learners to develop programs by snapping blocks together, removing the need for them to learn the complex syntax of a text-based language. However, block-based languages are intrinsically visual and are therefore not accessible to most blind and partially sighted learners. There have been efforts to make block-based languages more accessible in recent years (Koushik & Lewis, 2016; Lewis, 2014; Ludi, 2015), however the learners are only able to listen to one command at a time which means they are unable to gain a global overview of the program (Morrison et al., 2019). Physical representations are a potential solution to this challenge as they facilitate free exploration with the hands, enabling the learners to appreciate the relationships tangibly.

Physical or tangible programming languages have been in existence for a number of years (Horn & Jacob, 2007a), however they are largely inaccessible due to their reliance on vision to distinguish between the blocks. The Code Jumper physical programming language was designed to be inclusive of blind and partially sighted learners (Thieme, Morrison, Villar, Grayson, & Lindley, 2017). It employs pods that can be connected in order to produce sound in the form of music, stories and poems. The use of physical programming languages to teach programming to learners with visual impairments is a recent, but promising, development. Existing research regarding the use of Code Jumper with visually impaired learners has demonstrated that it is an effective alternative to block-based languages (Morrison et al., 2019). It

is clear that more research is needed in their area, specifically to examine the learning processes of blind and partially sighted students who work with physical programming languages. The findings of such research would inform programming pedagogy for teachers of the visually impaired. It is this need that has motivated the work in this thesis.

In framing my research I considered the argument put forward by Kravtsova (2017) suggesting that education which recognises and responds to the needs of individual learners must place an emphasis on sense over meaning. According to Kravtsova, most current education provision focuses on meaning over sense, which as a result is not truly student-centred. Student-centred learning can be particularly beneficial for learners that are most likely to face challenges in their education (Australian Institute for Teaching and School Leadership, 2013).

For Vygotsky (1987), the meaning of a concept is culturally defined and is relatively constant, whereas sense is unique for each individual. Vygotsky used the term *perezhivanie* to encompass both an experience and the working over of it. It is through the interaction of *perezhivaniya* that sense is formed (Blunden, 2016). In my research I sought to gain an understanding of the processes by which blind and partially sighted learners develop their sense of programming concepts.

1.2 Research Questions

This thesis seeks to address the dual concern of understanding the processes by which blind and partially sighted learners develop their sense of programming concepts, while building learning ecologies that would support engagement in these processes. It addresses the following three research questions:

1. How do blind, and partially sighted learners express their sense of sequence, threading, repetition, selection and variables?
2. What do these expressions reveal about the learning processes by which sense of programming develops?
3. How do the design structures embedded in the learning ecology support these learning processes?

1.3 Methodological Approach

In order to address my research questions, I required a methodological approach that would enable me to examine the processes by which visually impaired learners develop a sense of key programming concepts. Design-based research seemed to be suitable as it enables the development of models of learning processes through the collection of rich data in authentic settings (The Design-Based Research Collective, 2003). Jacob (1997) suggests that such an approach is more likely to produce models and interventions which transform into lasting change and improvement. I chose to employ a form of design-based research which has clear theoretical underpinnings, with the incorporation of Vygotsky's method of double stimulation (Engeström, 2011). This method enables the products of *perezhivaniya* to be highlighted at certain points in time to build a picture of the development of psychological functions (Vygotsky, 1997). It employs two stimuli: the first or 'stimulus-end' is the problem that the learner is asked to solve. The second stimuli or 'stimulus-means' is a tool that the learner could use to assist them in solving the problem. In the context of my investigation, the stimulus-end was the activity the learners were asked to complete, and the stimulus-means included the physical programming representations.

1.4 Structure of the Thesis

The thesis is organised into three main parts: literature review, empirical study and discussion. Starting in Chapter 2, the literature review outlines the theoretical framework which underpins this research. Chapter 3 examines the existing research relating to programming education, and this is followed by an exploration of the literature which focuses on making programming accessible to learners with visual impairments in Chapter 4.

The second part of the thesis covers the empirical study. The methods and methodology are outlined in Chapter 5 and the results are analysed in Chapters 6 to 8. Chapter 6 focuses on the analysis of the data relating to sequence and threading. The data pertaining to repetition is analysed in Chapter 7, and Chapter 8 deals with selection and variables.

The third and final part of the thesis starts with Chapter 9, which draws the data together in order to build a picture of how a sense of programming develops in blind

and partially sighted learners. The concluding chapter outlines how each of the three research questions have been addressed, followed by an exploration of the contributions and implications of this research.

Chapter 2. **Theoretical Framing**

This chapter will provide the theoretical background that underpins this research. At its centre is the concept of *perezhivanie* and the relationship between sense and meaning from a Vygotskian cultural-historical perspective. Additionally, in order to highlight the wider relationship between these concepts it, is necessary to examine other aspects of Vygotsky's theories.

The term *perezhivanie* refers to the intersection between personality and the environment. Each *perezhivanie* can be thought of as encompassing an experience and the internal working over of it (Blunden, 2016). Psychological functions are appropriated from *perezhivaniya* into the personality, and it could be argued that sense is one of those functions. Vygotsky viewed an individual's sense of a concept to be unique and able to vary in different contexts. Whereas, the meaning that is externally associated with a word is usually defined and relatively constant, that meaning only forms one part of the overall sense that each individual assigns to the word (Vygotsky, 1987).

Kravtsova (2017) argues that in order to provide truly student-centred education that recognises and responds to the needs of individual learners, we must place the emphasis on sense over meaning. This is particularly important as student-centred education has shown the potential to be particularly beneficial for students that have the greatest likelihood of facing challenges in their education (Australian Institute for Teaching and School Leadership, 2013).

Before exploring the relationship between *perezhivaniya*, sense and meaning further, it is important to consider the factors that can impact on the way in which Vygotsky's work can be interpreted. The following section will examine the grounds on which some scholars have challenged modern interpretations of Vygotsky.

2.1 Interpretations

Some contemporary scholars have challenged the common Western interpretations of Vygotsky's work. They suggest that these interpretations are often based on a small selection of his translated work, and that these translations may not accurately convey the true meanings of the originals or even, in some cases be incomplete

(Miller, 2011). Where possible, this synthesis will seek to include alternative interpretations in the discussion of the concepts relevant to this research.

From the perspective of Vygotsky, the human mind evolves out of cultural-historical processes. The individual cannot be separated from their environment; they are interlinked. The individual and their environment should not be considered as separate factors, they mutually mould each other as they develop (Daniels, Cole, & Wertsch, 2007). From this perspective it can be argued that to fully understand Vygotsky's work one must consider the socio-historical context in which it was developed (René van der Veer, 2007). Extending this further it could also be considered that the socio-historical context of each individual researcher will influence their own interpretation of Vygotsky, thus making a consensus on how his work should be interpreted unachievable. Scholars can however make reasoned arguments based on the sources available which are shaped by their own individual contexts. This is the approach that has been taken in producing this synthesis of Vygotsky's theories relating to the development and analysis of sense and meaning.

An example of the way in which western scholars may not accurately convey the concepts within Vygotsky's writing can be seen in the translation of the term *perezhivanie*, a key construct in this thesis. In recent years it has been argued that the common translations of the term do not capture its true meaning (Blunden, 2016). In the following section I will discuss the issue of translating *perezhivanie*, and its role as a key concept in Vygotsky's theoretical framework.

2.2 Perezhivanie

Vygotsky used the word *Perezhivanie* on a number of occasions and it has been translated in a variety of ways (Veresov & Flear, 2016), but is usually translated as 'experience' or 'lived experience'. However, this translation does not adequately convey the true meaning of the word and arguably there is no single English word that could take its place (Blunden, 2016). It is important to note that *perezhivanie* is a countable noun, not a mass noun like experience. The term experience, as we understand it in western culture, directly translated into Russian becomes 'opit', therefore, it seems that by using *perezhivanie* writers intend to signify a different concept (Blunden, 2016).

The original meaning of *perezhivanie* in Russian refers to life-changing episodes and their consolidation. It includes the processing of the experience in addition to the experience itself. This processing includes the assimilation into the personality. In simple terms a *perezhivanie* is an experience and the ‘working over of it’, this working over is referred to as catharsis. The plural of *perezhivanie* is *perezhivaniya* and together they form the essence of who we are as individuals (Blunden, 2016).

Each *perezhivanie* should be considered as a whole that is formed from the intersection of personality and the environment. It can be thought of like milk, it exists as a whole, but products can be abstracted from it like curds and whey (Blunden, 2016). Forms of analysis which attempt to decompose the “complex mental whole” (Veresov & Fleer, 2016, p. 330) into its constituent elements will lose the characteristics that are inherent to the whole. Therefore, a *perezhivanie* cannot be decomposed into separate elements, however we can analyse its products.

To illustrate this concept let us consider the scenario of a school trip in which a group of children take the bus to visit a church. All the children experience the same bus ride and church visit. However, the pictures that the children draw of the trip will differ as the *perezhivanie* through which the experience is refracted is different for every child. For example, one child may draw the church and one may draw the bus (Mackenzie & Veresov, 2013).

A *perezhivanie* can be considered as a unit, specifically a unit of personality and the environment (Veresov & Fleer, 2016), and more broadly the “unit of the development of the person as a whole” (Blunden, 2016). All aspects of an individual’s consciousness form from a unity of the *perezhivaniya* that have contributed to their development (Blunden, 2016; Veresov & Fleer, 2016). Veresov (2016) emphasises the power and significance of using *perezhivanie* as a unit of analysis in research as it allows us to examine the process of development, becoming a theoretical lens through which we can study the process of development.

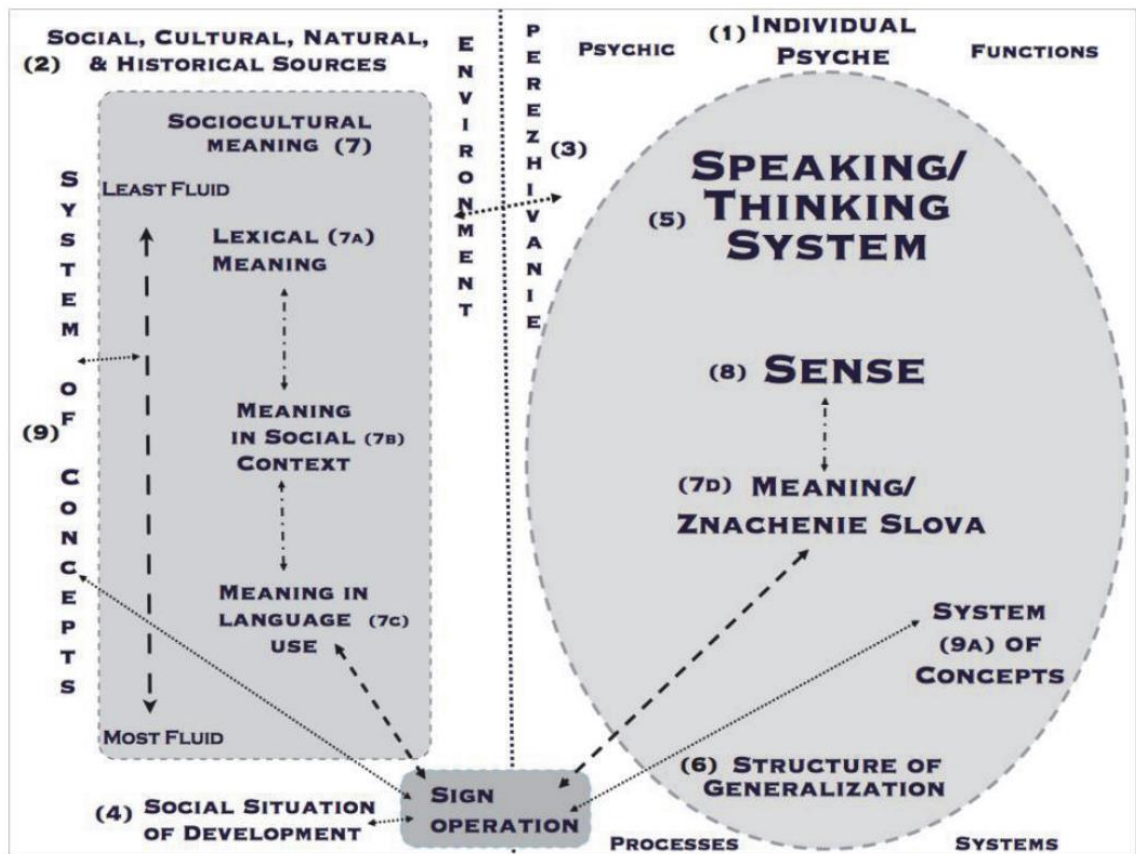


FIGURE 1: VYGOTSKY'S SPEAKING/THINKING SYSTEM WITH MEANING AT ITS CENTRE (MAHN, 2012)

In order to conceptualise how the concept of *perezhivanie* brings various aspects of Vygotsky's theoretical framework together, it would be beneficial to have a visual model to provide a representation of the relationships involved. Mahn (2012) put forward a model of Vygotsky's Speaking and Thinking System which incorporates *perezhivanie* (shown in Figure 1). It conceptualises the individual and their relationship to the environment. There is a dividing line between the environment and the individual psyche, with the latter given *perezhivanie* as a label. This seems to suggest that Mahn sees *perezhivaniya* as separate from the environment and additionally treats it as a mass noun like experience. These factors lead me to conclude that while Mahn's model does attempt to conceptualise how *perezhivaniya* fits into Vygotsky's wider framework, it does not address all the criticisms of the way in which the term has been translated and handled in western literature.

For this reason, I decided to propose my own model that takes these criticisms into account. As a starting point I drew on Blunden’s description of a *perezhivanie* as the intersection between the environment and personality; this language suggests a Venn diagram as a suitable representation. The proposed model can be seen in Figure 2. At the centre of the model are *perezhivaniya*, each of which forms at the intersection of the personality and the environment.

When discussing psychological functions that emerge from *perezhivaniya*, Blunden (2016) uses the term abstracted. In order to avoid confusion with the use of this term within the field of computing education I will instead use appropriated, a term used by Leontiev when describing the development of psychological functions (Mattosinho Bernardes, 2018). The model in Figure 2 demonstrates how psychological functions are appropriated from individual *perezhivaniya* which form part of and shape the personality.

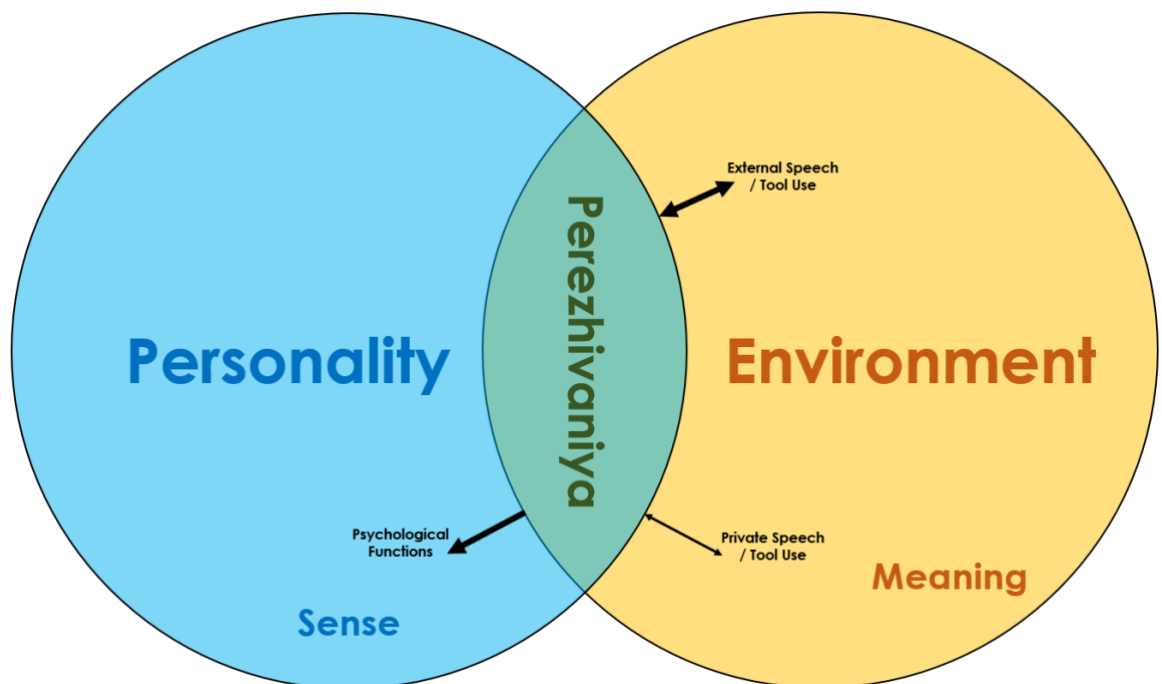


FIGURE 2: PEREZHVANIE - THEORETICAL MODEL

The model also demonstrates the role other aspects of Vygotsky’s work play in the development of the individual through their interaction with *perezhivaniya*. In the following sections I will examine these other aspects and discuss their role within my proposed model.

2.3 Tools and Mediation

Tools play a key role as they are seen to mediate the development of mental functions (Wertsch, 2007). Originally, when describing the development of mental functions, Vygotsky made the distinction between higher and lower mental functions. Lower mental functions are unmediated and could be described as genetically inherited, or our natural mental abilities; in contrast, higher mental functions are socially acquired and mediated by social meanings (Subbotsky, 1996). An example of a lower mental function is unmediated memory, as it stands without external aids or tools. On the other hand mediated memory, for instance when memory aids such as tying a knot in a handkerchief are used, is an example of a higher mental function (Hasan, 1992). It is important to acknowledge that Zavershneva's (2016) examination of the personal notes held within Vygotsky's personal archive indicates that he was moving away from the distinction between different levels of mental function in the latter stages of his career. For this reason, I decided not to make this distinction and to employ the broader term mental or psychological functions going forward.

Vygotsky described two different types of tool: material tools and psychological tools. Material tools enable humans to affect physical change in their environment. Whereas psychological tools, which were of particular interest to Vygotsky, mediate the development of mental functions. Psychological tools are often referred to as signs and material tools as simply tools (Miller, 2011). Types of sign include: "language, different forms of numeration and counting, mnemotechnic techniques, algebraic symbolism, works of art, writing, schemes, diagrams, maps, blueprints, all sorts of conventional signs, etc." (Vygotsky, 1997, p. 85).

It is common for modern scholars to group both signs and tools together under the heading of cultural tools or simply tools. Miller (2011) disagrees with this approach, arguing that it is a mistake to conflate the two, as Vygotsky made it explicit that the two types of tools serve very different functions. Miller does concede that some artefacts may have affordances that could place them under either category. I would go further, and suggest that the use of all tools has an impact on internal mental processes, and perhaps it would be more appropriate to propose that tools can have both physiological and material functions. From this perspective, it is not necessary

to distinguish between the different types of tool and we just need to select the relevant function for analysis purposes. Nonetheless, it is important to acknowledge that Vygotsky did pay a great deal of attention to the role that speech played in thought processes and the development of mental functions. Therefore, in the following section I explore Vygotsky's thoughts on the different forms of speech and the role they play within the wider context of tools.

2.4 Forms of Speech

When discussing the role that speech plays in the thought process, Vygotsky makes the distinction between inner and external speech. Inner speech is best described as speech for oneself and external is speech for others (Vygotsky, 1987). Inner speech is essential for the mediation and development of mental processes and according to Mahn (2002), language plays a key role in the process of making meaning from *perezhivaniya*.

Another form of speech which is outlined by Vygotsky is egocentric speech; it is self-directed like inner speech, however it is also externally perceivable, which inner speech is not (Vygotsky, 1987). The term egocentric speech was originated by Piaget. He associated it with a developmental stage he called egocentrism, in which children are not able to understand the point of view of others. However, Vygotsky criticised this view, stating that children start using socialised speech before developing internalised speech. Additionally, Vygotsky felt that Piaget underestimated the important function that egocentric speech plays (Sasso & Morais, 2014). To avoid confusion, egocentric speech, as Vygotsky defined it, is often referred to as private speech in academic literature (Berk, 1992). For this reason, going forward I will use the term private speech.

Vygotsky viewed private speech as an intermediary stage for children between external speech and private thought using inner speech. He suggests that as private speech distances itself from external, it becomes increasingly abbreviated, often resulting in it becoming meaningless for others (Berk, 1992). While on the one hand Vygotsky proposes that private speech transforms into inner speech as children develop, he also suggests that as task difficulty increases learners are more likely to return to the use of private speech (Fernyhough & Fradley, 2005). Alderson-Day and

Fernyhough (2015) take this idea further, suggesting that private speech continues to be employed as a tool in adulthood.

When considering speech, we do not have to limit ourselves to the spoken word. Rosborough (2014) suggests that from a Vygotskian perspective, gestures are an important part in the meaning-making process, and serve both inter and intra-personal functions. From the perspective of Lee (2008) gestures that perform a mainly intra-personal function play a key role in private speech and I would argue are a form of private speech themselves. In the same vein inter-personal gestures should also be considered as a form of external speech (Crowder, 1996). Additionally, notes for oneself could also be considered a form of private speech, for example Zavershneva (2016) suggests that many of the notes found in Vygotsky's personal archive are a form of private speech, as they are filled with abbreviations that would not make sense to others.

Given the covert nature of inner speech, it is not possible to directly observe its development. However, Vygotsky suggests that private speech is the observable counterpart to inner speech and as such offers an indirect route to the study of inner speech (Alderson-Day & Fernyhough, 2015). This would seem to suggest that private speech should be the focus of investigations into the development of internal thought processes. On the other hand, as the process of expressing external speech, in its various forms, shapes inner thought processes, I would argue that external speech should not be ignored when evaluating the development of internal thought processes.

In the creation of my model, I conceived inner speech as taking place within *perezhivaniya* as each *perezhivannie* incorporates the 'working over' of an event which would logically involve inner speech. However, I have not indicated it as a separate entity within the diagram as each *perezhivannie* exists as whole rather than as discrete elements. Private speech, the external counterpart of inner speech, is represented as two-way arrow indicating the relationship between inner speech within the *perezhivaniya* and the environment. External speech is represented in a similar manner, also using a two-way arrow, as Vygotsky makes it clear that the very process of externally expressing thoughts through external speech shapes the internal thought process (Vygotsky, 1987).

It is not only speech that shapes internal thought processes, but rejecting the distinction between physical and psychological tools implies that other forms of mediation contribute to the development of personality and sense in a similar manner. For this reason, I have extended the categorisation of different forms of speech to tool use. For example, external tool use would be described as interpersonal in the form of communication or collaboration. Inner tool use can be depicted as the covert, intra-personal employment of tools. Finally, private tool use is overt but also intra-personal and is closely linked with inner tool use.

According to Penuel & Wertsch (1995) inner speech places a greater emphasis on sense over meaning. In the following section I will explore how Vygotsky conceptualised these terms and the difference between them.

2.5 Sense and Meaning

Although Vygotsky makes numerous references to meaning throughout his work, the clearest distinction between sense and meaning appears in his last published work “Thinking and Speech” (Wertsch, 2000). In this he makes it clear that the external meaning of a word is relatively stable, whereas an individual’s sense is unique, fluid and is shaped by socio-cultural context. ‘The dictionary meaning of a word is no more than a stone in the edifice of sense’ (Vygotsky, 1987, p. 245). In systems which value the primacy of meaning, the focus is on external rules, whereas systems which place more emphasis on sense are focused on the individual (Kravtsova, 2017). In other words, in order to create a truly student-centred education system we must place a higher value on sense in relation to meaning.

There are many different interpretations of Vygotsky’s work, and Kravtsova (2017) argues that the primacy of sense over meaning is one of the central ideas behind Vygotsky’s psychology and that many modern interpretations of his work give primacy of meaning over sense. For example, it is suggested that Leontiev’s ‘psychology of activity’ prioritises meaning in relation to sense (Kozulin, 1995; Kravtsova, 2017).

In his writing, Vygotsky used the Russian word *znachenie* both as a general term that encompasses both sense and meaning, and as a term that stands in opposition to sense (*smysl*). This apparent contradiction could stem from the competing

philosophical traditions Vygotsky drew from in the development of his own theories (Wertsch, 2000). Chapter 7 of Vygotsky's 'Thought and Word' (1987) was written in the final months of his life and therefore, it could be argued, represents his most up-to-date thoughts on sense and meaning and perhaps he realised the necessity for clarifying this position.

In my model, meaning is included as part of the wider environment as it is culturally defined. Vygotsky described sense as "...the aggregate of all the psychological facts that arise in our consciousness as a result of the word" (Vygotsky, 1987, p. 275) and could therefore be thought of as an example of the psychological functions that are refracted from the *perezhivaniya*. For this reason, sense is placed within personality in the model, close to where psychological functions enter.

When discussing the difference between sense and meaning, Vygotsky talks about them in relation to the term 'word' which could also be described as 'sign'. I would argue that the same concept can have different or multiple signs associated with it by different individuals. For example, a learner may indicate a concept using a particular gesture that has become a sign that they associate with that concept. For this reason, I suggest that concept is a more appropriate term to use in relation to sense and meaning, as it allows for alternative signs to be associated with it.

A second model, shown in Figure 3, was developed in order to make the relationship between sense and meaning clearer. In this model sense, which is part of the personality, and meaning, which is part of the environment intersect to form the *perezhivaniya*. This illustrates how experiences which convey external meaning intersect with existing psychological functions in *perezhivaniya*. New psychological functions that are appropriated from each *perezhivanie* shape the individual's sense of a particular concept.

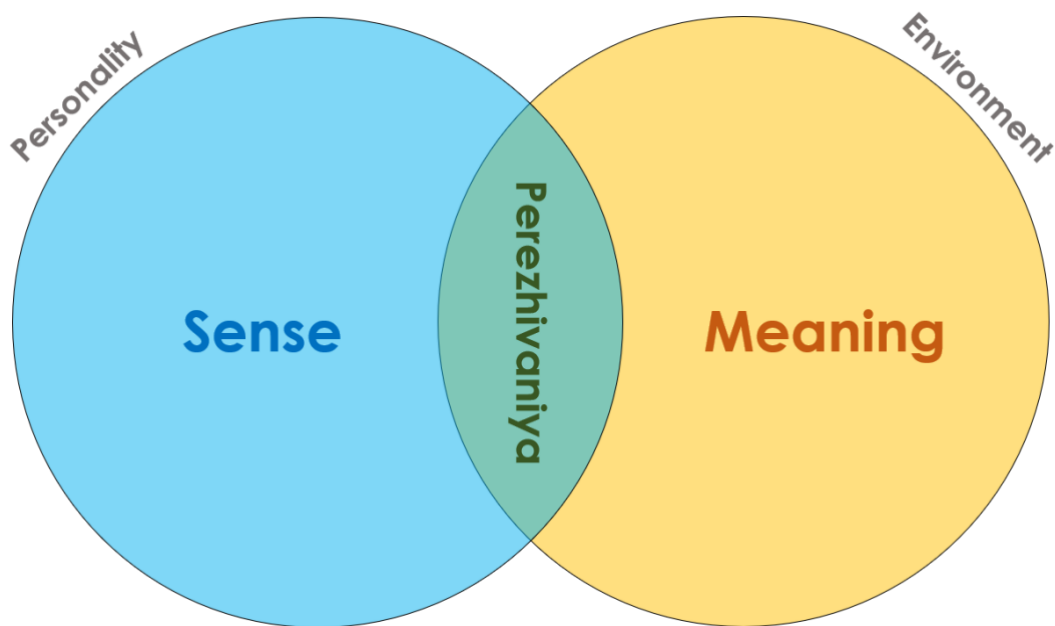


FIGURE 3: RELATIONSHIP BETWEEN SENSE AND MEANING

In order to investigate the development of sense of particular concepts, we need a window into internal mental processes. As previously mentioned, inner speech is key to the development of mental functions and private speech is an externally manifested partner of inner speech. Therefore, through examining private speech in its different forms, we can gain an insight into an individual's sense of different concepts, particularly as it is suggested that inner speech places a greater emphasis on sense (Penuel & Wertsch, 1995). However, as the process of expressing thoughts through external tool use shapes the thoughts themselves, external speech should not be excluded from the analysis as it may provide important insights. Additionally, it should be noted that it is not always possible to distinguish between private and external tool use for the purposes of analysis.

At the same time as writing 'Thinking and Speech' Vygotsky was also working on the lesser known 'The Teaching about Emotions: Historical-Psychological Studies' in which he highlighted the importance of emotion and affect in the development of thought (Mahn & John-Steiner, 2002). This aspect of Vygotsky's work will be explored in the following section.

2.6 Emotions

For Vygotsky the personality or consciousness consists of two key components: intellect and affect (Wertsch, 1985). Without taking both components into account, any analysis of human development is effectively incomplete:

The separation of the intellectual side of our consciousness from its affective, volitional side is one of the fundamental flaws of all of traditional psychology. Because of it, thinking is inevitably transformed into an autonomous flow of thoughts thinking themselves. It is separated from all the fullness of real life, from the living motives, interests, and attractions of the thinking human.

(Vygotsky cited in Wertsch, 1985, p. 189)

Vygotsky valued the important role that emotions and affect play in the development of thought. ‘Thought has its origins in the motivating sphere of consciousness, a sphere that includes our inclinations and needs, our interests and impulses, and our affect and emotions’ (Vygotsky, 1987, p. 282). Mahn and John-Steiner (2002) agree with this viewpoint, recognising the key part affect has to play in the learning process that is often neglected.

Emotions experienced during and as a result of a learning experience form part of the associated *perezhivanie* and as a result impact on the sense that is appropriated from it. For this reason, Mahn and John-Steiner (2002) suggest that emotions and affect should be taken into consideration when analysing *perezhivaniya*.

2.7 Summary

In this chapter I have explored the relationship between sense and meaning from a Vygotskian perspective. This exploration has revealed the need for a greater focus on sense within education and education research. In order to examine the development of sense I identified external and private tool use as potential windows into internal thought processes and sense.

Perezhivanie has been investigated as a potential unit of analysis for individual learning episodes, as it encapsulates both experiences and the processing of them, bringing together the personality and the environment. Using *perezhivanie* as a unit

of analysis enables the investigation of how sense develops over time, across multiple learning episodes.

Finally, this chapter has highlighted the importance of not trying to separate intellect and affect when researching development, as emotions form part of *perezhivaniya* and as a result, sense. As such, when examining external and private tool use, the role of emotions must be taken into consideration during analysis.

Chapter 3. **Programming Education and Constructionism**

This chapter will explore the contemporary landscape in programming education literature, highlighting the constructionist roots behind many modern approaches. This will be followed by examination of the differences between the way constructionism is currently represented in the field, and how it was conceived in the early programming education literature.

The research regarding programming education is very diverse, and in addition to investigating the teaching of programming skills, it also looks at the use of programming in the development of generalised problem solving skills (Scherer, Siddiq, & SánchezViveros, 2020). For the purposes of this review, I will focus on the development of programming skills, as this is the angle my research is taking.

3.1 The Contemporary Landscape

In this section I will examine the contemporary landscape of programming education research, exploring the perceived challenges and strategies that have been proposed to address them, misconceptions, and finally, pedagogical approaches.

3.1.1 Identifying the Challenges and Strategies to Address Them

Programming is considered challenging by many; however, it is not always clear why this is the case. Blackwell (2002) put forward three potential reasons learning to program may be considered challenging: loss of direct manipulation, use of notation, and abstraction as a tool to handle complexity. Although the piece is written from a cognitivist perspective, it does provide a useful starting point to consider the types of challenges that are faced when learning to program. Each of the three reasons will be described and examples of strategies which have been employed to address them given.

Many aspects of these challenges relate to abstraction. In computing, abstraction is viewed as an important skill that needs to be developed in order to allow learners to break problems down so that programmable solutions can be created. Conversely, the programming tools and concepts themselves are abstract to different degrees and this can make learning to program inaccessible to many learners. Therefore, many of

the approaches that have been employed to address these challenges have focused on making the tools and concepts more concrete.

3.1.1.1 Loss of Direct Manipulation

Programming often distances the programmer from the effect of their code in a number of ways. These can vary depending on the tools used and the context. This distance can be in time, as there is usually a delay in seeing the effect of the code. Another consideration is the nature of the objects or situations being programmed, which can often be abstract (Blackwell, 2002). For example, many learners find the abstract nature of purely mathematical programming problems proves to be a barrier for them (Veerasingam, D'Souza, & Laakso, 2016).

Many of the approaches for addressing these challenges have their roots in constructionism (Federici & Stern, 2011). Having close links to constructivism, Papert's constructionism suggests that learning is particularly fruitful when the learner is actively engaged in the construction of a public entity (Papert & Harel, 1991). It embraces the idea that we each create our own personal constructions of a concept through our experiences and therefore no two people's constructions can be identical.

One of the approaches that has origins rooted directly in constructionism is the use of microworlds. A microworld is a self-contained miniature world which can be thought of as a 'slice of reality' (Papert, 1987, p. 79). They are designed to enable learners to safely explore, to discover knowledge without having to worry about getting things wrong. Possibly the most well-known microworld is the Logo turtle microworld. It features an object that can be given commands in the Logo programming language to move around the world and draw lines. Sets of commands can be combined to, in theory, draw anything (Papert, 1987). Microworlds present learners with a less abstract and more relatable context in which to engage with the concepts they are learning. For this reason, many different microworlds have been employed in programming education over the years, enabling learners to use programming to explore a relatable and less abstract environment.

The original version of the Logo turtle microworld enabled learners to develop programs for a tangible floor turtle, however many microworlds are entirely virtual.

The concept of the microworld does not have to stop at a computer-controlled situation, but can be expanded to encompass the design of the learning environment as a whole (Edwards, 1991). For example, it may also include planned discussions and pen and paper activities.

Additionally, some microworlds tackle the challenge of loss of direct manipulation head on, by enabling learners to manipulate the position of an object by clicking on command buttons, which directly result in a corresponding movement in the object. The learners are then able to gradually build up to the creation of sets of commands which are not executed immediately (Gujberova & Kalas, 2013). A popular example of a microworld in modern programming education is the Scratch programming environment (Maloney et al., 2010). It builds upon the constructionist principles of Logo, enabling learners to “create interactive, media-rich projects” (Maloney et al., 2010, p. 1).

Another approach that seems to address the challenges of loss of direct manipulation is physical computing, which is often cited as having links to constructionism (Przybylla & Romeike, 2014b). In general, physical computing activities involve the design, creation and programming of tangible, real world products such as interactive objects or installations (Przybylla & Romeike, 2014a).

In the literature there has been some criticism of some aspects of constructionist-based approaches. For example, Meerbaum-Salant et al. (2011) suggest that Scratch engenders a bottom-up approach to programming, tying in with the bricolage approach associated with Turkle and Papert and constructionism. In this context the bottom-up approach is considered as bad programming practice, and it is implied that the constructionist approach encourages its use. Additionally, Kolikant & Mussai (2008) use the term bricolage, with negative connotations, when discussing the tendency for novice programmers to debug a program from a local rather than holistic angle.

Grover and Basu (2017) suggest that the exploratory nature of constructionist approaches to programming education can lead to the development of misconceptions in loops, variables, and Boolean logic. They propose that it is a flaw of constructionist approaches that do not explicitly teach or address these concepts.

They call for a balanced approach with other pedagogies, which they refer to as guided discovery. This seems to imply that the constructionist approach favours unguided discovery. However, there is no consensus in the field on what discovery learning is (Alfieri, Brooks, Aldrich, & Tenenbaum, 2011). The following extract describes discovery learning in general terms.

Allowing learners to interact with materials, manipulate variables, explore phenomena, and attempt to apply principles affords them with opportunities to notice patterns, discover underlying causalities... (Alfieri et al., 2011, p. 1).

3.1.1.2 Use of Notation

The nature of programming results in the need for some form of representational notation, and therefore results in some degree of abstraction (Blackwell, 2002). The properties of the representational notation can impact on the quality of interaction. Different forms of programming notation can have different affordances that can make it more suited to some tasks than others (Blackwell, 2002). According to Qian and Lehman (2017) it is common for novice programmers to find syntax challenging. They also suggest that there is a correlation between insufficient syntactic knowledge and levels of conceptual knowledge.

Traditionally programming has employed text-based commands, however there are other forms of representation such as block-based programming languages, that use graphical blocks which can be snapped together to represent commands. Probably the most well-known block-based language is used in the Scratch programming environment, that was mentioned earlier, and is employed widely with young learners (Meerbaum-Salant et al., 2011). The use of blocks makes the syntax of the language transparent, with the shapes designed in such a way that incompatible commands cannot be put together. Additionally, the commands do not have to be typed and therefore removes the chance of typographical errors (Maloney et al., 2010). An example of the Scratch programming environment can be seen in Figure 4.

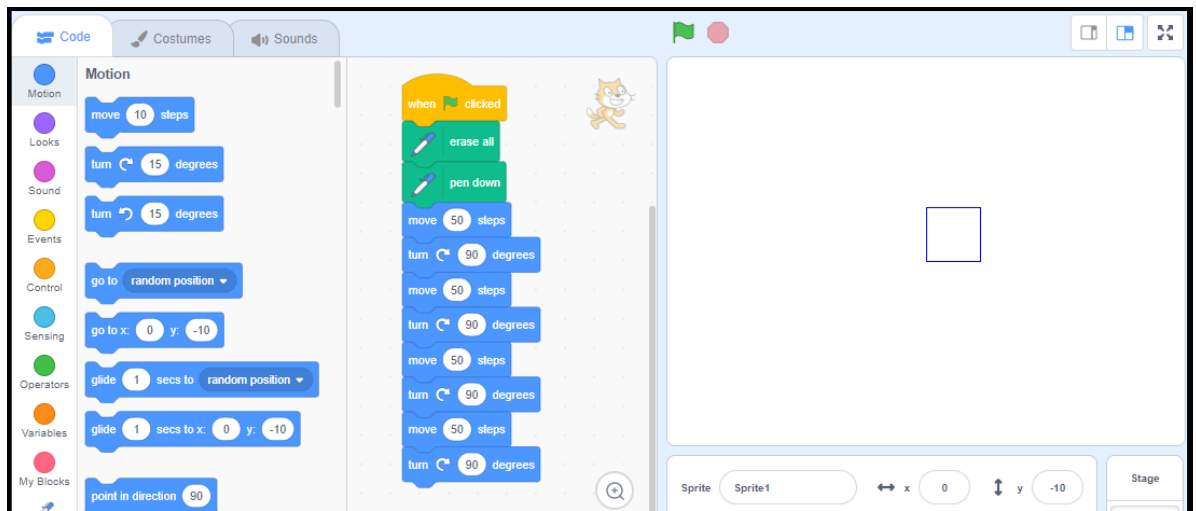


FIGURE 4: THE SCRATCH PROGRAMMING ENVIRONMENT

An alternative to the use of visual blocks, is commands that are represented in the form of tangible objects to make tangible or physical programming languages (Papavlasopoulou, Giannakos, & Jaccheri, 2017). One example of a physical programming language is Tern. Tern uses wooden blocks which slot together to represent commands (Horn & Jacob, 2007b). The design of the language limits the possibility of syntax errors in a similar fashion as block-based languages. Another example of a physical language is Code Jumper, which enables learners to develop programs that produce sound in the form of music, stories and poetry. Code Jumper was designed to be inclusive of learners with visual impairments (Morrison et al., 2019). It could be argued that the tangible nature of these forms of notation reduces the level of abstraction for the learner.

Another form of notation is Pseudocode which is often used in programming education. It is seen as a ‘blend of formal and natural languages’ and is often used in programming assessments. In national exams it is regularly seen as necessary for questions to be set in a non-specific programming language, as different schools use different languages. For this reason, each exam board sets their own form of pseudocode. It can be problematic as it produces ambiguity, as there are multiple forms of pseudocode. Cutts et al. (2014) recommend the use of the term ‘reference language’ rather than pseudocode for a formally-defined language that is used for all national assessments.

3.1.1.3 Abstraction as a Tool to Handle Complexity

As programs become increasingly complex, we need to find strategies to keep the code manageable. This is often handled through the creation of functions or modules to abstract the individual steps required to perform a common operation (Blackwell, 2002). In my experience, functions are usually taught as one of the later topics in programming courses, as they are considered to be more challenging. However, this perception does not seem to have been justified in the literature.

Microworlds have also been employed as a technique to address the challenge of using abstraction as a tool to handle complexity. For example, the Greenfoot environment provides a virtual world, in which learners can define the properties of different actors through the creation of classes (Kölling, 2010). When new instances of an actor are created, they will inherit all the properties defined in the class. The Greenfoot microworld can be seen in Figure 5.

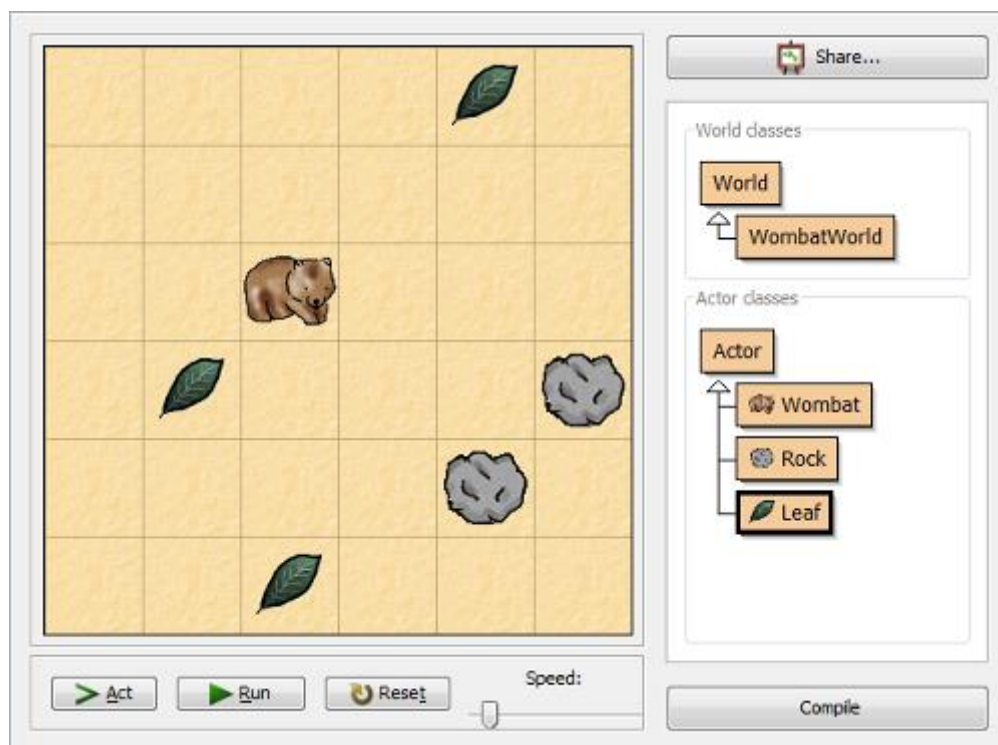


FIGURE 5: THE GREENFOOT PROGRAMMING ENVIRONMENT

3.1.2 Pedagogical Approaches

One area that has been explored in some depth is program comprehension; this refers to the techniques programmers use to develop an understanding of a given program. Much of this research has focused on programmers in professional settings, rather than on novices in education (Schulte, Clear, Taherkhani, Busjahn, & Paterson, 2010). Exton (2002) suggests that taking a constructivist perspective to learning would imply that there is no one model for the way in which a programmer comprehends a program. The strategies employed are driven by the programmer's existing knowledge (Rajich, 2002) and therefore the stages involved for each individual will vary and the tools should be designed to cater for this (Exton, 2002). This approach to program comprehension ties in with Vygotsky's views on learning and development.

When learning to program, novices are often expected to write code before they learn to read it (Lister et al., 2004). Given the importance that seems to be placed on program comprehension, this approach does not seem to make sense. There are teaching approaches that have been proposed to address this issue. For example, the use-modify-create approach places emphasis on starting with the use of an existing program rather than starting with a blank screen (I. Lee et al., 2011). This enables learners to read and comprehend existing code, before moving onto modifying it and finally creating their own original programs. The use-modify-create approach was developed further in the creation of PRIMM (Predict, Run, Investigate, Modify, Make) (Sentance, Waite, & Kallia, 2019), giving more structure to Use part of Use-Modify-Create.

Another consideration is the role of collaboration in programming pedagogy. Approaches which foster collaboration in programming education have been shown to have a positive impact on learning outcomes (Scherer et al., 2020). One such approach is pair programming, developed as part of the Extreme Programming (XP) software development methodology. Pair programming is a practice in which two programmers develop a program by working together on the same computer (Beck, 1999). A number of studies have investigated the effectiveness of pair programming as a pedagogical tool, and a broad range of research has indicated that pair programming has an overall positive impact on the development of programming

skills in learners (Bevan, Werner, & McDowell, 2002; Charlie McDowell, Brian Hanks, & Linda Werner, 2003; Salleh, Mendes, & Grundy, 2011).

3.1.3 Misconceptions

Misconceptions is a popular area of research within the field of programming education (Qian & Lehman, 2017). In programming education, a misconception can be viewed as ‘an incorrect understanding of a concept or a set of concepts, which leads to making mistakes in writing or reading programs’ (Swidan, Hermans, & Smit, 2018, p. 1). It does not imply a complete absence of knowledge, but it suggests incomplete, self-interpreted knowledge that may originate from other subject domains.

Du Boulay (1986) argued that the learner develops an internal model of how the machine they are programming works, which is shaped by their experiences of interacting with it; he called this the notional machine. This notional machine can also be influenced by the use of analogies, when educators explain how the machine and its language functions, and additionally through the alternative meanings of keywords also found in natural language. Being based on experience, the notional machine of each individual will be unique and could, perhaps, be seen as an individual’s sense of how the machine functions, which is shaped by multiple and varied *perezhivaniya*. It is suggested that misconceptions often have their roots in the individual’s notional machine. This, by implication, means that misconceptions will vary considerably between learners, however there are themes that have emerged in the literature.

Sorva (2012) identified 162 programming misconceptions within computing education literature. He later narrowed this list down to 41 misconceptions, which included two additional misconceptions (Sorva, 2018). Swidan et al. (2018) also produced a narrowed down version of Sorva’s original list, containing 11 misconceptions. They narrowed down the list by selecting the most commonly occurring misconceptions in the literature. They then refined this list further by removing misconceptions that are not applicable to tools used in early programming education such as Scratch. I discounted 23 of the 41 misconceptions identified by Sorva (2018) and one those identified by Swidan et al. as they related to concepts such as objects and subroutines which are not present in the chosen programming

environment. Five of the remaining misconceptions appeared in both lists, and removing these duplicates left 23 misconceptions in total. A summary of the remaining misconceptions can be seen in Table 1. The number originally assigned to each misconception by Sorva is indicated in the table.

TABLE 1: LIST OF MISCONCEPTIONS - ADAPTED FROM SORVA (2018) AND SWIDAN ET AL. (2018)

	Sorva 2012	Sorva 2018	Swidan et al. 2018	Description
1	Y (9)	Y	Y	A variable can store multiple values; it may store the 'history' of values assigned to it
2	Y (14)	Y	Y	A variable is merely a pairing of a name with a value. It is not stored in the computer
3	Y (15)	Y	Y	An assignment statement stores the equation rather than the result
4	Y (13)	Y	N	A program, particularly those with assignment statements, are essentially groups of equations
5	Y (8)	Y	N	Several lines of a non-concurrent program can be active simultaneously
6	Y (12)	Y	N	Assignment statements work in both directions
7	N	Y (8)	N	A variable name needs to be a single letter
8	Y (33)	Y	Y	Loops terminate as soon as condition changes to false
9	Y (25)	Y	N	An if statement triggers whenever its condition becomes true
10	Y (16)	Y	N	Assignment statements move values rather than copy them
11	N	Y (12)	N	Variables are initially empty containers and do not need to be initialised
12	Y (20)	Y	N	Incrementing a variable is a single operation and there is no concept of evaluation and assignment
13	Y (2)	Y	N	The computer is able to deduce the intention of the programmer
14	Y (4)	Y	N	The computer will not allow operations that are unreasonable or pointless

	Sorva 2012	Sorva 2018	Swidan et al. 2018	Description
15	Y (17)	Y	Y	The name of a variable affects the value that is assigned to it
16	Y (27)	Y	N	Both then and else branches are always executed
17	Y (29)	Y	N	The code following an if statement is the same as an else branch
18	Y (161)	Y	N	Booleans are perceived only as part of control structures and not as values
19	Y (30)	N	Y	Adjacent code executes within the loop
20	Y (31)	N	Y	Control goes back to start of the program when condition is false
21	Y (11)	N	Y	Assignment works in the opposite direction
22	Y (23)	N	Y	Difficulties in understanding the order in which statements are executed
23	Y (26)	N	Y	A false condition ends program if no else branch exists

Kolikant and Mussai (2008) took another angle when investigating misconceptions, by looking at identifying learner's conceptions of program correctness. They found that novices tend to focus on debugging a program from a local angle, rather than holistic, locating individual errors rather than focusing on developing an understanding of the overall structure of the program as a whole. They suggest that a program with any type of error should be considered incorrect, as that would be the view of professional programmers. They attribute this misconception to the standard assessment techniques in education, that award marks for partially complete programs.

Although not technically misconceptions, there are suggestions in the literature that learners are not applying programming constructs as they should be. For example, in their investigation of student Scratch projects, Meerbaum-Salant et al. (2011) point out that control structures were not being employed as a professional computer scientist would expect. Meerbaum-Salant et al. also identified that young novice programmers found the concept of concurrency or threading to be challenging.

Franklin et al (2017) also found that novice programmers found it challenging to create sequences which consist of events with multiple actions. For example, an event with one action could be moving forward a set amount, whereas an event with two actions could both rotate and move in the same event.

3.2 Back to Constructionism's Roots

As Tenenberg and Knobelsdorf (2014) point out, much of the contemporary computing education literature is built upon a cognitivist theory of mind. From the cognitivist perspective, knowledge within the mind is free from context, making this view incompatible with the cultural-historical theories of Vygotsky. However, their literature search only focused on modern sources, and seemingly excluded early programming education literature from the 1980s and 90s, when constructionism was a major player in programming education research. Although constructionism is still around today, it seems to be mainly applied to the development of tools and not the methodology applied to research, which tends to take a cognitivist lens. This section will re-examine constructionism and explore whether the modern criticisms are based on misrepresentations of the constructionist philosophy.

3.2.1 Does Constructionism Advocate Undirected Discovery?

As discussed earlier, some contemporary literature characterises constructionism as advocating totally undirected discovery-based learning. It seems that this view may partly be down to the way in which Scratch is depicted by the team at MIT, whose focus is largely on learning in informal settings. For example, they use phrases such as “self-directed learning, and emphasizes tinkerability” (Maloney et al., 2010, p. 14) which could be seen to be promoting undirected discovery. The fact that Scratch is explicitly portrayed as building on “... the constructionist ideas of Logo” (Maloney et al., 2010, p. 3) could lead some to believe that this approach characterises constructionism as a whole.

Research has indicated that totally unaided discovery learning is not as effective as direct instruction (Kirschner, Sweller, & Clark, 2010), however directed or guided discovery can often have significant benefits over direct instruction (Alfieri et al., 2011; McLaughlin, 1981). Although this does tie in with Grover and Basu's assertion that unguided discovery is ineffective in programming education, it is

uncertain whether constructionism advocates this approach. In *Mindstorms* Papert does not mention discovery learning, however he does describe microworlds as ‘discovery rich’ (Papert, 1980). Later Papert stated that constructionism does not specifically question the value of direct instruction, instead it is a reminder that it needs to be kept in check (Papert, 1992). McLaughlin (1981) suggests that consideration of level of learner control is a key aspect of constructionist approaches; at the start of the learning process students may need more direction than they require later. Therefore, the instructional approach must be flexible, so it can be adapted to the changing needs of the learner. Considering these points, I would suggest that Grover and Basu’s argument is based on a misrepresentation of constructionist approaches. From the constructionist perspective, discovery is not learning by doing without any instruction or direction, it is guided or facilitated to a greater or lesser extent, depending on the needs of the learners (McLaughlin, 1981).

The ‘play paradox’ proposed by Noss and Hoyles (1996) also backs up this interpretation of constructionism. The play paradox suggests that direct instruction where learners are simply told what they need to know, is not an effective method of facilitating the development of a deep understanding of a concept, whereas learning through play and exploration can engender a deeper understanding. However, with totally undirected exploration, there is a chance that the learner will go off in a totally different direction and not develop any understanding of the target concept at all. In reality there is no perfect solution to this paradox, the best we can do is strive to maintain a balance between play and direction.

3.2.2 *Abstract vs Concrete*

Many of the more concrete tools which have been developed to address the challenges of learning to program, are often viewed as stepping stones to more traditional and abstract forms of programming. Papert was careful to clarify that constructionism’s focus on the benefits of the concrete does not imply that it should be used as a “stepping-stone to the abstract”; it concerned him that this approach “...would leave the abstract ensconced as the ultimate form of knowing.” (Papert, 1992). He criticised the valuing of the abstract over the concrete, suggesting that in many cases, concrete could be considered more valuable, with abstract concepts serving as tools to enhance concrete thought. From his perspective, the emphasis on

abstract and formal knowledge actively discriminates against many children, as it impedes their opportunity to learn. For Papert the goal is to recognise and facilitate different preferred approaches to learning. For example, some learners prefer approaches to thinking that enable them to stay close to the physical and concrete, whereas others prefer abstract means which distance them from the concrete (Papert & Harel, 1991).

Similarly, Vygotsky also questioned the view of abstract to concrete being a linear, one-way process. He suggested that it is a bi-directional process, which enables us to connect our abstract knowledge to the concrete and vice-versa (Vygotsky, 1987). Vygotsky also recognised that learning or memorising abstract concepts in the absence of connections to the concrete, does not afford the development of deep understanding (Swanson & Williams, 2014).

3.2.3 Misconception or Transitional Theory?

In their evaluation of the state of misconception literature relating to mathematics and science education, Smith et al. (1993) highlighted that the framing of misconceptions, at the time, was not compatible with a constructivist approach to learning, as it emphasized the flaws in student conceptions rather than focusing on the gradual refinement of existing conceptions. From my perspective, the framing of misconceptions in programming education has followed a similar pattern, in which misconceptions are conceived as deficits in the learner and must be replaced. However, there are signs that the emphasis of misconception research in computing education is beginning to shift, as misconceptions are starting to be viewed as learning opportunities in some circles (Margulieux, Denny, Cunningham, Deutsch, & Shapiro, 2021).

From a Vygotskian perspective an individual's sense of a concept is unique and constantly developing. Rather than focusing on the idea of misconceptions, it would perhaps be better to think of a learner's sense of a concept as under development, and to look at methods of assessment which recognise the current shape of their conception, and how they can be supported in developing it further.

Papert believed that the development of misconceptions, or as he referred to them 'transitional theories', are an essential aspect of bridging the gap between formal

subject domain knowledge and personal knowledge and experience. He argued that the theories and systems that learners develop in this process may not be recognised by a specialist in that area as being a valid part of their field (Papert, 1980).

The unorthodox theories of young children are not deficiencies or cognitive gaps, they serve as ways of flexing cognitive muscles, of developing and working through the necessary skills needed for more orthodox theorizing.

So, rather than stifling the children's creativity, the solution is to create an intellectual environment less dominated than the school's by the criteria of true and false (Papert, 1980, p. 133).

This perspective can also be applied to the way in which we judge the efficiency of code. I would argue that while there are industry practices in programming to ensure the efficiency and maintainability of the code, utilising another method that achieves the same goal is not, strictly speaking, wrong. I suggest that early on in programming education, it is more important to enable the learners to develop their own sense of programming constructs in a manner that works for them. Later on, when they feel more secure in their understanding, we can discuss professional programming practices and why they are used. Papert himself questioned the imposition of particular styles of programming with the implication that they were “the right way” (Papert, 1992).

If we take transitional theory 8 from Table 1 as an illustration, when working with conditional controlled loops, learners may believe that as soon as the variable that is used to control the condition changes to meet the stopping condition, the loop will exit rather than continuing until the next time the condition is checked. Looking at Program A in Figure 6 the statement that updates the variable which is used in the condition appears at the end of the loop. It is quite reasonable for a learner to conclude that a loop will end as soon as the controlling variable changes to meet the condition, particularly if they have only encountered loops in which the variable update occurs in the last statement. In these cases, any hypothesis based on this theory regarding the last value to be outputted is likely to be correct. If they were

then to apply their theory to Program B, they may not guess the correct last value as the variable is updated in the first statement.

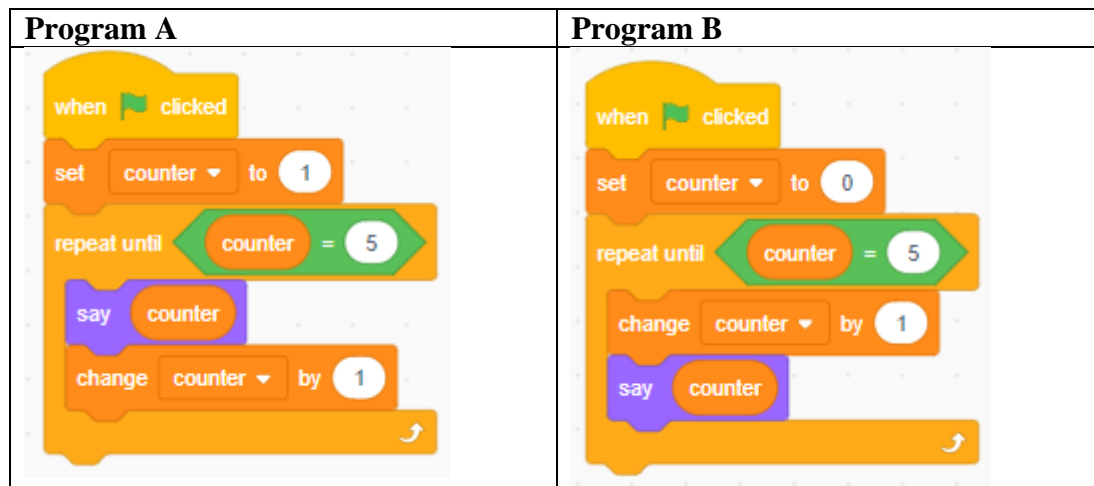


FIGURE 6: EXAMPLE PROGRAMS TO ILLUSTRATE TRANSITIONAL THEORY 8

Sorva identified this behaviour as a misconception, but I suggest that this exemplifies a transitional theory, as it is based on a reasonable assumption and existing experience with loops. Papert’s term ‘transitional theory’ seems much more appropriate, as it suggests something that is under development but not necessarily wrong. When the learner encounters a program like Program B, and find their current theory does not work in that situation, they may refine their theory through exploration of the program under different conditions.

3.2.4 Pedagogical Approaches from a Constructionist Perspective

3.2.4.1 Program Comprehension

As previously mentioned, much of the contemporary research in the field of computer science education takes an implied cognitivist paradigm, viewing the mind as context-free (Tenenber & Knobelsdorf, 2014). This approach also seems to have been applied to much of the research applying program comprehension to programming education. For example, the review conducted by Schulte et al. (Schulte et al., 2010) focuses on program comprehension models which lean more towards the cognitivist view of the mind. They do make reference to what they call constructivist viewpoints. However, they do not include them in their review.

The constructivist view of program comprehension sees it as a learning process. From this point of view the programmer always has some pre-existing knowledge

when they begin the program comprehension process. The knowledge of the program is not empty but incomplete. The way that a programmer reads a program is shaped by the nature of their pre-existing knowledge (Rajich, 2002). As the nature of pre-existing knowledge will vary significantly between individuals, so will the way in which they comprehend programs.

From a constructionist perspective, the focus of program comprehension should be on the learner, rather than the material they need to understand, as the learning process is different for every learner. Learning materials and the way they are presented is a way to facilitate the learner to build their own individual understanding (Exton, 2002). For example, the teacher and the tools could provide multiple different pathways to building an understanding through the material. This would enable learners to utilise the materials in the manner that suits them best (Exton, 2002). This approach ties in with Papert's constructionism and these tools could be realised through the creation of microworlds.

3.2.4.2 Pair Programming and Collaboration

The literature has demonstrated that pair programming and other forms of collaboration are effective in improving learning outcomes in programming education. However, Papert's constructionist approach does not seem to factor in the value of collaboration, although other constructionists did place a high level of importance on the role of collaboration. Hoyles and Noss (1992) claimed that small group work was essential to bridge the gap between a learner's individual sense of a concept and externally defined meaning. This point of view ties in with Vygotsky's use of sense and the emphasis he placed on the importance of social interactions in the learning process (Tudge, 2012).

3.2.4.3 Instructional Approaches

We have already established that constructionist approaches do not discount the value of direction in the learning process, however it is important to strike the right balance. This means that frameworks that are designed to inform the choice and arrangement of learning activities can be applied to the constructionist approach. In fact both the Use-Modify-Create and PRIMM methods place an emphasis on the importance of starting with an example program rather than a blank screen, an

approach that was actively employed in constructionist research during the 1980s and 90s, as exemplified in Hoyles & Noss' work on developing pedagogies for Logo based microworlds (Hoyles & Noss, 1987, 1992).

3.3 Summary

This chapter has highlighted that although many of the strategies that have been developed to address the challenges of learning to program have constructionist roots, the approach taken to data collection and analysis in research in contemporary literature usually takes a cognitivist lens. This has resulted in an approach which sees there being a right and a wrong way of programming, and there being only one way of seeing programming concepts. This is seen in the way in which misconceptions are discussed in the literature. Papert proposed that they be called "transitional theories" rather than misconceptions, as they are not necessarily wrong; they are an essential part of the learning process. Transitional theories are constructed and evolve through the bringing together of new and existing experiences. This view ties in with Vygotsky's *perezhivanie* and its role in the development of sense. For the context of my research, I see transitional theories as a part of an individual's sense and therefore in my analysis, I explore the transitional theories that learners develop in relation to programming, and how these theories evolve over time.

Learning approaches that the literature has highlighted, range from the type of tool to use, to the instructional approach to employ. A variety of tools have been suggested. However, many of these can be grouped under the heading of microworld, particularly if any approach which provides an environment with a set of pre-defined resources are included, specifically designed to enable the discovery of a concept through exploration and the creation of a public entity. When selecting an instructional approach, the research highlights the importance of employing pedagogies which facilitate the development of program comprehension strategies. Such approaches include Use-Modify-Create and PRIMM (Predict, Run, Investigate, Modify, Make). Additionally, approaches that facilitate collaboration, such as pair programming, have been shown to be effective. However, Papert does not place much emphasis on the importance of collaboration, but this is an important part of Vygotsky's theories of learning. In my own research I apply the concept of the

microworld to the learning situations which I design. This includes the physical programming language, the way in which it is set up, the activity structure, the use of non-electronic tangible tools and facilitating collaboration. The structure of the activities is informed by the PRIMM framework, which ties in with the programming approach employed in early constructionism.

Criticisms of the application of the constructionist approach to programming education have been identified, and seem to stem from a misunderstanding of the nature of discovery-based learning in a constructionist context. It is suggested that the constructionist approach is purely exploratory, with little teacher direction and therefore it needs to be blended with other approaches to be effective. However, looking back at early constructionist literature reveals that it was always acknowledged that a certain amount of teacher direction is necessary, and the amount needed will vary from learner to learner. It is important to maintain a balance between exploration and direction for the best outcomes, and this is the approach I took towards the design of the activities I employed in my research.

Chapter 4. **Accessibility of Programming Education for Blind and Partially Sighted Learners**

This chapter provides an overview of the current state of the literature relating to programming education for blind and partially sighted learners. It starts by looking at the research which aims to address the barriers visually impaired learners face when learning to program.

4.1 Making Programming Accessible to Learners with Visual Impairments

Programming can be challenging to learn, and for blind and partially sighted learners there are numerous additional barriers to the learning process. Many modern programming environments are inaccessible to these learners, being challenging or impossible to interface with using a screen reader (Baker et al., 2015; Stefik, Hundhausen, et al., 2011) and user interfaces often employ highly graphical depictions (Ludi, 2013). Kane & Bigham (2014) identified the following criteria for the development of environments in which VI children can learn to program:

“Programming tools must be accessible to the student and must work with the assistive technology that he or she uses.”

“The student must be provided with programming tasks that hold their interest and provide encouraging feedback.” (Kane & Bigham, 2014, p. 257).

The following section sets out to provide an overview and discussion of the different strategies that have been employed in order to make learning programming accessible to learners with visual impairments. Additionally, areas that require further research are identified and discussed.

4.1.1 Overview of Literature

Four main themes emerged when examining the literature relating to this field: making text-based languages accessible, making block-based languages accessible, physical artefacts as well as auditory and haptic feedback. Each of these themes is explored in turn in the following sub-sections.

4.1.2 Making Text-Based Languages Accessible

4.1.2.1 Accessibility of Programming Environments

A large number of programming environments are not fully compatible with screen readers, and are therefore not accessible to many blind and partially sighted learners. In order for software to be accessible to blind users, a screen reader needs to be able to vocalize not only the content, but also the interface (Asakawa & Leporini, 2009). In the case of programming environments, many of them have elements of the content or interface that cannot be effectively vocalized using a screen reader. IDLE, the standard programming environment for the Python programming language, is a good example of the challenges that visually impaired users may face. The way that the user interface of IDLE is designed makes it difficult for screen readers to interpret and vocalize the content. This content includes both the code the user has written and outputs that the code produces.

One approach that has been taken to address the inaccessibility of programming environments, is the use of a standard text editor alongside a screen reader (Bigham et al., 2008; Cheong, 2010; Kane & Bigham, 2014). A drawback of this approach, is the loss of debugging tools that are standard in most modern programming environments, and are designed to support users in the identification and correction of ‘bugs’. Tools have also been developed to improve the accessibility of programming environments. For example the Wicked Audio Debugger (WAD) was developed to work with the popular Visual Studio programming environment, to assist blind and partially sighted programmers with the debugging process by aiding navigation through the use of audio cues (Stefik, Alexander, Patterson, & Brown, 2007).

An alternative strategy is the development of new, accessible programming environments. An example is JavaSpeak, which was developed as a tool to assist visually impaired undergraduate students learn how to program in Java (Francioni & Smith, 2002; A. C. Smith, Francioni, & Matzek, 2000). Java was chosen as the focus due to its popularity in undergraduate computer science courses. JavaSpeak is based on the concept of Emacspeak (Raman, 1996), which has a speech interface aimed at experienced programmers. Unlike Emacspeak, JavaSpeak is designed for undergraduate students that are learning to program, enabling them to experience

their code at different granularities. The development process of the JavaSpeak environment has been described, however there is no evidence of evaluation of the tool in use.

More recently, the JBrick programming environment was developed to make the programming of Lego Mindstorms robots accessible (Ludi, 2013). The NXC language (Not eXactly C) has been used in outreach programs along with the BricxCC programming environment, to enable blind and partially sighted learners to program Lego Mindstorms robots (Dorsey, Chung, & Howard, 2014; Ludi & Reichlmayr, 2011). However, the BricxCC programming environment is not fully compatible with JAWS (a popular screen reader), for example line numbers are not read out, making code navigation challenging. JBrick was developed as an alternative to BricxCC to address the compatibility issues. BricxCC works with common screen readers and braille displays, enables code to be easily located by line number, and provides both audio and visual feedback (Ludi, Ellis, & Jordan, 2014).

A common theme that occurs among the literature, is the difficulty visually impaired learners have navigating their code and understanding the overall structure when using a screen reader, particularly when blocks of code are nested within each other (Bigham et al., 2008; Kane & Bigham, 2014; Ludi et al., 2014). This can often result in learners inserting code in the incorrect position. There are steps that can be taken to mitigate these difficulties; in order to gain a better understanding of their position in the code, learners can be encouraged to move the text cursor in order to hear the characters read out. In addition, learners can also be provided with code samples in braille to help them develop an understanding of the overall structure of the code.

The challenge of navigating the code and understanding its structure was considered during the development of StructJumper, a plugin for the Eclipse programming environment, which enables visually impaired users to navigate through a program written in Java (Baker et al., 2015). StructJumper generates a tree that is made up of the nested structures contained within the program; this enables the user to easily jump between each nested structure in the code. The participants that took part in a small-scale evaluation of StructJumper found that it helped them speed up their navigation through the code.

4.1.2.2 Accessibility of Programming Languages

Another important consideration is the choice of programming language; the complex syntax of many languages can make typing mistakes more likely and debugging more challenging. Languages such as Ruby, which use mainly text and limit the number of non-alphanumeric symbols, are preferable as they are less likely to cause problems with screen readers (Kane & Bigham, 2014). In their study, Kane and Bigham also considered Python, as it meets most of the previously mentioned criteria. However it also uses white space, which could cause confusion as it is ignored by most standard screen readers. During the course of their study, which took place over a week, and involved 12 blind and partially sighted learners, Kane and Bigham found that the students were successful in writing programs in Ruby. However the mispronunciation of some of the terms by the screen reader caused minor challenges.

There are text-based languages that have been designed specifically for visually impaired users, for example the APL (Audio Programming Language) was developed by visually impaired learners for visually impaired learners (Sánchez & Aguayo, 2006). APL features a reduced set of commands which can be accessed and selected through a circular command list, with no requirement to memorise commands. Sánchez & Aguayo (2006) conducted a small usability study of APL, the results of which indicate that the language enables learners to understand and apply core programming concepts, such as variables, selection and iteration. The authors note that APL is not suitable for completion of high-level computing tasks and alternative solutions need to be investigated.

In 2011 Stefik, Hundhausen, & Smith conducted an exploratory study to evaluate the accessible programming environment Sodbeans, along with the Hop programming language, which they developed. Sodbeans is aimed at middle and high school students and makes use of audio cues for navigation, along with an auditory debugger for the Hop programming language. The findings from an evaluation involving 12 learners indicate an increase in learner self-efficacy, and a decrease in concerns about performance, after participation in a programming workshop that employed Sodbeans and Hop.

The Hop programming language was developed further, becoming Quorum, a language designed for all, while still being accessible to visually impaired learners (Stefik, Siebert, Stefik, & Slattery, 2011). The development of Quorum was informed by empirical studies investigating the intuitiveness of the syntax of different languages and the accuracy rates of novice programmers using them (Stefik & Siebert, 2013).

4.1.2.3 Consideration of Learner Capabilities

It is also important to consider that the level of vision among visually impaired learners will vary considerably, as will their preferred assistive technologies (Bigham et al., 2008; Ludi et al., 2014). Experience with assistive technologies may also vary. Bigham et al. (2008) found that students that were already proficient in the use of a screen reader were the most successful. Another factor that can impact on progress of blind and partially sighted learners, is their familiarity with keyboard layout, with typing skills also being identified as an important skill for learning to program in a text-based language (Ludi, 2013; Ludi et al., 2014).

4.1.2.4 Working with Graphical User Interfaces

The accessibility of tools designed to create graphical user interfaces (GUIs) also needs to be considered, as existing tools that are employed to generate GUIs are either not accessible, or very challenging to use for visually impaired learners as the nature of the feedback from these tools is usually highly visual in nature. In order to address this issue Siegfried (2006) developed a scripting language to enable blind and partially sighted programmers to produce Visual Basic Forms. More recently, Konecki (2014) developed GUIDL, a tool that enables visually impaired learners to create GUIs for their programming projects, through the use of a simple scripting language that follows the Comprehensive Assistive Technology (CAT) model. GUIDL was evaluated by a small group of adult novice programmers who found they were able to use the tool to successfully create GUIs that could be used in their own programs.

4.1.2.5 Summary

Table 2 provides a summary of the different strategies that have been discussed, along with any bespoke tools which were employed in the studies.

TABLE 2: OVERVIEW OF STRATEGIES FOR MAKING TEXT-BASED LANGUAGES ACCESSIBLE

Strategy	Reference(s)	Bespoke Tools
Use of plain text editors with screen readers	Bigham et al. (2008) Cheong (2010) Kane & Bigham (2014)	
Development of tools to improve the accessibility of existing programming environments	Stefik, Alexander, Patterson, & Brown (2007) Baker et al. (2015)	WAD StructJumper
Development of new, accessible programming environments	Smith, Francioni, & Matzek (2000) Francioni & Smith (2002) Ludi (2013) Ludi, Ellis, & Jordan (2014)	JavaSpeak JavaSpeak JBrick JBrick
Use of existing programming languages with simple syntax that are less likely to cause problems with screen readers	(Kane & Bigham, 2014)	
Development of new, accessible programming languages	Sánchez & Aguayo (2006) Stefik, Hundhausen, & Smith (2011) A. Stefik, Siebert, Stefik, & Slattery (2011)	APL Sodbeans & Hop Quorum
Development of accessible tools for creating GUIs	Siegfried (2006) Konecki (2014)	GUIDL

Although there are a number of studies focusing on teaching visually impaired learners to program in a text-based language, these mainly focus on high school and undergraduate students. The following section will look at the accessibility of block-based languages, which are targeted at students in primary school.

4.1.3 Making Block-Based Languages Accessible

When learning how to program, a significant amount of time is spent learning the syntax of a specific language; this can potentially hinder the development of an understanding of the core programming concepts. As mentioned in Chapter 2, Block-based languages such as Scratch (Maloney et al., 2010) enable learners to

develop programs by snapping on-screen blocks together, removing the need for them to learn the complex syntax of a text-based languages.

Block-based languages are intrinsically visual and are therefore not accessible to most visually impaired learners. There is a need for an alternative to block-based languages such as Scratch (Koushik & Lewis, 2016; Ludi, 2015). One such alternative is Noodle, a programming system for creating sound and music that has program elements which can be inserted and arranged purely using keyboard commands (Lewis, 2014). The concept of Noodle is promising; however, it does not appear to have been trialled with learners, and the language used in the audio feedback is not appropriate for primary school children. This makes it an unsuitable choice for the introduction of programming to young blind and partially sighted children.

The Lady Beetle and World of Sounds programming environments were developed in order to introduce visually impaired children to the basic concepts of programming (Jašková & Kaliaková, 2014). The Lady Beetle programming environment enables the learner to select single word commands, without having to type them. These commands control the movement of a beetle across a grid. As the beetle moves, the coordinates of the current square are read out; this requires the learners to have prior knowledge of coordinate systems, which could potentially have an impact on usability for younger learners. World of Sounds, on the other hand, enables learners to create simple programs that produce sequences of sounds.

Ludi (2015) and her team have been working on making the Blockly language accessible to visually impaired learners. The language that Ludi and her team are developing will enable navigation purely by keyboard, and also incorporate audio cues in order to communicate the level of nesting. Following on from the work on Noodle, Lewis has been working with Koushik in the development of another accessible Blockly-based language called the Pseudospacial Blocks (PB) language (Koushik & Lewis, 2016). Pseudospacial refers to the distorted nature of the geometry of movement, for example when the user moves left from the workspace into the toolbox, they will end up in the same position in the toolbox regardless of their starting vertical position in the workspace. In PB the learner selects an insertion point using the keyboard and they can select the program element they want from a

filtered list; the program elements are filtered by syntactic category. Koushik and Lewis (2016) argue that PB has advantages over visual languages for all learners as invalid program blocks for a given space are filtered out.

The development of these accessible Blockly-based languages is a promising step forward in the quest to find an accessible alternative to block-based languages, however they have not yet reached the user-testing stage. Additionally, The Lady Beetle and World of Sounds programming environments have been shown to be effective in teaching sequence and iteration, however selection is yet to be addressed. All of these potential solutions could still present learners with difficulties gaining an understanding of the overall structure of their code when using a screen reader.

4.1.4 Physical Artefacts

4.1.4.1 Physical Computing

The physical computing devices, such as programmable robots, make them a common tool for the teaching of introductory programming and it has been shown to be just as appealing to visually impaired learners (Ludi, 2013). When teaching computing with robotics, the robots can either be pre-assembled, or learners can be required to build their own robots as part of the learning process. This has its own challenges, particularly for blind and partially sighted learners.

Dorsey Rayshun, Chung Hyuk, & Howard (2014) conducted an evaluation of four educational robotics kits during a series of summer workshops, which investigated their suitability for use with partially sighted learners. In each workshop the blind and partially sighted learners were paired with a sighted buddy and tasked with building robots using the various kits. The LEGO Mindstorm RCX was found to be the easiest for visually impaired learners to work with, requiring the least support from their sighted buddies.

A number of studies have been conducted, which investigate outreach programs designed to increase participation of visually impaired students in computing using robotics (Dorsey et al., 2014; Ludi, 2013; Ludi et al., 2014; Ludi & Reichlmayr, 2011). The findings of these studies indicate that after the workshops the confidence

level of the students in programming improved, as did their desire to take computing in school or pursue it as a career.

4.1.4.2 Physical Programming Languages

Most systems used in physical computing, whilst being physical themselves, are still programmed using a GUI on a computer. In physical programming languages commands are represented by physical objects which can be joined together to create programs. The Tern physical programming language uses wooden blocks that can be joined together in order to construct programs and a webcam is used to convert physical into digital code (Horn & Jacob, 2007b, 2007a). Tern was initially evaluated over the period of one week with nine sighted children. The children used Tern to program robots, however, not all of them were able to understand the effect of their programs on the robot. This may be partially down to the delay between code parsing and execution, as it has to be converted to digital code using a webcam connected to a computer. A novice programmer may test their programs frequently during development; these short delays between code parsing and execution could be disengaging for learners, who are making a number of small incremental changes that they want to test at each stage.

The physical nature of physical programming languages means they have the potential to be a powerful learning tool for visually impaired children; however, Tern itself is not accessible, as the individual blocks are not distinguishable through touch alone. On the other hand there is Code Jumper, a physical programming language that is designed to be inclusive of blind and partially sighted learners (Thieme et al., 2017). Code Jumper features pods which can be joined together to create programs that produce sound and music. Each pod features dials, which act as parameters and enable the learner to change the sound sample or note and the duration. The physical nature of Code Jumper programs could potentially enable the learner to gain an understanding of the structure of the whole program.

4.1.4.3 3D Models

It is common practice for computing teachers to use diagrams, graphics or animations to illustrate programming concepts such as data structures; “most tools used to teach data structures, algorithmic thinking and basic programming are

visually oriented” (Papazafropoulos, Fanucci, Leporini, Pelagatti, & Roncella, 2016, p. 491). While assistive technologies enable visually impaired learners to access information, they are unable to present a complex concept in a simple form in the same way a visual representation can.

3D models can be used to represent abstract concepts in a way that is accessible to blind and partially sighted learners. As part of their research Stefik et al. (2011) interviewed teachers in one school for visually impaired children and found that where possible, new concepts should be introduced through the use of physical objects. In response to this, they developed ‘manipulatives’ for teaching key programming concepts, such as variables. Jašková & Kaliaková (2014) used a tactile table consisting of a 10 x 10 grid to teach blind and partially sighted children how to write simple algorithms. The children were given the task to write a sequence of commands in a text editor, that guided a bee to follow a pre-set path through the tactile grid. The learners would simulate the execution of the program by moving the bee with their hands.

With the advent of 3D printers, 3D models have become much easier to produce. Papazafropoulos et al. (2016) used 3D printed models in a small feasibility study to teach concepts such as data structures and algorithms to visually impaired children. The model they used features cylinders of varying heights, with the height representing the value of the element. The cylinders slot into a tray which represents the array. It was used to teach how sorting and searching algorithms could be applied to arrays. 3D printing was also used by Kane & Bigham (2014) as part of a week-long programming workshop, in which children produced code to generate physical visualizations of data. They found that the ability to generate and print their own tactile maps was extremely engaging for the children. However, the speed of 3D printing was a limitation as they had to be printed overnight. They also identified the need for universal tools that can be used to easily create tactile graphics.

Lego provides a quick and simple method of producing basic 3D models for use in the teaching of programming concepts to visually impaired learners. Capovilla et al. (2013) discovered this when they employed Lego models in the teaching of sorting and searching algorithms to a small group of adult blind and partially sighted learners. Once the learners had familiarized themselves with the algorithms using the

Lego models, they were then asked to solve sorting and searching tasks in a spreadsheet. All participants were able to complete the assigned tasks.

Table 3 provides an overview of the different strategies that employ physical artefacts to aid visually impaired students learn to program, along with the associated references.

TABLE 3: OVERVIEW OF STRATEGIES THAT EMPLOY PHYSICAL ARTEFACTS

Strategy	Reference(s)
Use of programmable devices such as robots	Ludi (2013) Ludi et al. (2014) Ludi & Reichlmayr (2011)
Physical programming languages such as Code Jumper	Thieme et al. (2017)
Use of 3D models to teach abstract concepts such as data structures which are traditionally taught using diagrams	Stefik et al. (2011) Leporini et al. (2016) Capovilla et al. (2013)
Use of physical representations to aid the development of algorithms in route-based activities	Jašková & Kaliaková (2014)
Getting learners to write algorithms to generate tactile maps, which are then 3D printed	Kane & Bigham (2014)

4.1.5 Auditory and Haptic Feedback

As previously discussed, some tools for learning to program produce sound as their primary output, for instance Code Jumper (Thieme et al., 2017) and World of Sounds (Jašková & Kaliaková, 2014). Sound can also be used to provide feedback to the user regarding the state of a system in order to improve usability. For example, sounds that vary in tone and pitch can be used to indicate the different states of a physical object or virtual representation, as can haptic feedback in the form of vibrations. PLUMB EXTRA (Exploring data sTRuctures using Audible Algorithm Animation) was developed to enable visually impaired undergraduate students to access simulations of algorithms designed to manipulate data structures (Calder et al., 2007). It is based on PLUMB, a system designed to enable blind and partially sighted learners navigate graphs (Calder, Cohen, Lanzoni, & Xu, 2006). The

PLUMB EXTRA system enables learners to explore the state of data structures at any point using a series of audio cues. In the Calder et al. (2007) study, the development of the system is described; however, the evaluation of the system is limited.

During a series of workshops, Dorsey Rayshun, Chung Hyuk, & Howard (2014) made use of different piano notes and vibrations in a Wii remote, in order to indicate the different states of a robot while navigating a maze. The results of this study indicate that if sufficient haptic and auditory feedback is provided, visually impaired learners are able to perform tasks that are considered to be highly visual.

4.2 Discussion

This review has demonstrated the dominance of text-based languages in the literature. This is despite the fact that in primary computing education block-based languages are most prevalent, as highlighted by the recent Royal Society Report (The Royal Society, 2017). According to the national curriculum (Department for Education, 2014), all children in England should learn the basic concepts of programming from the age of 5. However, the inherent inaccessibility of block-based languages, along with their widespread use in primary computing lessons can lead to visually impaired learners being excluded from programming lessons. Initial steps have been taken towards making block-based languages accessible to visually impaired learners, however there is still a long way to go and more research is needed.

Research relating to the use of text-based languages with blind and partially sighted learners, has identified the difficulty learners can have in gaining an understanding of the overall structure of their code, as they can only listen to one line of code at a time, putting a heavy reliance on short-term memory. Even though it has been shown that it is possible to make block-based languages accessible to visually impaired learners, this difficulty could still present a barrier for learners. Physical programming languages, on the other hand, could potentially enable blind and partially sighted learners to develop an understanding of the structure of the code through touch, as long as the individual blocks or elements used in the physical programming languages are physically different. Therefore, the use of physical

programming languages with visually impaired learners needs to be investigated in terms of learning processes and possible benefits.

The literature relating to text-based languages has identified a number of potential challenges for visually impaired learners, in addition to possible strategies to overcome them. This research can be used to inform the teaching of programming to high-school blind and partially sighted learners, however more research is still required. If visually impaired learners are successfully introduced to programming in primary school, through physical programming languages or accessible block-based programming languages, they will enter high-school understanding the basic concepts. This highlights the urgent need for research into strategies for making programming accessible to primary visually impaired learners.

4.3 Conclusion

Much of the research carried out in this space to date focuses on the development of interventions and their impact on student perceptions and engagement, with limited attention given to the pedagogy of teaching programming to visually impaired learners. This is certainly an area that warrants further research.

Currently the most popular languages for introductory programming in primary schools in the UK are block-based (The Royal Society, 2017), which are currently not accessible to visually impaired learners. Therefore, there is a need for further investigation into potential accessible alternatives to block-based languages.

Physical programming languages are a promising candidate, given their potential to enable learners to gain an understanding of the overall structure of their code.

Chapter 5. **Research Methods**

5.1 Introduction

My research set out to explore the ways in which a sense of programming constructs is manifested, among learners with visual impairments, during the process of learning with the aid of physical programming languages. I conducted a study involving the delivery of a series of coding workshops, that were video-recorded for the purposes of qualitative analysis, in order to answer the following aim and associated research questions:

To address the dual concern of understanding the processes by which blind and partially sighted learners develop their sense of programming concepts, while building learning ecologies that would support engagement in these processes.

1. How do blind and partially sighted learners express their sense of sequence, threading, repetition, selection and variables?
2. What do these expressions reveal about the learning processes by which sense of programming develops?
3. How do the design structures embedded in the learning ecology support these learning processes?

In this chapter I will outline and discuss the methodological decisions I made during my research. I will start by outlining my epistemological position, followed by methodological choices, and finishing by outlining the data collection and analysis methods employed in my pilot and main studies.

5.2 Epistemology

The underpinning element of the research process is epistemology. A researcher's epistemological position describes their theory of how knowledge can be generated and validated (Mason, 2002). Opinions regarding what constitutes an epistemological viewpoint vary; for the purposes of my research, I am drawing on the three broad views defined by Crotty (1998) and adapted by Gray (2014): these

are objectivism, subjectivism and constructivism¹. The main distinguishing feature of each of these views, is the different relationships that exist between the object being observed and the human subject that is observing it.

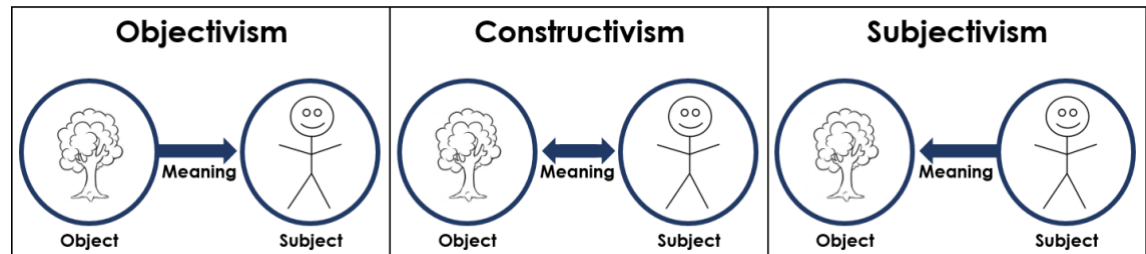


FIGURE 7: ILLUSTRATION OF EPISTEMOLOGICAL PERSPECTIVES

From the objectivist perspective, objects have an inherent meaning which is received by the subject; illustrated in Figure 7. Bernstein argues “What is ‘out there’ (object) is presumed to be independent of us (subjects), and knowledge is achieved when a subject correctly mirrors or represents objective reality” (Bernstein, 1983, p. 9). In other words, “reality exists independently of consciousness” (Gray, 2014). From this view point it is considered that there is objective truth outside the human experience, and the truth can be tested and validated objectively (Popper, 1963), implying that objective truth can be discovered through research. In practical terms, a social scientist operating from an objectivist viewpoint, aims to discover universal laws through their research that explain how society operates (Cohen, Manion, & Morrison, 2017).

Subjectivism and constructivism both value the role of human consciousness in the creation of meaning; however, their view of the role the object plays in this process differs. The subjectivist perspective holds that meaning is imposed on objects by humans; it does not develop as a result of interactions, as is the case with constructivism/constructionism (Gray, 2014); illustrated in Figure 7. From the subjectivist perspective, true knowledge can be considered as a belief that is well founded or justified through evaluation against particular rules or criteria (Popper,

¹ Constructivism is employed as an umbrella term that encompasses a number of different theories that share a similar epistemological outlook.

1963). Through their research, the subjectivist social scientist aims to discover the ways in which different people interpret their world (Cohen et al., 2017).

Now we will turn our attention to constructivism. In his original model Crotty (1998) uses the term constructionism, whereas in the version adapted by Gray (2014) it becomes constructivism. However, both authors infer the same general meaning for both terms. In their view, meaning is not independent from human thought, it is constructed through our interactions with the external world, or through interactions between the subject and object, as illustrated in Figure 1. This implies that different meanings can be constructed from the same phenomenon, and each of those meanings could be considered equally valid, which ties in with Vygotsky's views regarding sense versus meaning. Therefore, from the perspective of research, all findings, to some extent, reflect the researcher's standpoint (Blaikie, 2010).

Some scholars hold the view that constructivism can be considered as a branch of psychology, often associated with Piaget, who placed emphasis on the individual and how they construct meaning through interaction with their environment (Burr, 2015; Scott & Marshall, 2009). From a philosophical viewpoint constructivism can also be considered as a sociological position, which concerns how knowledge in general is produced (Crotty, 1998; Scott & Marshall, 2009). On the other hand, some view the meaning of the term constructivism as an open question, as there is no clear and unchallenged usage of the term, nor an agreement as to how it was formulated (Velody & Williams, 1998). Lynch (1998) views the terms constructivism and constructionism as synonymous, opting to utilize the former in his own writing. From his perspective "The constructivist movement might best be described as a fragile coalition of marginal, nomadic academic bands" (Lynch, 1998, p. 14).

In education circles the term constructionism is often associated with the work of Papert, as discussed in Chapter 3. Therefore, to avoid confusion I have opted to adopt the approach of Lynch (1998) and Gray (2014) and use constructivism as an umbrella term, that encompasses a number of different theories that share a similar epistemological outlook. In Table 4 I have provided a summary of the different theories that can be viewed as coming under the constructivist umbrella.

TABLE 4: THEORIES UNDER THE 'CONSTRUCTIVIST' UMBRELLA

Theory	Type of Theory	Description	Key Contributors
Cognitive Constructivism	Psychological	The individual plays an active role in the construction of meaning through interaction with their environment. Places an emphasis on the cognitive development of the individual (Amineh & Asl, 2015).	Piaget (1982)
Cultural-historical Theory ²	Psychological	The individual generates meaning through social interaction with others and their environment. Places an emphasis on the importance of language as a psychological and developmental tool (Amineh & Asl, 2015).	Vygotsky (1978)
Social Constructionism	Sociological	Places an emphasis on the role of social interactions in the generation of knowledge and meaning within society (Scott & Marshall, 2009).	Berger & Luckmann (1966)
Papert's Constructionism	Pedagogical	Proposes that learning is most effective when people are actively creating artefacts in the real world. (Ackermann, 2001)	Papert (1980)

A term closely related to epistemology is ontology, which considers theories of existence. Ontological and epistemological perspectives often arise together, making them challenging to separate conceptually. In some cases the terms are even confused with each other in the literature (Crotty, 1998). In regard to my own epistemological position, I believe that meaning is constructed rather than discovered. This could potentially lead to either a constructivist or subjectivist viewpoint. However, it is my view that meaning is constructed through interaction with objects, rather than meaning being imposed on objects, and this ties in with

² Vygotsky's cultural-historical theory is also known as social constructivism to differentiate it from Piaget's constructivism, as it has a much greater emphasis on the importance of social interactions.

Vygotsky's views on the creation of meaning. Therefore, I decided to take a constructivist perspective to this research.

5.3 Methodology

When choosing my methodology, I needed to consider which approach would enable me to gather the in-depth data required to address my research questions, in addition to ensuring it aligned with my epistemological stance. I required an approach which would enable me to develop theories, regarding the way in which visually impaired learners express and develop a sense of key programming concepts. One potential approach was design-based research, which I will explore further in the following section.

5.3.1 Design-Based Research

It has been claimed that a great deal of educational research fails to transform into lasting change and improvement (Bradley & Reinking, 2011a; Engeström, 2011; Jacob, 1997; The Design-Based Research Collective, 2003). Jacob (1997) suggests that this is partly due to the positivist lens that is applied to many studies, which focus mainly on cognitive concerns and outcomes in the absence of context. Many researchers consider the minimising of context necessary to maximise fidelity and replicability of findings. However it is possible that this is also having a negative impact on successful implementation in real world classrooms. On the other side of the coin, is the interpretivist approach of educational anthropology, which focuses almost totally on context. Jacob argues that a balanced approach which brings these two aspects together is required. She suggests using approaches that draw upon the cultural-historical tradition which has its roots in Vygotsky's work.

These approaches can come under the broad heading of design-based research (The Design-Based Research Collective, 2003) and encompass methodological approaches to education research, that seek to address the gap between research and classroom practice, and acknowledge the complexities and countless variables within education settings (Bradley & Reinking, 2011b). One of the key benefits of design-based research is its ability to develop models of learning processes, which are contextualised, and valuing the importance of context can lead to "improved theoretical accounts of teaching and learning" (The Design-Based Research

Collective, 2003, p. 7). In traditional methods the researchers may try to boil the context down to a list of factors, which are separate from the actual intervention. Whereas traditional research may focus on perfecting a particular product, design-based research aims to investigate the nature of learning in complex systems and to refine hypotheses.

There are various methodologies that can be considered to come under the umbrella of design-based research, and they all share five key characteristics:

- Seeking to develop both learning environments and models
- Development and research go through cycles which include designing, enacting, analysing and redesigning.
- Research results in models regarding learning processes that can be shared with practitioners and educational designers.
- Should account for how designs operate in authentic settings and focus on interactions to develop understanding of learning processes.
- Requires methods which can record and draw connections between processes of enactment and outcomes.

Adapted from The Design-Based Research Collective (2003)

I will now outline how the needs of my research align to these characteristics.

5.3.1.1 Seek to develop both learning environments and models

To address my research questions, I needed to design a learning environment which employed a physical programming language and a series of activities. I borrowed the term ‘microworld’ from Papert (1980) to encompass the tools and pedagogical approach applied to a particular activity. The data collected while the learners interacted with each microworld enabled me to develop models regarding the development of a sense of programming concepts, among visually impaired learners using physical programming languages.

5.3.1.2 Development and research go through cycles which include designing, enacting, analysing, and redesigning

I initially conducted a pilot study to test both my data collection methods and the design and delivery of activities. Analysis of these data facilitated the redesign of the activities for the main study. Additionally, throughout the delivery of the main study I adapted the shape of the intervention based on my observations of the learners. For example, if I observed that some learners found expressing their sense of a concept in a particular manner helpful in solving certain problems, I would introduce that form of expression to the other learners.

5.3.1.3 Research should result in models regarding learning processes that can be shared with practitioners and educational designers

It was important for me to develop models which encompass the development of a sense of programming constructs among learners with visual impairments, which could both inform the practice of educators and the design of future interventions.

5.3.1.4 Should account for how designs operate in authentic settings and focus on interactions to develop understanding of learning processes

To obtain the rich data required to answer my research questions, I needed to carry out the intervention with small groups of learners in authentic educational settings. I took the approach of Roth (2001), for whom the dual role of teacher-researcher is an important part of his research, as it enables him to work with learners as they build their understanding, and develop interpretations of what is important. His position in the role of teacher enables him to test and refine his interpretations through rearrangement of the learning context.

5.3.1.5 Requires methods which can record and draw connections between processes of enactment and outcomes

To be able to draw conclusions regarding the way in which the sense of each learner evolved, and what this can tell us about their learning processes, I needed to gather rich data spanning multiple sessions. The method of capture needed to record the different ways in which the learners expressed their sense of different programming constructs. I chose video recording as it enabled me to capture expressions of sense through sound, gestures and actions.

5.3.2 *Formative Experiments and Design Experiments*

As previously mentioned, there are multiple forms of design-based research, and two of the most prominent methodologies are formative experiments and design experiments. The design experiment is an approach which enables researchers to develop ‘humble’ theories, regarding learning processes specific to a domain, and it is suggested that the emphasis on theories is important if we want educational improvement to be a long-term process (Bradley & Reinking, 2011a). Cobb et al. (2003) characterised design experiments as resulting in a deeper understanding of a learning ecology, a term used to encompass the problem students are asked to solve, forms of discourse that are facilitated, classroom norms, the tools provided and how teachers manage the relationship between these elements. Ecology is used as a metaphor to highlight that these designed contexts are interacting systems rather than separate factors.

Formative experiments are another form of design-based research which places emphasis on achieving a specific pedagogical goal, rather than the development of theories (Bradley & Reinking, 2011a). Engeström (2011) criticised the design experiment method, suggesting that it does not value to the role of learner agency in the research process, and does not make explicit its theoretical underpinnings. I would argue that although learner agency is not as explicit in design experiment literature, it is still present. For Engeström the link to Vygotsky’s method of double stimulation is a core aspect of design research and it is strongly indicated in formative experiments or interventions.

The double stimulation strategy makes a significant departure from the standard experimental method, which focuses on the product or outcome, as it shifts emphasis onto the process of the learner engaging with a task (Ellis, 2010). It achieves this through the use of two stimuli. The first or the ‘stimulus-end’ is the problem the learner needs to solve. The second stimuli, or ‘stimulus-means’ is a tool that could be used by the learner to support the problem-solving process (Vygotsky, 1987). In fact, Vygotsky referred to the ‘stimulus-means’ as a series of stimuli, implying the possibility of multiple tools. By observing the way in which the learner interacts with the ‘stimulus-means’ in order to solve the problem, we can gain an insight into the learning process and the development of mental functions (Ellis, 2010).

The double stimulation strategy is a key part of Vygotsky's experimental-genetic method, which enables the discovery of the genetic history of psychological functions and the inter-connections between them (Vygotsky, 1997). As discussed in Chapter 2, psychological functions are appropriated from *perezhivaniya*, which consist of experiences and the working over of them (Mattosinho Bernardes, 2018). Each *perezhivanie* is a whole, and cannot be decomposed into its constituent parts without losing the characteristics that are inherent to the whole (Blunden, 2016), as it is the "indivisible unity of personal characteristics and situational characteristics" (Vygotsky, 1994, p. 342). Vygotsky proposed the experimental-genetic method to enable the products of *perezhivaniya* to be highlighted at certain moments in time through double stimulation, without breaking them down, to build a picture of the genetic development of psychological functions (Vygotsky, 1997).

Engeström (2011) suggests that the tools or stimulus-means do not necessarily have to be introduced by the researchers, they could be spontaneously created by the learner themselves. An example of a spontaneously created stimulus-means could be the application of a technique that they have seen others use in other contexts. Even when using tools provided by the researchers, learners may not use them in the way that they anticipated, and in the context of traditional psychological research this could be viewed as a negative, as it adds an additional variable into the situation which, for some, could impact the validity of the experiment. However, for Engeström, these situations should be valued and used to inform the ongoing research process, as they are examples of learner agency in the research process and are techniques that have worked in authentic settings.

Although the model of formative experiment that Engeström and his team at the Change Laboratory developed is grounded in theory and facilitates learner agency, it is also specifically geared to solve workplace problems. The example cases which are included in the publications produced by the Change Laboratory mostly deal with supporting teams in overcoming challenges in the workplace (Engeström, 2011; Sannino, Engeström, & Lemos, 2016). When the model is applied to education settings, the cases revolve around teacher development rather than the learning processes of children. Whereas, design experiments have been applied to a wide

variety of settings, in particular education. They have been employed to examine both the learning processes of students and the professional development of teachers.

Taken individually, neither of these two approaches would be suitable to address my research questions. As previously discussed, the design-based research approach is appropriate for my research. However, it could be considered a little vague as a methodology. For this reason, I decided to add more depth to it by drawing in relevant details from formative experiments and design experiments. I will outline the aspects that I have selected, and explain why they are important, when answering my research questions.

5.3.2.1 Theoretical foundation

The methodology that I employ needs to have a strong theoretical underpinning which ties in with the Vygotskian approach I have taken to carrying out my research. Although design experiments do have Vygotskian roots, there does not seem to be an explicit connection between his theories and the techniques employed within this approach. Formative experiments, on the other hand, specifically employ Vygotsky's double stimulation strategy, which provides a window into the development of mental functions. The double stimulation strategy is appropriate for my research, as it enables me to provide stimuli to provoke development of sense relating to programming concepts. The different forms of expression that learners produce during this process can provide a window into how their sense develops over the course of the sessions.

5.3.2.2 Collection of rich data in educational settings

As previously mentioned, much research which has taken on the formative experiment approach has been conducted within workplace settings, whereas design experiments are usually employed within educational settings. Design experiments typically take on a number of different forms. One form is the classroom experiment which involves the researcher working with the teacher to design and evaluate a learning ecology for a class of students (Cobb et al., 2003). Another form is the teacher-experimenter and student. In this setting the researcher teaches a series of sessions with a small group of students, or even one-to-one. This form of design experiment aims to create a version of the learning ecology on a small scale so it can

be studied in great detail. The methodological approach I chose needed to facilitate the collection of rich data, that would capture the various ways in which a visually impaired learner could express their sense of different programming concepts. Practically speaking, working with small groups, or even one-to-one, seemed to be most appropriate for this kind of data collection. Therefore, I decided that conducting teacher-experimenter and student design experiments would be appropriate to address my research questions. Additionally, as previously mentioned, taking the dual role enabled me to continually refine the intervention based on how the learners responded.

5.3.2.3 Pedagogical goals and models

Formative experiments are primarily focused on achieving a pedagogical goal, whereas design experiments place a greater emphasis on the development of models of learning processes. In my research I am seeking to achieve a pedagogical goal in the development and refinement of an intervention, and at the same time I wish to develop models which encompass how learners express their sense of programming concepts. Therefore, I chose to employ a mixture of both approaches in my research.

5.3.2.4 Learner agency

Learner agency is an important consideration in my research, as I am exploring how learners express their individual sense of programming concepts, and therefore unexpected or unique forms of expression should be valued and shape the direction of the intervention. In the context of my research, learner agency refers to the extent to which they are able to influence the intervention and direction of the research. For example, if the learners found a style of task to be particularly engaging, I would try and incorporate more of them within the intervention. Additionally, if a learner developed a novel stimulus-means which they found to be helpful, I may incorporate it into the intervention for the other learners to use.

5.3.2.5 Conclusion

We have seen that the design experiment approach caters for data collection in a range of educational settings, and it affords the development of models of learning processes in specific domains. However, we have also seen that the theoretical foundation of design experiments is not clearly defined. Additionally, some also

argue that they do not place an emphasis on learner agency, however I am unsure about this. For these reasons, I decided to employ a form of design-based research which incorporates aspects of both approaches, and from here on I will use the term design-based research to refer to the version employed in my research.

5.3.3 *Unit of Analysis*

Engeström (2011) had another criticism of the methods that fall under the design-based research umbrella, in that they are not always explicit regarding the unit of analysis. He proposed that the activity system should be the unit of analysis and produced a complex model illustrating how all the different elements interact. Blair (2017) argues that Engeström's model is not a unit of analysis, but a framework to enable solving problems in the workplace. So, while Engeström was correct in calling out the lack of an explicit unit of analysis, I would agree with Blair in arguing that the proposed model is not a suitable solution.

Jones (2008) proposed that the instruction should form the unit of analysis, as it can be seen to initiate and organise activity. For example, an instruction could take the form of a teacher presenting their students with a problem and asking them to solve it. In this case the teacher is initiating and organising the activity with their instruction. The instruction contains the stimulus-end and the stimulus-means, which includes the tools the teacher has provided the students with to aid in the problem-solving process. Following each instruction, the expressions relating to psychological functions at that point in time could be examined to build a picture of the development of different psychological functions.

I would argue that in focusing on the instruction, Jones is not encompassing the way in which each learner will interpret the instruction and process the resulting activity in their own unique way, based on their existing *perezhivaniya*. Veresov (2016) suggested that the *perezhivanie* would be a powerful unit of analysis, as it recognises the complex and organic nature of the development of the human mind, and represents the unity of the personality and the environment. For example, each problem a learner is asked to solve, how they solve it, and how they process the overall experience could be thought of as a *perezhivanie*. I chose the *perezhivanie* as a suitable unit of analysis for my research as it values the unique way each learner develops their sense of a concept when participating in an activity.

It is important to note that it is not possible to obtain an exact picture of an individual's *perezhivaniya* at a given point in time. Similarly, their sense of a concept which is a product of the *perezhivaniya* also cannot be externally perceived in its entirety. However, some aspects of sense can be externally manifested, providing a window through which we can gain an impression of the nature of their sense at that point in time. Therefore, to address my research questions I chose to develop and deliver a series of activities which would result in a series of *perezhivaniya* for each learner. The *perezhivaniya* of each learner could then be examined in turn to identify the external manifestations of sense which occurred in each one.

5.3.4 Reflexivity

There are many approaches within educational research in which the researcher also takes on the role of teacher or other 'insider', commonly known as participant-observers. These approaches can be categorised under the broad term of first-person inquiry (D. L. Ball, 2000), and design-based research is one approach which comes under this umbrella. Rather than looking in at teaching and learning from the outside, first-person inquiry examines it from the inside. This approach offers possibilities for insights and understanding that would not be achievable with traditional controlled experiments.

Traditional controlled experiments in psychology are characterised by their rigid controls. When developing his double stimulation method, Vygotsky found it necessary to let go of this rigid control in order to acknowledge that participants may introduce their own novel stimulus-means into the learning environment (Rene Van der Veer & Valsiner, 1993). In controlled experiments such unexpected variables would be seen as a problem whereas, rather than try to remove them from the equation, design-based research seeks to identify and investigate their role in any learning processes that occur. Thus embracing learner agency is an integral part of design-based research as it enables unexpected variables, including novel stimulus-means, to be included and valued as an important part of an experiment (Engeström, 2011).

The procedures employed within traditional controlled experiments remain fixed throughout the experiment, whereas design-based research works on the principal of

continual refinement (Collins, Joseph, & Bielaczyc, 2004). This is an extremely important part of the process, as it facilitates the development of interventions and theories which are more likely to be successful in real world settings. A detailed history should be maintained of the design of the interventions, the changes implemented and the justifications. This ensures that research audiences are able to judge findings in relation to the complex contexts through which they emerged.

While being an ‘insider’ is important for the first-person inquiry approach, the researcher also runs the risk of becoming so engrossed in the context that they fail to see problems or difficulties with the study (D. L. Ball, 2000). To address this concern, a reflexive approach should be taken, in which the researcher actively acknowledges their own agency in the research process, enabling an increased awareness of the consequences of their decisions throughout the process (S. j. Ball, 1990). As the Design-Based Research Collective (2003) highlighted, researchers will often have to take on both the role of the advocate and the critic, however this is a necessary tension. In design-based research this contradiction can be addressed through the identification of an appropriate unit of analysis, which takes into account the interconnections between the various aspects of the learning ecology. As previously discussed, the unit of analysis for my study is the *perezhivanie*. This brings together all the aspects of the learning environment which make up a particular activity, and how an individual learner processes these aspects. This includes my influence in the form of activity design and approach to instruction. Therefore, through the lens of *perezhivaniya*, I am able to critically evaluate my own influences on the research process.

5.4 Data Collection

5.4.1 Expressing Sense

Before considering data collection methods, I first needed to identify the different ways that visually impaired learners may express their sense of programming concepts. One way of expressing their sense would be through the spoken word, however gestures also seemed to be a rich source, particularly given the physical nature of the programming language that was employed in the intervention. Gesture can be used as an umbrella term to describe a wide variety of different types of hand

movement. There are many different views in the field regarding the nature of gestures, and these views are rarely stated explicitly (Armstrong, Stokoe, & Wilcox, 1995). These views will be explored in this section.

5.4.1.1 Defining Gestures

There is the view among some academics in the field, that gestures are closely related to speech or even an integral part of speaking; from this perspective we can define gestures as hand movements we make when we talk (Kenton, 1980; McNeill, 1992). However, for others, gestures can occur separately from speech. Co-thought gesture is the term used to describe gestures that are produced during silent thought, with co-speech gesture being used as a contrasting term, to describe gestures that are produced during speech (Chu & Kita, 2008).

Gestures could be perceived as simply a method of communication, however they have also been shown to have an influence on thought processes (McNeill, 1992). Embodied cognition theory suggests that external actions have an influence on internal thought processes; gestures, being a form of action, have been seen to have an impact on internal thought processes (Goldin-Meadow, 2014; Goldin-Meadow & Beilock, 2010). Chu & Kita (2011) evaluated the use of gestures during problem solving tasks, finding that gestures increased as the task difficulty increased; this could suggest that gestures can potentially aid the problem-solving process.

It is also important to consider how the different types of gesture may be categorized, for example McNeill (1992) identified four main types of gesture that can be distinguished through analysis:

1. Iconic gestures are characterized by the close relationship between the gesture and the topic of the speech; they describe an element of the scene being described. Together the gesture and speech provide a more complex picture of the speaker's thought processes.
2. Metaphoric gestures represent abstract ideas, rather than a concrete event or object.
3. Beats are simple hand movements, such as a flick of the hand, which are used to indicate significant words or phrases in speech.

4. Deictic gestures are used to indicate objects and events, usually through pointing. Deictic gestures can be abstract, for example by indicating the location of an object that is not physically present.

(Adapted from McNeill, 1992).

It is also possible to group some of these gesture types into larger categories. For instance Cartmill et al. (2012) consider iconic and metaphoric gestures to be a part of a larger category, known as representational gestures. Jelec and Jaworska (2014) expand this category further to also encompass deictic gestures.

For many, gestures do not involve the direct manipulation or the exploration of objects. However it can be argued that gestures can involve physical contact; for example, the use of a finger to draw a path across an object is a gesture that has a meaning for the observer (Streeck, 2009b). The types of gesture outlined by McNeill do not consider how the understanding of an object is obtained through touch; exploratory procedures can be used for this purpose. Exploratory procedures can be thought of as a form of gesture, that are used to explore objects systematically in order to inspect specific properties (Streeck, 2009a). In the following section I will provide an overview of the literature relating to exploratory procedures.

5.4.1.2 Exploratory Procedures

The strategies employed to discover properties of physical objects through touch can be referred to as exploratory procedures, a term coined by Lederman & Klatzky (1987). Lederman & Klatzky identified eight different exploratory procedures that are used to obtain knowledge about 3D objects, as shown in Figure 8. They demonstrated a link between the type of exploratory procedure employed and the knowledge the person wishes to gain from the object. Table 5 outlines the object properties that each of the eight exploratory procedures can be used to inspect.

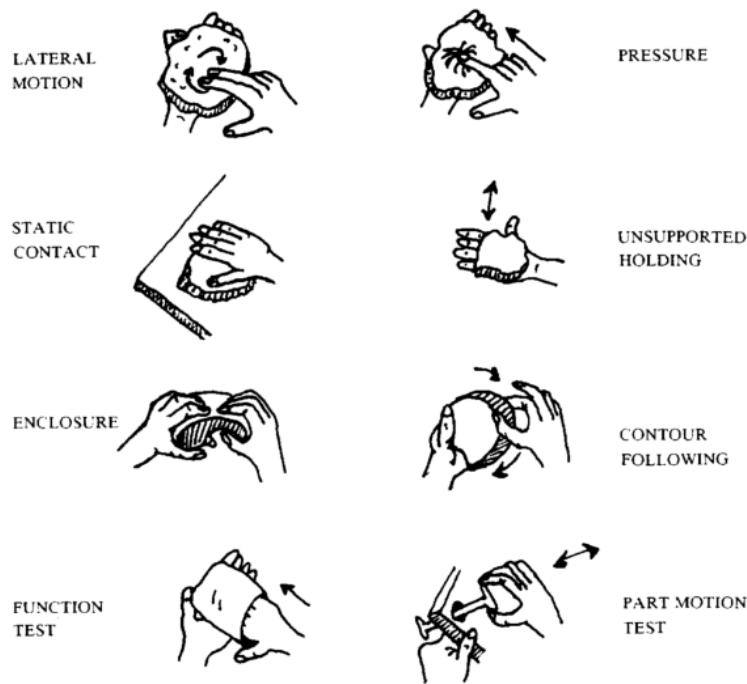


FIGURE 8: EXPLORATORY PROCEDURES AND ASSOCIATED MOVEMENT PATTERNS
(Lederman & Klatzky, 1987)

Although originally developed in 1987, the exploratory procedures identified by Lederman & Klatzky continue to be adopted in contemporary research (Jansen, Bergmann Tiest, & Kappers, 2013; Kalagher & Jones, 2011; Klatzky, Lederman, & Mankinen, 2005). It is important to consider that current exploratory procedure research mainly focuses on single objects that can be held in two hands, whereas physical programming languages usually take the form of larger networks of objects.

TABLE 5: EXPLORATORY PROCEDURES AND ASSOCIATED PROPERTIES (ADAPTED FROM LEDERMAN & KLATZKY, 1987)

Exploratory Procedure	Object Property
Lateral Motion	Texture
Pressure	Hardness
Static Contact	Temperature
Unsupported Holding	Weight
Enclosure	Volume, Global Shape
Contour Following	Exact Shape
Function Test	Specific Function
Part Motion Test	Part Motion

5.4.1.3 Gestures and Visually Impaired Learners

Research has indicated that speakers who have been blind since birth tend to use co-speech gestures, even when they are interacting with other blind individuals, who would not be able to perceive the information contained within the gestures (Iverson & Goldin-Meadow, 2001). This backs up the assertion that gestures are not exclusively used for communication but also support the thought processes of the speaker, aiding them in cognitive tasks (Jelec & Jaworska, 2015). The concept of gestures as a mediating tool for cognitive processes is further supported by research that investigates the use of gestures by visually impaired learners in mathematics. Studies that have explored how blind learners use concrete artefacts and gestures to learn about mathematical concepts suggest that gestures play a key role in the learning process (Healy, Hassan, & Fernandes, 2011; Healy, Ramos, Fernandes, & Peixoto, 2016).

Simulation Gestures

As previously discussed, the iconic, metaphoric and deictic categories of gesture can be described as coming under the representational umbrella. In their study, Jelec and Jaworska (2015) identified an additional type of representational gesture used by blind speakers, which involves acting out a scene or scenario using speech, sound effects and hand movements. This type of gesture was observed when the blind participants were asked to describe an abstract concept. The label simulation gesture has been assigned to this behaviour.

Exploratory Procedures and Tactile Images

Research has also been conducted focusing on the exploratory procedures that visually impaired children use when investigating tactile images. Tactile images are 2D graphics that feature raised lines to enable visually impaired learners to evaluate them through touch. Berila, Butterfield, & Murr (1976) conducted an experiment looking at how accurately visually impaired children can recreate a tactile map. They identified a link between the exploratory procedure employed and accuracy of the recreation, recommending that visually impaired learners be explicitly taught techniques for tangible map exploration. In 1995 Ungar, Blades, & Spencer carried out a similar study, however this time also included sighted children as participants. They found that the recreations of the visually impaired children were less accurate

than those of the sighted children, however the vision of the sighted children was not limited in any way and therefore this does not seem to be a valid comparison. Based on their findings they recommended the need for explicit strategies to be taught.

More recently, Vinter et al. (2012) investigated the exploratory procedures of both sighted and visually impaired children, when working with 2D patterns made out of thin foam attached to card. The participants were only permitted to use touch to explore the patterns and were asked to recreate them through drawing. All participants were familiar with drawing and could choose the approach they felt most comfortable with. It was found that there was no significant difference in the drawing performance of the different groups, however the type of exploratory procedure employed did have a direct impact on the success with which the children were able to recreate the patterns. It was noted that there was a positive correlation between the use of the contour following exploratory procedure and drawing performance, however the contour following exploratory procedure can be slow and cognitively demanding. It is suggested that the efficiency of the contour following exploratory procedure is improved when both hands are employed in a symmetrical pattern.

Vinter et al. built on the exploratory procedures identified by Lederman & Klatzky (1987) by adding local enclosure and the pinch procedure. Local enclosure involves moulding fingers to a specific part of the object in order to obtain precise information about a specific feature. The pinch procedure involves holding an edge between the thumb and a finger; this is also used to gather information about specific local features. An updated list of exploratory procedures that incorporates those identified by Vinter et al. (2012) can be seen in

Table 6.

TABLE 6: UPDATED LIST OF EXPLORATORY PROCEDURES INCORPORATING FINDINGS FROM VINTER ET AL. (2012)

Exploratory Procedure	Object Property
Lateral Motion	Texture
Pressure	Hardness
Static Contact	Temperature
Unsupported Holding	Weight
Pinch Procedure	Local Features
Local Enclosure	Local Features
Global Enclosure	Volume, Global Shape
Contour Following	Exact Shape
Function Test	Specific Function
Part Motion Test	Part Motion

5.4.1.4 Discussion

The literature has demonstrated the varying views regarding what constitutes a gesture (Armstrong, Stokoe, & Wilcox, 1995), with some suggesting that gestures must accompany speech (Kenton, 1980; McNeill, 1992) and others also including gestures that accompany silent thought (Chu & Kita, 2008). Additionally, there is the view that gestures do not involve the manipulation of objects (McNeill, 1992), whereas others place exploratory procedures under the gesture umbrella (Streeck, 2009a). For the purposes of my research gestures are considered as hand movements which aid discovery and depiction. This can include physical contact, as long as the action is not part of the creation or adaptation of a program. Figure 9 illustrates the types of hand movements that I consider as coming under the umbrella term ‘gesture’ in the context of this investigation. Apart from the spoken word, there is one more form of expression that falls outside of the ‘gesture’ umbrella and that is actions. I am considering actions to cover hand or body movements which result in a change in the state of physical tools. For example, connecting parts of the physical programming language together and changing the properties of them would be actions.

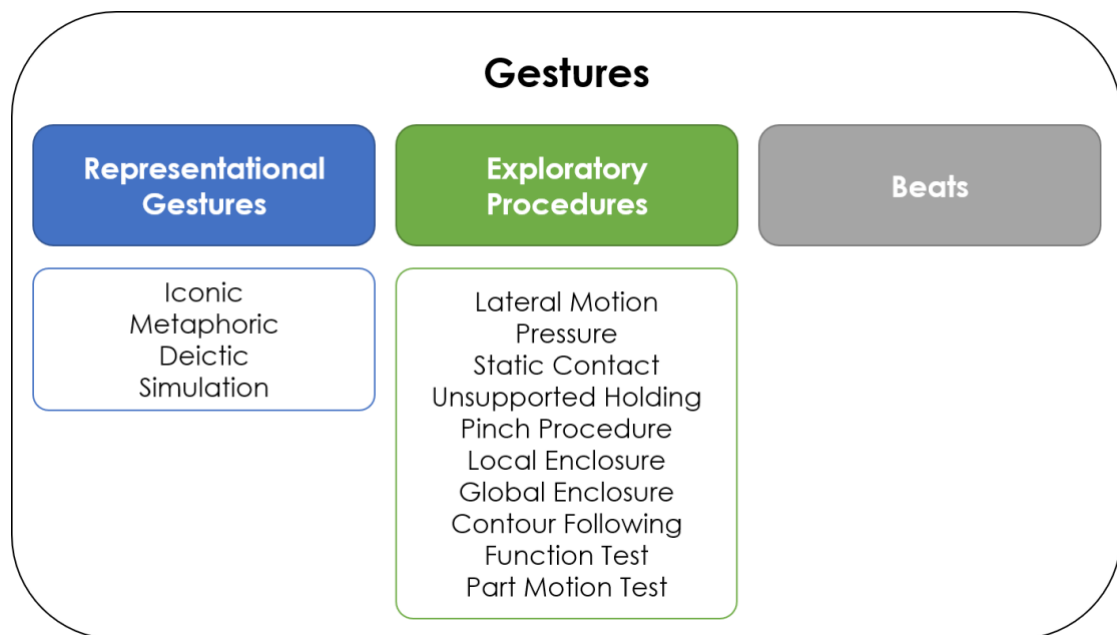


FIGURE 9: ILLUSTRATION OF THE TYPES OF HAND MOVEMENTS THAT COME UNDER THE GESTURE UMBRELLA

5.4.2 Video Recording

In this research, video recording is employed in order to capture and analyse phenomena effectively, which could be used to identify different ways of expressing sense. While observation alone could potentially be used, video recordings capture much more information than a human can process in real time, thus giving a relatively complete record of the interactions that took place (Erickson, 2006). Video records have many benefits, including enabling the researcher to alter the speed of playback, in order to capture subtle interactions that would normally go unnoticed. They can also be viewed multiple times or even at a later date in order to investigate new research questions (Barron, 2007).

Aspects of the “progressive refinement of hypotheses” approach outlined by Engle, Conant & Greeno (2007) are employed during the video recording and analysis. In this approach a general question is framed, and records are collected in an appropriate setting. Once records are collected, an initial theory is formed after some viewing of the records. This theory is then examined in relation to other aspects of the data set, and a more complete explanatory theory is developed; this approach fits in well with the design-based research methodology. Engle et al. (2007) argue that multiple iterations through theory generation and evaluation lead to greater

robustness of findings and increased likelihood that they might be replicated in other contexts.

Powell et al. (2003) put forward a model for the study of the learning process of students in mathematics, which encompasses the principles mentioned above. It features a set of suggested, non-linear stages, which I decided to adapt for the purposes of analysing the video recordings that I collected during my main study.

5.5 Ethical Considerations

Before undertaking any research, it is important to consider its ethical implications. King's College London has a clear ethics approval process which was adhered to throughout the entirety of this research. The first ethical question that I encountered was whether visually impaired children are able to give informed consent. After discussion with the ethics team at the university, it was decided that visual impairments on their own did not hinder the ability of children to give informed consent to take part in research.

Another consideration is the accessibility of the data that was collected. I planned to use video records as the main form of data collection, which are inaccessible to many visually impaired learners. This means that some of the participants would not be able to access the data that is collected about them. Unfortunately, due to the nature of the phenomena under investigation, video recording was the only suitable method of data collection. Although the video records themselves are inaccessible it was ensured that the findings of the research were presented in an accessible form for all participants.

I completed a low-risk application, as the participants were under the age of 16, and therefore parental consent was also required. I created separate information sheets and consent forms for both the participants and their parents, with the language adapted appropriately for each. Ethical approval for the pilot study was granted on February 22nd 2018. The King's College Research Ethics Committee reference number is LRS-17/18-5607 and is included in Appendix 1. Ethical approval for the main study was granted on June 22nd 2018. The King's College Research Ethics Committee reference number is LRS-17/18-7723 and is included in Appendix 6. The

identities of the participants have been protected through the use of pseudonyms and by ensuring that faces were not captured in the video recordings.

5.6 Pilot Study

As previously noted in this chapter, an iterative approach is an important aspect of design-based research, and reflexivity is key to ensuring any findings are robust. Therefore, I conducted the pilot study in order to design and test the planned approach for data collection, which enabled me to reflect on and refine my methods that I would employ in the main study.

5.6.1 Learning Ecology

I have used the term learning ecology, to refer to everything that makes up the learning environment and the interconnections between the different elements. The learning ecology includes both the existing aspects of the environment and those that are introduced through the intervention. However, it is also important to acknowledge that learning ecologies are extremely complex phenomena, and it is impossible to create a complete specification of everything they contain, although we can highlight the elements which are particularly important to the investigation. I have previously mentioned that I am using the term microworld to refer to design of the intervention, including activities, tools and pedagogical approach. The microworld sits within the wider learning ecology that the intervention is situated within. This section will outline the learning ecology of the pilot study.

5.6.1.1 Participants and Site

When selecting a site for design-based research, it is important to locate sites where the problem you wish to investigate exists and are also open to innovation and collaboration (Sandoval, 2013). I recruited the site for my pilot study through running a workshop on programming for visually impaired learners, and invited the attendees to speak to me if they were interested in taking part, as their attendance at the event indicated that they already had an interest in the area. A teacher from a grammar school in outer London that included a unit for the visually impaired was interested in taking part in my research. There were 7 visually impaired learners aged between 11 and 14 at the school with little or no programming experience and there were also 2 students aged between 14

and 16 with some programming experience. An overview of the participants is given in Table 7.

TABLE 7: PILOT STUDY PARTICIPANTS

Name	Level of Vision	Year Group	Approximate Age
Susan	Blind	9	13/14
Ian	Partially sighted	8	12/13
Barbara	Partially sighted	10	14/15
Vicki	Blind	7	11/12
Sean	Partially sighted	9	13/14
Polly	Partially sighted	9	13/14
Ben	Partially sighted	8	12/13
Jamie	Blind	10	14/15
Zoe	Partially sighted	7	11/12

5.6.1.2 *Microworld Design*

Programming Language

As discussed in Chapter 4, most tools that are currently used for introductory programming are not suitable for learners with visual impairments. The tangibility of physical programming languages would seem to make them a good choice, however most are not fully accessible. Therefore, for the purposes of my research, I chose to employ the Code Jumper physical programming language, as it was specifically designed to be inclusive of learners with visual impairments (Morrison et al., 2018). Additionally, the design of the language prioritised learner agency as visually impaired learners were actively involved at every stage of development.

Code Jumper employs pods that can be connected in order to produce sound in the form of music, stories and poems (Thieme et al., 2017). The Code Jumper pods were designed to enable them to be distinguishable through touch as well as through vision, thus making the language inclusive of visually impaired children. Code Jumper can be seen in use in Figure 10.



FIGURE 10: CODE JUMPER IN USE

Figure 11 depicts a simple Code Jumper program. The program starts at the hub, which features a speaker to play the program output, a play and a pause button. There are four ports on the hub and each one can be assigned a different sound set. Each set of pods that are plugged into the hub form an individual thread, and all threads will play at the same time. There is a loop pod connected to the second port and the dial on the top can be used to set the number of repetitions. There is a play pod inside the loop, which can play one sound from a set of up to 8. The sound and speed can be set using the dials on top. There is another play pod that is connected to the exit of the loop; this will play once the loop is complete.

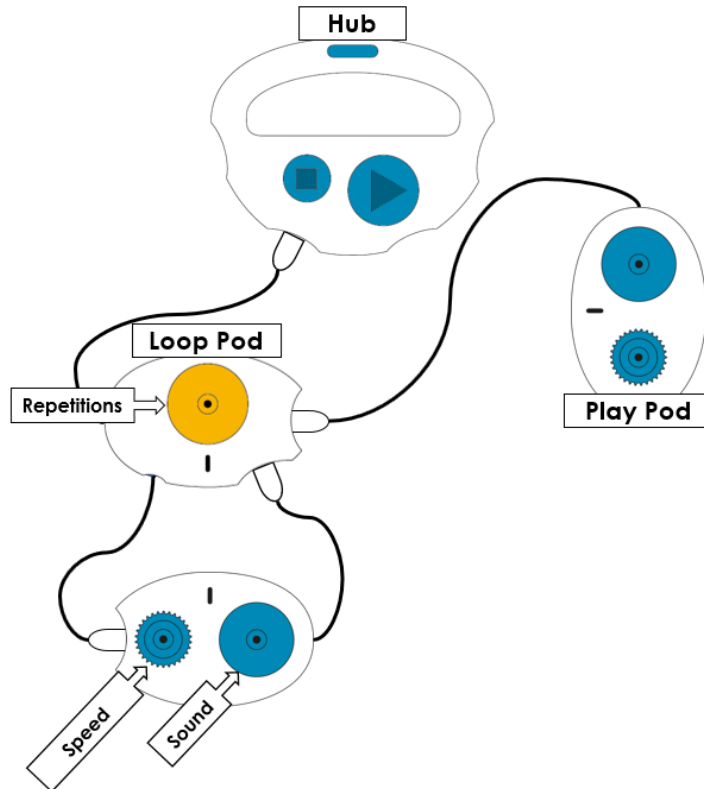


FIGURE 11: DIAGRAM OF A CODE JUMPER PROGRAM

Activity Design

The curriculum that was developed for a study evaluating Code Jumper fed into the design of a lesson plan for the coding workshop (Hadwen-Bennett & Thieme, 2018). As part of the evaluation, teachers were asked to provide feedback on various aspects of Code Jumper, including the curriculum. Feedback from teachers suggests that visually impaired learners may struggle with the purely musical tasks, as they rely on being able to distinguish between different notes (Morrison et al., 2019). Many teachers reported the tasks that feature words and sound effects, such as stories and songs with lyrics, to be more accessible to a wider range of learners.

In the UK learners aged between 7 and 11 need to understand how the control flow of a program is influenced by the three main programming constructs. These are sequence, selection and repetition (Department for Education, 2014). Sequence is the first programming construct to be introduced in the Code Jumper curriculum, with a whole lesson dedicated to it. In the pilot study I wanted to start to explore how learners expressed different programming constructs. Sequence is the simplest programming construct and I felt that if it was taught in isolation the forms of

expression displayed may be limited. Therefore, it was decided to focus on both sequence and repetition in the pilot study lesson.

The sequencing lesson from the Code Jumper curriculum begins with guided exploration of the Code Jumper pods and hub, to enable the learners to familiarize themselves with the hardware. This is followed by the guided creation of a sequence. These aspects of the lesson were retained for the pilot study. One of the later tasks in the original lesson consisted of a challenge that involved the recreation of a musical scale. This activity was removed taking into account the teacher feedback regarding music only tasks. This task was replaced by a word-based activity in which the learner had to recreate a story. However, they were not asked to recreate the program from nothing, they were given a partially complete program which they could explore and then finish. This follows the principle of PRIMM (Sentance et al., 2019), in which the learner is given an existing program to investigate, run and modify. Normally following the modify stage learners are given the task to make a new program from scratch, however this was not incorporated due to time constraints.

The next part of the lesson starts with a guided exploration of the loop pod, inspired by the beginning of the loops lesson from the Code Jumper curriculum. Following this, the learners were guided in the creation of a simple program that used repetition. Next, the learners were given a partially complete program for the song 'Row Your Boat'; this is taken from the loops and sequences lesson in the Code Jumper curriculum and was chosen due to its use of lyrics. As before, the learners were given the task of completing the program. As the ages and abilities of the participants were unknown at the planning stage, it could not be predicted how much material they would get through in one lesson, and to accommodate for this I also included an extension activity. This activity involved recreating the story program they made earlier in the lesson and adding another thread with a sound effect, ensuring it plays at the correct time in the story. An overview of the activities can be seen in Table 8. As designed, each activity featured a problem that the learners needed to solve, and served as the stimulus-end in the method of double stimulation. The tools employed, including Code Jumper and my instruction, served as stimuli-means.

TABLE 8: PILOT STUDY ACTIVITIES

Activity	Teacher Tasks	Student Tasks
1 Code Jumper Intro	Explain that Code Jumper is a programming language. Introduce play pod, pause pod and hub.	Explore hub, play pods and pause pods.
2 Exploration and 1 st Sequence	Lead the creation of sequence consisting of 3 play pods. Once complete explain that this is a sequence, the simplest type of computer program.	Create a program consisting of 3 play pods
3 Story	Plug in partially complete story program. Q: How many play pods are there? Q: What order will they play in? Q: What type of program is it? Support completion of the story program.	Explore the program with their hands, predict what they think it will do when it is played. Play the program to test their predictions. Complete the program based on the example.
4 Loop Pod Intro	Introduce loop pod.	Create a simple program using the loop pod. Play the program, following the pods as it executes.
5 Row Your Boat	Plug in partially complete Row Your Boat program. Q: How many play pods are there? Q: What order will they play in? Q: What type of program is it? Support recreation of the Row Your Boat program.	Explore the program with their hands, predict what they think it will do when it is played. Play the program to test their predictions. Complete the program based on the example.
Extension: Threaded Story	Play example story program with sound effect. Support recreation of the Threaded Story program.	Recreate the two-threaded story program.

5.6.1.3 Procedure

The pilot study took place during one school day, with each workshop lasting for one 50-minute period. In each workshop the students worked in pairs or a group to explore how they expressed their sense of programming constructs when working collaboratively. The groupings were pre-determined by the school based on their

prior knowledge of the students; an overview of the groupings can be seen in Table 9.

TABLE 9: PILOT STUDY GROUPS

Workshop	Participants
1	Susan and Barbara
2	Ian and Vicki
3	Sean, Polly and Ben
4	Jamie and Zoe

Each workshop followed the lesson plan described earlier and the pace was adapted depending on the individual needs of the learners; as a result, some groups got further through the activities than others. Prior to each workshop, the partially complete programs were built and put to one side to enable them to be presented to the students at the appropriate point in the lesson. All workshops were video recorded to enable later analysis. The shot was framed carefully in order to capture the hand movements students generated when working with the physical programming language.

5.6.2 Data Analysis

The aim of the pilot study was to test and refine the data collection approach, therefore the data was not analysed to the depth that was performed in the main study. Each video recording was viewed individually, and notes taken highlighting any potential codes. A second viewing was then conducted to further refine the codes.

Findings

The ways in which the partially sighted learners expressed their sense of programming concepts differed to the expressions of the blind learners. When exploring programs, blind learners tended to use the wires to guide them; this seems to be an example of the contour following exploratory procedure (Lederman & Klatzky, 1987). When the blind learners did not employ contour following, they tended to lose their place in the program. Partially sighted learners, on the other hand

would use vision to explore programs and would sometimes make representational gestures such as pointing.

Some of the partially sighted learners expressed their sense of repetition through a circular gesture. On one occasion I unconsciously made a similar gesture before the student did, however a different student made the same gesture independently.

Another student made a circular gesture, followed by a straight-line gesture when expressing their sense of repetition followed by sequence. During the analysis I did not find it beneficial to employ the subcategories of representational gesture put forward by McNeill, as they are mainly designed for the analysis of gestures that accompany speech, and in my opinion are quite restrictive.

Another observation from the analysis, related to when the students were asked to build a program from scratch based on a sound that they had heard. They seemed to find it challenging to know where to start and to remember what order the instructions should go in.

Reflection

This study provided some interesting insights into how visually impaired learners express their sense of programming concepts, and also demonstrated that this was a suitable approach to take to data collection and analysis. However, these insights needed to be investigated further through the design, delivery, and analysis of a series of sessions, in order to develop a detailed understanding of how sense is expressed and what this could tell us.

As previously mentioned, I found that it was not helpful to divide the gestures I observed into the subcategories outlined by McNeill, or to separate beats from representational gestures. I therefore chose to not to do this going forward and to class them all as representational gestures. I came to a similar conclusion in regard to exploratory procedures, as the nature of physical programming languages may result in the application of novel procedures by learners; I did not want to restrict myself to the procedures that had been previously defined. However, I also recognised that there could potentially be an overlap with existing procedures and did not want to totally exclude them either. As such, I chose to include them in the analysis where appropriate.

The blind learners made more use of gestures during exploration of programs, due to the reliance on touch as a core sense. For this reason, I felt that it would be best to focus more on blind learners for phase two of my research, as their hand movements could potentially provide me with a unique window into the development of their sense of programming concepts.

The challenges the students encountered when building a program from scratch led me to reflect upon the research which discusses the importance of design in programming education. It is suggested that design is an important part of the process of creating a program, particularly for novice programmers (Waite, Curzon, Marsh, Sentance, & Hawden-Bennett, 2018). There is a perception among some computing educators that the design stage should involve some form of formal notation, such as flowcharts or pseudocode. However, Waite et al. suggest that the design could take many potential forms, for example, a storyboard or written description. I decided to introduce a design stage to my activities to facilitate the planning process and to give the learners something to refer to during implementation, rather than having to rely on their memory.

5.7 Main Study

5.7.1 Learning Ecology

5.7.1.1 Participants and Sites

For my main study I was keen to engage with additional sites, however for practical reasons it was also important that the new site was in London. Initially, I found it challenging to find a suitable site, but I was put in touch with a Qualified Teacher of Children and Young People with Vision Impairment (QTVI) that was working with a 15-year-old blind boy in an inner London comprehensive school. The grammar school which participated in the pilot study was also keen to take part again. The participants are summarised in Table 10.

TABLE 10: SUMMARY OF MAIN STUDY PARTICIPANTS

Site	Name	Level of Vision	Year Group	Approximate Age
London Comprehensive	Steven	Blind	11	15/16
Outer London Grammar	Adam	Blind	7	11/12
Outer London Grammar	David	Blind	7	11/12
Outer London Grammar	Gregg	Partially sighted	7	11/12
Outer London Grammar	Sarah	Partially sighted	7	11/12

5.7.1.2 Microworld Design

Based the findings of the pilot study, I took the existing Code Jumper curriculum and adapted it to develop a set of activities, which went through the concepts of sequence, threading, repetition, nested loops, selection and variables. I applied the principles of PRIMM where possible, and added in a design stage. For each new concept that was introduced, the students were provided with a partially complete program which they needed to explore and investigate. They were then provided with a design for the complete program, and asked to modify the program to match the design. Following this the students were provided with a design for a program which they needed to create from scratch. Finally, they would be given a problem which they needed to design a solution for, and implement their design using Code Jumper.

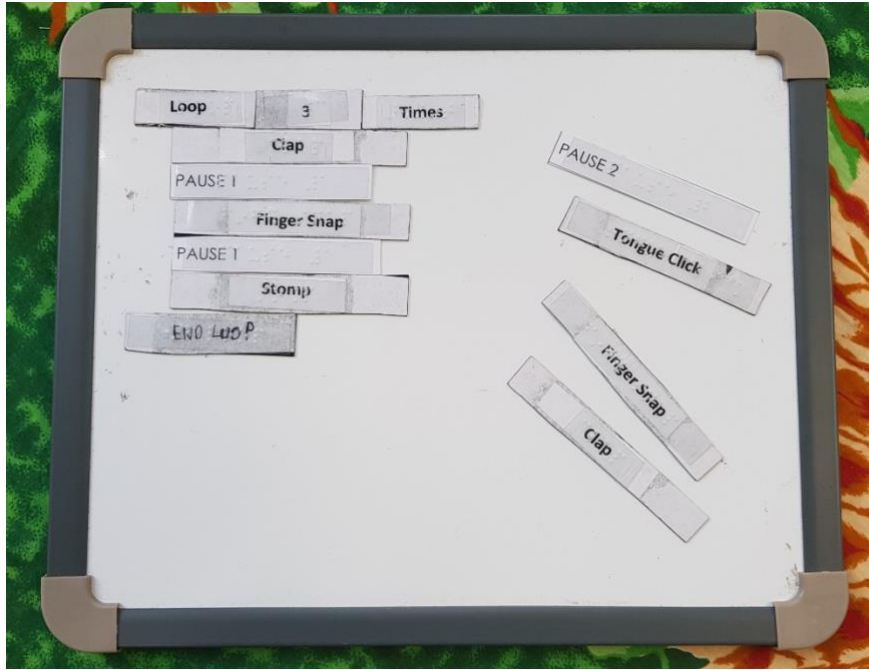


FIGURE 12: DESIGN BOARD

I created a design board as a tool or stimulus-means that would enable the students to create their designs. I prepared a set of magnetic strips, with a braille representation of each instruction, that could be arranged as desired on a magnetic white board. At first, when I was asking the students to create their own designs, I would provide them with the board with all the pieces they needed already on it but jumbled up. Later on, as they became more confident with the design process, I asked them to tell me which pieces they needed. The design board can be seen in Figure 12. For some of the later activities, it was impractical to use the design board as the sounds used were samples from songs, which could not easily be described in words. Table 11 shows a summary of the activities and full details can be found in Appendix 11

TABLE 11: MAIN STUDY ACTIVITIES

	Activity Name	Objective
1	Code Jumper Introduction	Play Pod and Hub Introduction
2	First Sequence	Build own original sequence
3	Limerick 1: There was a young man from Leeds	Complete sequence
4	Limerick 2: There once was a Thingamajig	Turn sequence design into program

	Activity Name	Objective
5	Limerick 3: A funny young fellow named Perkins	Design sequence and create program
6	Threads Introduction	Create original sequences and turn into threaded program
7	Dr Foster	Complete threaded program
8	Threaded Story	Turn threaded design into program
9	Poem: Mashed Potatoes on the Ceiling	Design and build threaded program
10	Loop Pod Introduction	Become familiar with the loop pod in Code Jumper
11	Counting Loop	Complete program that uses repetition
12	Jingle Bells	Turn design with repetition into program
13	Row Your Boat	Design and build program with repetition
14	Frere Jacques	Build program using repetition
15	Original Story	Design and build program for an original story
16	Body Percussion 1	Complete program with loops on different threads
17	Body Percussion 1 Extension	Add an additional loop on another thread
18	Body Percussion 2	Build a program using sequential repetition
19	Body Percussion 3	Design and build a program using sequential repetition
20	Nested Loop Introduction	Develop an understanding of nested loops
21	Nested Loop Percussion 1	Build program that features nested loops
22	Nested Loop Percussion 2	Design and build program that features nested loops.
23	Eye of the Tiger	Create program that features nested loops on multiple threads
24	Gimme Gimme Gimme	Create a program that features a combination of nested and single loops
25	Popcorn	Create program that features nested loops on multiple threads
26	Introduction to Selection	Become familiar with the selection pod in Code Jumper
27	Dynamic Story	Be able to build an original program that uses selection
28	Dynamic Story Extension	Be able to combine selection and repetition

	Activity Name	Objective
29	Introduction to Variables	Be able to use variables in Code Jumper programs
30	Random Music	Be able to combine variables with random values
31	Counter Introduction	Be able to use counters in Code Jumper programs
32	Countdown	Be able to create programs that employ selection, repetition, variables and counters
33	Original story	Building an original program that uses some of the concepts learnt.

5.7.1.3 Procedure

The main study took place over the course of one academic year, with the sessions divided into three blocks. The first block of four sessions took place at the start of the year, the second block of four took place in the middle, and the third block of two sessions took place towards the end. In each workshop for the grammar school site the students worked in pairs or a group. The groupings were pre-determined by the school, based on their prior knowledge of the students; an overview of the groupings can be seen in Table 9. There was only one participant at the other site, so they worked alone. There was an additional participant during the first four sessions that was paired with Sarah. When the other participant withdrew from the study Adam was moved from working with David and Gregg to work with Sarah. The formation of the groups before and after session 4 can be seen in Table 12.

TABLE 12: MAIN STUDY GROUPS

Group	Participants (Sessions 1 to 4)	Participants (Sessions 5 to 10)
1	David, Gregg and Adam	David and Gregg
2	Sarah and withdrawn participant	Sarah and Adam
3	Steven	Steven

Each group worked through the activities at their own pace throughout the ten sessions, so the total number of activities completed varied between the groups. As previously discussed, valuing agency is an important aspect of my methodological approach. As such, when learners introduced their own novel stimulus-means which

seemed to be beneficial I would introduce them to other learners. For example, some learners spontaneously started counting the pods in a sequence, either using a gesture or speaking. Steven initially found working with sequences very challenging and when I introduced counting as a stimulus-means he seemed to find it to be a very helpful tool.

5.7.2 Data Analysis

When analysing video recordings in education research, it is recommended that they are examined progressively at different levels of detail, to ensure findings are as robust as possible (Engle et al., 2007). This process facilitates the development and refinement of theories. The entirety of the data may not be examined at each stage. For example, a subset of the data may be analysed, and an initial theory developed; more of the data could then be examined to test and further refine the theory. For my data analysis, I adapted the method proposed by Powell et al. (2003) which was developed to study learning processes of students in mathematics classrooms. They proposed a non-linear process which includes several stages, not all of which will always be appropriate to every research project. The main stages I adopted were:

1. Describing Video Data
2. Identifying Critical Events
3. Transcribing Critical Events
4. Coding
5. Composing a Narrative
6. Constructing a Storyline

Stage 1 of the process involves writing time coded descriptions of the data in order to map out the content of the video. This process enables the researcher to become more familiar with the content than they would by simply viewing the video (Powell et al., 2003). I started by describing one of Steven's sessions using Inqscribe, a tool designed for creating subtitles for videos. This enabled me to easily insert a timecode with one key press, and describe what was happening in the video at that point in time. As it was not possible to describe every single event second by second, I chose to focus on things that could possibly be thought of as an expression of a sense of a programming concept. This would include sounds, gestures and actions. Additionally, I recorded details of my prompts or questions as they may

have had an influence on the way in which the students expressed their sense of programming concepts. An example timeline for Steven's eighth session can be seen in Appendix 12.

A fellow researcher and I went through the description independently and highlighted what we felt to be significant events; we then discussed and refined this selection. I then transcribed those critical events in greater detail and added context. This was followed by open coding with the other researcher, which involved independently coding the transcript and coming together to discuss and refine the codes. An example coded transcript from Steven's eighth session can be seen in Appendix 13. The process described above was repeated for the video recording of another of Steven's sessions.

At this stage I was getting a feel for what constituted a significant event in the context of my research, and I went through the remainder of Steven's video recordings carrying out stages 1 to 3 simultaneously. After creating the transcript for each video, I once again carried out open coding with another researcher. Once I had completed this stage of analysis for Steven, I repeated the same process for the other groups but this time independently.

The next stage of the process was writing a narrative for each participant, which highlighted how their learning processes developed over the course of the sessions. During this process, I frequently revisited the video recordings in order to add further detail to emerging significant events. I chose to write separate narratives for each individual rather than for each group for two reasons. Firstly, while the students working together shared the same experiences, the way that they processed those experiences is different as they each have a unique *perezhivanie* resulting from the event. Secondly, as the structure of the groups changed during the sessions, it did not seem appropriate to carry out analysis at group level. An extract from Steven's narrative is shown in Appendix 14 and the analysis of his narrative at that stage in the iterative process is shown in Appendix 15.

As Powell et al. (2003) highlight, the process of writing is a form of data analysis itself, enabling the refinement of earlier interpretations, and I certainly found this to be the case. While writing the narratives was an important part of the process, they

were quite extensive and did not make the key points clear enough for the reader. Starting with sequence and threading, I identified the different forms of expression relating to those concepts. These are summarised below:

- **Explore in order** - the learner explores a sequence in order of execution. This could apply both to an unfamiliar program and one which they have created themselves.
- **Explore out of order** - the learner explores a sequence in a different order to the order of execution and may have gotten lost in the program.
- **Contour following** - the learner follows the contours of the program, including the wires, during their exploration.
- **Not contour following** - the learner skips straight to pods without using the wires to guide them.
- **Counting pods** - when following a sequence, the learner may stop briefly, tap or make a gesture on each pod. Additionally, this may be accompanied by the learner counting out loud, or even saying the pod type.
- **Sequence gesture** – when talking about a sequence the learner may make a linear gesture indicating a sequence of instructions.
- **Design sequence in order** – the learner creates the design for a sequence by placing the instructions in a logical order.
- **Design sequence out of order** – the learner finds it challenging to place instructions in a logical order within a program design.
- **Build sequence in order** – the learner adds the pods and sets the sounds in a logical order.
- **Build sequence out of order** – the learner finds it challenging to add pods and set sounds in a logical order.

- **Explain sequence** - the learner may explain the concept of sequence in their own words, identify a sequence in a program or identify the need for a sequence in a program.
- **Create threaded program** – the learner has created programs that employ multiple threads.
- **Identify the need for threading / number of threads** – the learner identifies that a problem will need to be solved using a multi-threaded program and/or the identify the number of threads needed to solve the problem.
- **One event multiple actions** - the way in which the learner talks about working with the play pod indicates that they may see it as performing multiple actions.
- **Lack of confidence** - the learner expresses a lack of confidence in their ability in relation to the sequence or threading activity in question.
- **Confidence** - the learner expresses increasing confidence in their ability in relation to the activity in question.
- **Engagement** - the learner expresses engagement with the task through speech, and other sounds such as laughter or smiling.
- **Success** - the learner expresses a sense of satisfaction and accomplishment upon completing a task.

I then produced a table for each participant, which provides an overview of the storyline of their expressions of sense relating to sequence and threading. This process was repeated for repetition, and finally selection and variables. In constructing these tables I drew upon the transcriptions, narratives and the original video recordings themselves to further refine the models I was developing. The following chapters present the results of this data analysis process.

5.8 Summary

Constructivism is the most appropriate epistemological stance for my research, as I believe that meaning is constructed through our interaction with objects, and this ties in with Vygotsky's views on the creation of meaning. From a methodological standpoint, I needed an approach which would enable me to capture rich data, and facilitate the development of models that captured the way in which a sense of programming concepts is expressed by visually impaired learners. Design-based research fitted this brief. However there are different forms, and I chose to draw on elements of design experiments and formative experiments, in order to ensure that my approach was theoretically grounded in Vygotsky's method of double stimulation.

It has been highlighted that the choice of a unit of analysis is an important factor in carrying out design-based research that produces robust findings. I chose *perezhivanie* as my unit of analysis, as it encompasses the problem, how the learner approaches it, and how they process the overall experience. In particular, it values the unique way each learner develops their sense of a concept when participating.

In regard to data collection, I identified the different ways in which learners may express their sense of programming concepts. I highlighted gestures, which include representational gestures and exploratory procedures; actions including building programs; and sounds including the spoken word. I chose to employ video recording to capture the expressions of learners during the sessions, as they capture much more information than a human can process in real time. Aspects of the "progressive refinement of hypotheses" approach outlined by Engle, Conant, & Greeno (2007) were employed during the video recording and analysis. I also drew upon the method proposed by Powell et al. (2003).

A pilot study was conducted which informed the design of the procedure and learning ecology for the main study. The main study involved five learners across two sites, with ten sessions delivered over the course of one academic year. The analysis process for the main study went through multiple iterations, starting with time-coded descriptions, transcripts of significant events turning into individual narratives. These gradually evolved into tables for each learner which summarise the

different forms of expression they display in relation to different programming concepts.

Chapter 6. **Data Analysis: Sequence and Threading**

6.1 Introduction

In this chapter I will review and analyse data gathered from activities which focused on sequence and threading. The students were introduced to sequence during the first session, and to threading in the second. Activities 2 to 9 focus on either sequence or threading, although these concepts are also present in many later activities. Of these activities, the first four focus on sequence and the remainder on threading. More details of the individual activities can be found on page 95 in Chapter 5. I will examine the ways in which the participants expressed their sense of sequence and threading, and how these expressions evolved over the course of the sessions. I will also explore what these expressions can tell us about the learning process in relation to sequence and threading.

A summary table has been produced for each participant, providing an overview of their expressions of sequence and threading, and highlighting a different set of activities. The activities have been selected to represent the way in which each individual's sense of sequence and threading evolved throughout the sessions. Their journeys will be explored individually, before bringing the themes together in the discussion.

A number of types of expression which relate to sequence and threading were identified, however some were not in evidence for all participants. An overview of each type of expression is provided below:

- **Explore in order** - the learner explores a sequence in order of execution. This could apply both to an unfamiliar program and one which they have created themselves.
- **Explore out of order** - the learner explores a sequence in a different order to the order of execution and may have gotten lost in the program.
- **Contour following** - the learner follows the contours of the program, including the wires, during their exploration.

- **Not contour following** - the learner skips straight to pods without using the wires to guide them.
- **Counting pods** - when following a sequence, the learner may stop briefly, tap or make a gesture on each pod. Additionally, this may be accompanied by the learner counting out loud, or even saying the pod type.
- **Sequence gesture** – when talking about a sequence the learner may make a linear gesture indicating a sequence of instructions.
- **Design sequence in order** – the learner creates the design for a sequence by placing the instructions in a logical order.
- **Design sequence out of order** – the learner finds it challenging to place instructions in a logical order within a program design.
- **Build sequence in order** – the learner adds the pods and sets the sounds in a logical order.
- **Build sequence out of order** – the learner finds it challenging to add pods and set sounds in a logical order.
- **Explain sequence** - the learner may explain the concept of sequence in their own words, identify a sequence in a program, or identify the need for a sequence in a program.
- **Create threaded program** – the learner has created programs that employ multiple threads.
- **Identify the need for threading / number of threads** – the learner identifies that a problem will need to be solved using a multi-threaded program and/or the identify the number of threads needed to solve the problem.
- **One event multiple action** - the way in which the learner talks about working with the play pod indicates that they may see it as performing multiple actions.

- **Lack of confidence** - the learner expresses a lack of confidence in their ability in relation to the sequence or threading activity in question.
- **Confidence** - the learner expresses increasing confidence in their ability in relation to the activity in question.
- **Engagement** - the learner expresses engagement with the task through speech, other sounds such as laughter or smiling.
- **Success** - the learner expresses a sense of satisfaction and accomplishment upon completing a task.

For the learner, each of these types of expression could take the form of speech and other sounds, gestures, including exploratory procedures, and the use of tools involving physical manipulation. These forms of expression have been broadly categorised under the following categories:

- Gestures (including exploratory procedures) ☺
- Tool use (physical manipulation) 🖐
- Verbal/sound (speech, noises, laughter etc.) ◀

Each category is followed by a symbol, which will be used to indicate the category in the summary tables for each participant. The following sections will explore the development of a sense of sequence and threading for each participant in turn.

6.2 Steven's Sense of Sequence and Threading

In Table 13 I have provided a summary of the ways in which Steven's expressions of sequence and threading were manifested throughout the sessions. In the following sub-sections I will explore these expressions in order to uncover the ways in which Steven's sense of these concepts evolved over time.

TABLE 13: STEVEN’S EXPRESSIONS OF SEQUENCE AND THREADING

Steven	2	3	4	5	7	8	9	A1	18	19	26	27
Explore in order	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞
Explore out of order	☞	☞									☞	☞
Contour following	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞
Not contour following	☞	☞									☞	☞
Counting pods		☞◀	☞◀	☞◀	☞		☞	☞◀				
Sequence gesture												
Design sequence in order				☞			☞			☞		
Design sequence out of order												
Build sequence in order	☞	☞	☞	☞	☞	☞	☞		☞	☞	☞	☞
Build sequence out of order	☞	☞	☞		☞							☞
Explain sequence												
Create threaded program					☞	☞	☞					
Identify need for threading / number of threads												
One event, multiple actions		◀				◀						
Lack of confidence	◀	◀☞						◀				
Confidence			◀	☞			◀					
Engagement												
Success				◀	◀	◀	◀					

Note: Some boxes in the table are greyed out to indicate that a particular form of expression is not applicable for that activity. For example, when a concept was yet to be introduced or there was not an opportunity to design the program in that activity.

6.2.1 Exploration and Contour Following

Contour following, an exploratory procedure identified by Lederman and Klatzky (1987), was an important technique that enabled Steven to explore programs featuring sequences. Initially, Steven would try and move his hand straight to the pod he wanted to work with, however this would result in him losing his place in the program. During Activity 2, I suggested to him that he use the wires to help him find his way through programs and he employed this technique in the form of contour following. Figure 13 shows Steven using contour following in Activity 4 in order to locate the end of the program where he needed to add the next pod. I often asked

Steven to follow his completed programs, as it gave me an indication that he understood the order of execution to some extent.



FIGURE 13: STEVEN CONTOUR FOLLOWING IN ACTIVITY 4

Looking at Table 14, it can be seen that at some points during activities 2 and 3, Steven did not contour follow and explored programs out of order. When reminded to follow the contours, he was then able to explore the program in order once again. Throughout most of the remaining sessions, Steven was able to explore his programs in order of execution consistently. He did stop contour following again during Activities 26 and 27. These Activities involved selection, which he had just been introduced to, and initially found challenging. It is possible that the introduction of this new concept had some part to play in his lack of contour following. Nonetheless, during these later activities, he quickly remembered that he needed to follow the contours and he was then able to explore the sequences in order successfully.

6.2.2 Designing and Building Sequences

Steven initially struggled to logically order the instructions within a sequence, often adding a new pod and then setting the sound on the previous pod. Additionally, when he realised that one of the sounds in a sequence was set incorrectly, he found it challenging to locate the correct pod to change. These observations would tie in with

the literature that suggests that novice programmers often have difficulties in understanding the order in which statements are executed (Swidan et al., 2018). During Activity 3, I suggested to Steven that he could count the pods and the instructions in his design, in order to locate the correct ones to change. He started to employ this technique and it seemed to be very successful. By the time Steven came to create his own design for a program he seemed to be more confident with sequences. He first designed an original sequence in Activity 5, and he was able to do so with relative ease. He then turned his design into a program using Code Jumper, and seemed to find it helpful to have the design to refer back throughout the building process. He also designed sequences on two further occasions without any difficulty.



FIGURE 14: STEVEN COUNTING PODS IN ACTIVITY 4

Counting for Steven was usually expressed through gesture and voice. When exploring a program, he would briefly pause on each pod as he counted them out loud. In Figure 14, Steven can be seen counting the pods in his partially complete program for Activity 4. After Activity 5, Steven no longer counted out loud, but would still occasionally make a counting gesture by pausing briefly on each pod, and at some points external expressions of counting ceased altogether. During this period of reduced external expressions of counting, Steven also designed and built sequences in order more consistently. Additionally, he expressed more confidence and satisfaction in completed programs. Therefore, it could be argued that as Steven's confidence with creating sequences grew, his external expressions of

counting reduced. This does not imply that he stopped counting all together, it just means that the process was not externally perceivable.

6.2.3 Sequence Assessment Activity

The sequence assessment activity (A1) was designed as a formative assessment to evaluate understanding of sequence. This activity utilised three example programs consisting of miniature pods stuck to a Velcro board. The use of 3D printed representations of the pods enabled me to prepare multiple example programs in advance, something that is not possible with Code Jumper due to the number of pods available in a set. I also prepared a recording of a program featuring four sounds with short pauses between them. Figure 15 shows illustrations of the three example programs. Both Program A and Program B could have created the output captured in my recording, depending on whether you consider the pause to be a part of the sound sample or not. Whereas, Program C could not have created the output in the recording as it only has three play pods and there are four sounds in the recording.

When I introduced Steven to the miniature versions of the pods, he was initially confused as they were smaller, and the texture was different. This confusion seemed to result in him expressing a lack of confidence once again, and also to employing externally perceivable counting techniques such as gestures and use of voice. Despite this, Steven was able to explore the sequences in order of execution and worked out that both Program A and Program B could have created the sound he heard in the recording.

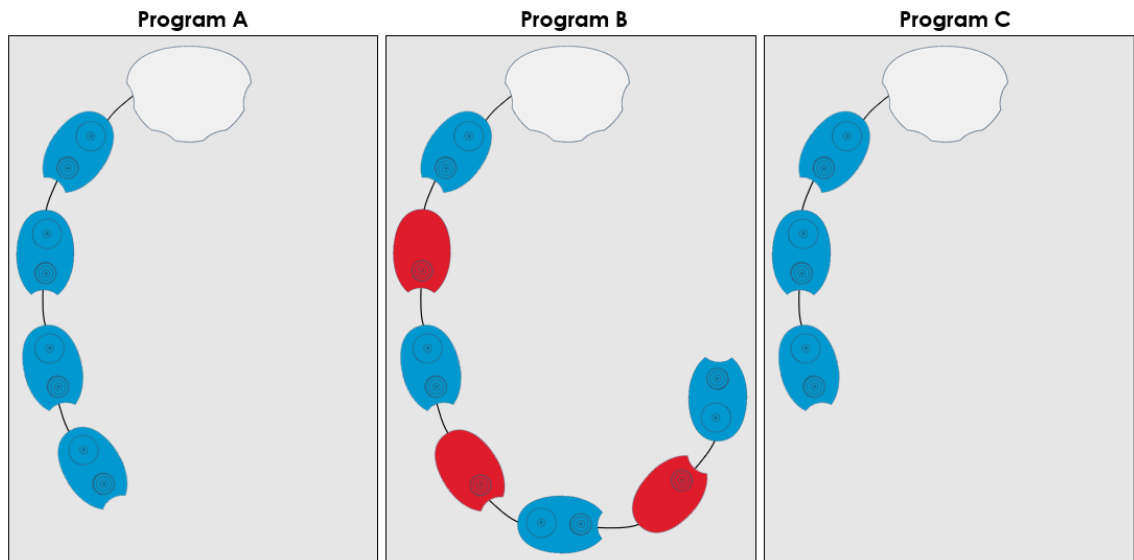


FIGURE 15: SEQUENCE ASSESSMENT ACTIVITY PROGRAMS

6.2.4 Threading

Steven was introduced to threading in Activity 6 and was initially confused by the sounds playing simultaneously; however after I explained that threading allows us to play multiple sounds at the same time, he was quickly able to apply the principle to other tasks. Steven expressed his sense of threading through the design and building of multi-threaded programs. This can be seen in Activities 7 to 9 in Table 14. The remainder of the threading row is greyed out, as the other activities highlighted did not involve multiple threads.

6.2.5 One Event, Multiple Actions

Early on, when designing programs, Steven would sometimes place instructions next to each other rather than underneath, seemingly making them part of the same instruction. This would usually be when two instructions completed a sentence and it seemed logical to place them together. However, on a couple of occasions when I asked Steven what he needed to set the next play pod to, he would say two commands rather than just one. It is almost as if sometimes Steven was viewing the play pod as representing multiple actions.

6.2.6 Summary

Throughout his sessions, Steven’s use of tools, voice, and gestures provided a window into the development of his sense of sequence and threading. The

development of his sense of sequence is clearly demonstrated through his initial difficulty exploring programs in order of execution, and struggling to build sequences in order. He successfully employed the contour following exploratory procedure and was able to confidently build sequences. Additionally, he employed counting, using both gesture and voice, as a tool when building sequences. External manifestations of counting seemed to reduce as his confidence with sequence and threading grew, and they increased again when he encountered concepts which he found challenging.

6.3 Adam's Sense of Sequence and Threading

Table 14 provides a summary of the ways in which Adam's expressions of sequence and threading were manifested throughout the sessions. In the following subsections I will explore these expressions to uncover the ways in which Adam's sense of these concepts evolved over time.

TABLE 14: ADAM'S EXPRESSIONS OF SEQUENCE AND THREADING

Adam	3	4	5	6	7	8	13	A1	19	27	28
Explore in order	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞
Explore out of order	☞			☞	☞	☞	☞		☞		
Contour following	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞
Not contour following	☞				☞	☞	☞		☞		
Counting pods					☞			☞◀			
Sequence gesture											
Design sequence in order							☞		☞		
Design sequence out of order											
Build sequence in order	☞	☞	☞	☞	☞	☞	☞		☞	☞	
Build sequence out of order		☞		☞	☞		☞			☞	
Explain sequence											
Create threaded program				☞	☞	☞					
Identify need for threading / number of threads											
One event, multiple actions									☞		
Lack of confidence	☞	☞			☞		☞				
Confidence				☞		☞					
Engagement	◀	☞		◀	◀		◀				
Success		☞◀	☞		☞	☞					

6.3.1 Exploration and Contour Following

In the third activity, Adam initially struggled to explore sequences in order of execution, and I suggested that he should use the wires to guide him. This led him to employ the contour following exploratory procedure. Following this, in Activities 4 and 5, Adam successfully employed contour following to explore programs.

Threading was introduced during Activities 6 to 8, and this seemed to coincide with a recurrence of following out of order, which sometimes was related to a lack of contour following. Throughout the remaining sessions, Adam would sometimes initially forget to contour follow, seemingly resulting in him exploring out of order. He would either be reminded to contour follow, or remember himself and be able to explore in order of execution successfully. For example, in Activity 7 Adam needed to add more pods to the end of a sequence, and he initially got lost, as he was not contour following. After being reminded, he was able to follow the sequence to locate where he needed to add the next pod. In Figure 16, Adam is in the process of contour following, holding the pod he is going to add in the other hand. Towards the end of the sessions, Adam's use of contour following became more consistent.



FIGURE 16: ADAM CONTOUR FOLLOWING IN ACTIVITY 7

6.3.2 Designing and Building Sequences

Adam demonstrated the ability to build sequences in a logical order early on in the sessions. At the start of a number of activities, he would add a new pod to a sequence and then go to set the sound on the previous pod, in the same manner that Steven did. Although this behaviour continued throughout the sessions, over time he seemed to find it easier to overcome. The occurrences of creating sequences out of order often coincided with the introduction of a new concept or tool. For example, when threading was introduced during Activities 6 and 7, and when variables were introduced in Activity 27. Adam did not have the opportunity to create designs for

programs earlier on in the sessions, however when he came to create designs later on, he was able to do so successfully.

During the majority of the activities, Adam did not utilise counting when exploring or creating sequences. However, he did spontaneously use a counting gesture during Activity 7, when he had just been introduced to threading. Additionally, he employed gesture and voice when exploring the example programs provided to him in the Sequence Assessment Activity (A1).

6.3.3 Sequence Assessment Activity

When Adam was introduced to the three example programs in the assessment activity (A1), he explored them in order utilising contour following and counting. His use of counting involved the use of gesture, in the form of tapping each pod and counting or naming the pods out loud. He identified that the recording of the example program featured four sounds and he also felt that there were pauses in between them. For this reason, he chose Program B (see Figure 3) as the most likely candidate to have created the sound of the recording he heard.

6.3.4 Threading

Like Steven, Adam was introduced to threading in Activity 6 and was quickly able to build multi-threaded programs. Table 15 shows that Adam expressed his sense of threading through designing and building programs containing multiple threads in activities 6 to 8. Unlike Steven, Adam did not seem surprised by all the sounds playing at the same time.

6.3.5 One Event, Multiple Actions

At one point, during Activity 19, Adam placed instructions next to each other rather than underneath when designing the program. It seemed as if he was viewing the two instructions as one. This behaviour is similar to that of Steven when designing programs, however for Adam this only occurred once. Additionally, whereas Steven would sometimes name two commands when setting a single play pod, Adam did not do this. Based on this data it is not possible to conclude whether or Adam was viewing the play instruction as representing multiple actions.

6.3.6 Summary

Adam's sense of sequence and threading is clearly demonstrated throughout the sessions through his use of tools, gestures and to some extent his voice. However, there is not such a pronounced evolution as was the case for Steven. There was some change however, particularly in his use of the contour following exploratory procedure. Adam initially struggled to follow sequences in order of execution, although this quickly changed when I suggested he use the wires to guide him. After this, there would still be occasions when he did not follow a sequence in order but upon being reminded of contour following, or remembering himself, he was able to explore in order of execution successfully. On a number of occasions, Adam built parts of sequences out of order but was able to correct them. The occurrences of Adam exploring or building programs out of order seemed to coincide with the introduction of a new concept, and as time went on, he explored and built in order more consistently. Adam only occasionally employed counting, and when he did, he had been introduced to a new concept or tool. On a number of occasions when working with sequences and threading, Adam expressed both a lack of confidence, engagement and success. As time went on, Adam started to display a sense of confidence which also helps to build a picture of how his sense was developing.

6.4 David's Sense of Sequence and Threading

Table 15 gives an overview of the ways in which David's expressions of sequence and threading were manifested throughout the sessions. In the following subsections I will explore these expressions in order to uncover the ways in which David's sense of these concepts evolved over time.

TABLE 15: DAVID'S EXPRESSIONS OF SEQUENCE AND THREADING

David	3	4	5	6	7	8	13	15	A1	32	33
Explore in order	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
Explore out of order	☺		☺	☺		☺	☺				
Contour following	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
Not contour following				☺		☺	☺				
Counting pods	☺◀				☺◀				☺◀		
Sequence gesture											
Design sequence in order			☺				☺	☺			
Design sequence out of order											
Build sequence in order	☺	☺	☺	☺		☺	☺	☺		☺	☺
Build sequence out of order	☺	☺	☺			☺					
Explain sequence				▶						▶	
Create threaded program				☺		☺		☺			☺
Identify need for threading / number of threads				▶	▶						▶
One event, multiple actions											
Lack of confidence											
Confidence			▶	▶	☺						
Engagement	▶		▶	▶	▶						
Success	☺	▶☺	▶								

6.4.1 Exploration and Contour Following

From the first activities, David employed contour following when exploring programs, however there were a few occasions when he forgot to use this exploratory procedure. From looking at Table 16 it can be seen that during Activities 6, 8 and 13, David did not follow the contours of the program at one stage, which

resulted in him exploring it out of the order of execution. However, on these occasions, David either quickly remembered, or was reminded, and was then able to explore the program in order successfully. As time went on, David's successful use of the contour following exploratory procedure became more consistent. Two out of the three occurrences of not contour following shown in Table 16, coincided with the introduction of a new concept or topic. For instance, in Activity 6 threading had just been introduced and in Activity 13 loops and sequences had been brought together for the first time.

6.4.2 Designing and Building Sequences

Early on, there were a few occasions when David built sequences out of order. However he was quickly able to overcome this and was able to create sequences in order consistently. For Adam there seemed to be a potential correlation between the occurrences of building out of order and the introduction of a new concept; this does not seem to have been the case for David. David also demonstrated the ability to successfully design sequences on a number of occasions.

Like both Adam and Steven, David employed external manifestations of counting when exploring programs at certain stages during the activities. The ways in which David expressed counting included the use of gestures and voice. At different points, David counted the pods out loud and also said what each pod should say as he gestured to it. One occasion that he used counting was in Activity 7, in which the students were given a partially complete program with two threads that recreated part of the poem 'Dr Foster' with sound effects. As can be seen in Figure 5, David explored the two threads of the program simultaneously while also counting out loud. At this point threading had only just been introduced and was a new concept for David. He also employed counting when he first encountered the mini representations of the Code Jumper pods in sequence assessment activity. This could suggest that, for David, there is a potential relationship between the external manifestation of counting and encountering new concepts or tools.



FIGURE 17: DAVID COUNTING THE PODS IN ACTIVITY 7

6.4.3 Sequence Assessment Activity

When David was first introduced to a program created using the mini representations of Code Jumper pods, he started in the middle and explored outwards until he located the hub. I do not consider this as exploration out of order, as he was simply orientating himself within the program. Once David had located the hub, he kept one hand on it as he used the other hand to explore the sequence in order, stating the type of each pod as he went, as Figure 17 illustrates. David was able to identify that there were four sounds in the recorded program and that Program A, shown in Figure 18, could have created the sound they heard. When asked why, David replied, “because there are four play pods” and he proceeded to count the pods in the program with a gesture and saying the number of each pod. He also identified that Program B could have also produced the sound.

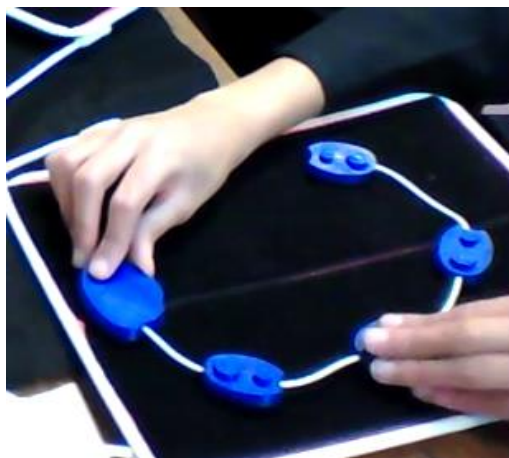


FIGURE 18: DAVID EXPLORING PROGRAM C IN THE SEQUENCE ASSESSMENT ACTIVITY

6.4.4 Threading

In a similar manner to Steven and Adam, David seemed to quickly grasp working with threads and was able to build multi-threaded programs confidently. He demonstrated this confidence both in the way he discussed solving problems and in the manner he built the programs, with purposeful actions and little hesitation. Table 16 demonstrates that David expressed his sense of threading through creation of multi-threaded programs, in the same way that both Steven and Adam did. However, David also expressed his sense of threading through external speech, by either identifying the need for threading to solve a problem, or identifying the total number of threads required. For example, in Activity 33 the students were required to use the sound effects sound set to come up with an original story. When building the program, David decided that he wanted the sound of a dog barking to play throughout the story, and identified the need to put it on a different thread to the main story.

6.4.5 Summary

The ways in which David expressed his sense of sequence and threading throughout the activities demonstrates how his sense evolved over time. He had some initial problems with building programs in a logical order, however these were quickly overcome and by the end he was creating sequences and multi-threaded programs confidently. David also employed the contour following exploratory procedure throughout the activities, only occasionally forgetting to when a new concept was introduced. When David did forget to follow the contours of the program, he would usually explore it out of order. Sometimes, David would also utilise counting using gesture and voice to explore programs. The use of counting also seemed to coincide the introduction of a new concept or a new tool.

6.5 Sarah's Sense of Sequence and Threading

In Table 16 I have produced a summary of the ways in which Sarah's expressions of sequence and threading were manifested throughout the sessions. In the following sub-sections I will explore these expressions in order to uncover the ways in which her sense of these concepts evolved over time.

TABLE 16: SARAH'S EXPRESSIONS OF SEQUENCE AND THREADING

Sarah	2	3	4	5	6	7	8	9	13	15	17	A1	27	31
Explore in order	☞	☞	☞	☞		☞	☞	☞	☞	☞		☞		
Explore out of order												☞		
Contour following			☞	☞		☞	☞	☞	☞	☞				
Not contour following	☞	☞		☞								☞		
Counting pods		☞										☞◀		
Sequence gesture						☞								☞
Design sequence in order				☞				☞						
Design sequence out of order									☞					
Build sequence in order	☞	☞	☞	☞	☞	☞	☞	☞	☞				☞	
Build sequence out of order						☞			☞				☞	
Explain sequence								◀						◀
Create threaded program					☞	☞	☞	☞		☞	☞			
Identify need for threading / number of threads														
One event, multiple actions			☞											
Lack of confidence						☞								
Confidence		☞	☞			☞	☞							
Engagement														
Success														

6.5.1 Exploration and Contour Following

Looking at Table 17, it can be seen that Sarah explored sequences in order of execution throughout most of the activities. She also did not always employ the contour following exploratory procedure, and this did not seem to impact on the order in which she explored programs. This is likely due to the fact that Sarah has

some limited vision, which enabled her to see rough shapes and colours, therefore she was able to identify the relative position of different pods using their colours. The participants we have discussed thus far have used gestures as their primary method of exploring the structure of programs, whereas Sarah employed a combination of gesture and vision. As the data collection method employed only enables us to examine the use of gesture and not the use of sight, the data may not provide as much of an insight into her development of a sense of sequence as it did for other participants. However, it can be seen that Sarah initially did not employ the contour following exploratory procedure and then quickly adopted it. Although its use does not seem to have affected her ability to explore programs in order, it may be that the additional information provided to her through the use of this method aided the development of her sense of sequence.

On a couple of occasions, Sarah produced a gesture that seemed to represent sequence as a concept, something the other participants did not do. The first time she produced the gesture was during Activity 7 when she was working with a multi-threaded program. She indicated the sequence that the next sound needed to be added to by making a linear gesture above, but following the line of the sequence, as shown in Figure 19.

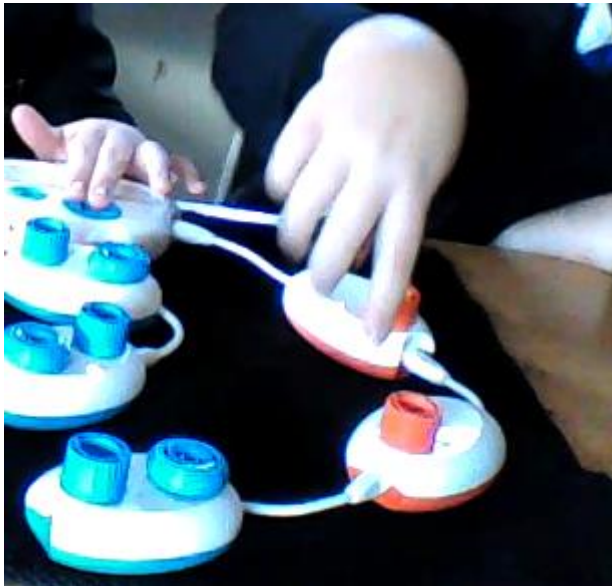


FIGURE 19: SARAH PRODUCING A SEQUENCE GESTURE IN ACTIVITY 7

6.5.2 Designing and Building Sequences

Throughout most of the activities, Sarah was able to confidently design and build sequences in a logical order. For Sarah, her confidence was expressed in the manner she constructed sequences, with purposeful actions and little hesitation. In some of the later activities, there were a few points when she initially started to create sequence in a non-logical order. As was the case with Adam, there seems to be some correlation between the introduction of a new concept and occurrences of not creating programs in a logical order. For example, in Activity 7 threading had just been introduced and Sarah initially found creating the program in a logical order challenging.

On a couple of occasions Sarah did display external manifestations of counting when exploring programs. On the first occasion this took the form of a gesture, and on the second she also named the sound that each pod would make. Although there were only two apparent occurrences of counting, Sarah may have utilised her limited vision to count pods on other occasions. The first external manifestation of counting occurred during Activity 3, when the concept of sequence had just been introduced to Sarah. The second time occurred during the sequence assessment activity, when Sarah was first introduced to the mini representations of Code Jumper pods. Therefore, once again, it is possible that there is some correlation between external manifestations of counting and the introduction of new concepts or tools.

6.5.3 Sequence Assessment Activity

When I introduced Sarah to the first program, she explored it in reverse using contour following, seemingly orientating herself within the program. Sarah then explored the second program, exploring the sequence in order, tapping each pod and saying the sound she thought it would make. She was able to identify that both Program A and Program B, as shown in Figure 3, could have created the sound of the recorded program. Sarah also pointed out that Program C could not have created the sound, as it only had three play pods and there were four separate sounds in the recorded program.

6.5.4 Threading

Like the other participants, Sarah expressed her sense of threading through the design and building of multi-threaded programs. She was quickly able to confidently and consistently construct multi-threaded programs, seeming sure about what actions she needed to perform to achieve her desired outcome. Sarah seemed to grasp the concept of threading relatively quickly.

6.5.5 One Event, Multiple Actions

As previously mentioned, there is some indication that both Steven and Adam viewed the play pod as representing multiple actions at certain points throughout the activities. At one stage during Activity 4, Sarah was following a program while it was playing, and she seemed to view one of the play pods as producing two sounds, because she stayed on that pod when both sounds played. She then repeated this behaviour when following the program for a second time. Although this only occurred in one activity, it is possible that she thought of that play pod as representing multiple actions.

6.5.6 Summary

Sarah's limited vision meant that she did not have to rely solely on the use of gestures to explore programs, in the way that other participants did. She did, however, produce a gesture which seemed to represent a sequence on two occasions, something none of the other participants did. Like other participants, Sarah did display external manifestations of counting on a couple of occasions, and these did seem to coincide with the introduction of a new concept or tool, as was also the case with some of the other participants.

Sarah quickly developed the capacity to create sequences and multi-threaded programs in a logical order, and on many occasions during these activities she seemed confident in her ability to build sequences. There were a few occasions where Sarah started to create sequences in an order that did not seem logical, and these occasions seemed to coincide with introduction of new concepts, or bringing concepts together for the first time. Like David, Sarah demonstrated the ability to explain the concept of sequence verbally, unlike Steven and Adam.

6.6 Gregg's Sense of Sequence and Threading

In Table 17 I have provided a summary of the ways in which Gregg's expressions of sequence and threading were manifested throughout the sessions. In the following sub-sections I will explore these expressions, in order to uncover the ways in which his sense of these concepts evolved over time.

TABLE 17: GREGG'S EXPRESSIONS OF SEQUENCE AND THREADING

Gregg	3	4	5	6	7	8	13	A1	18	28	32	33
Explore in order			☺		☺		☺	☺			☺	
Explore out of order												
Contour following												
Not contour following			☺		☺		☺	☺			☺	
Counting pods					◀☺		◀☺	◀☺				
Sequence gesture												
Design sequence in order												
Design sequence out of order							☺					
Build sequence in order	☺			☺	☺		☺		☺	☺	☺	☺
Build sequence out of order					☺		☺			☺		
Explain sequence				◀								
Create threaded program				☺	☺	☺						
Identify need for threading / number of threads												
One event, multiple actions												
Lack of confidence												
Confidence			☺		☺							
Engagement	◀		◀	◀	◀	◀						
Success	◀	◀			☺							

6.6.1 Exploration and Contour Following

Like Sarah, Gregg also has some limited vision, which enabled him to familiarise himself with programs without relying solely on gestures. As can be seen in Table 18, Gregg did explore sequences in order of execution consistently throughout the activities. He did not need to use the contour following exploratory procedure to

achieve this, instead he skipped directly between the pods, sometimes barely touching them.

6.6.2 Designing and Building Sequences

Gregg was quickly able to build sequences in a logical order and did this consistently throughout the activities with two exceptions. One example was in Activity 28, when Gregg was required to create sequences to use within a program which brought together repetition and selection, he initially started to build the sequence in a non-logical order. As we have seen before, for some participants the introduction of new concepts, or the bringing together of concepts for the first time, seems to correlate with occurrences of building programs out of a logical order. Gregg was in a group with Adam and David when he was working with sequences, and due to them taking turns working on different parts of the activities, Gregg only had one opportunity to design a sequence and on that occasion he found it challenging to design in a logical order.

Gregg also expressed external manifestations of counting on a few occasions. In Activity 7, when he had just been introduced to threading, Gregg explored the program by touching each pod one at a time and stating the type of each pod. Activity 13 involved bringing together sequence and repetition, and Gregg initially found it challenging to build the sequence in a logical order. He utilised counting to locate the pod in the sequence that needed to be changed, displaying both a gesture and counting out loud. This is illustrated in Figure 20, in which Gregg is pointing at each pod as he counts it. The other occasion that Gregg employed counting explicitly was during the sequence assessment activity, when he also gestured to each pod and counted them out loud. As with other participants, there does seem to be some correlation between external manifestations of counting and the introduction of a new concept or tool and bringing concepts together.

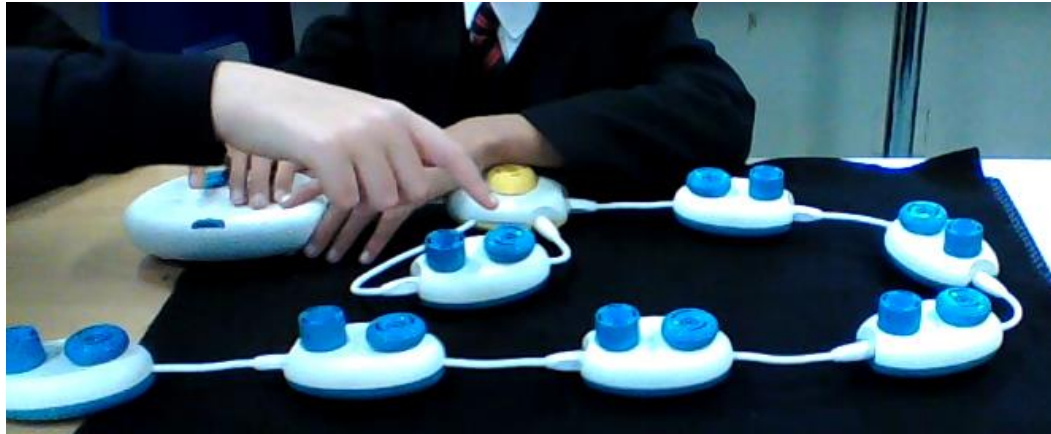


FIGURE 20: GREGG COUNTING PODS IN ACTIVITY 13

6.6.3 Sequence Assessment Activity

In this activity, Gregg was able to explore the example sequences in order of execution. He identified that there were four sounds in the recorded program and therefore the program that created it would have to have four play pods. Gregg identified Program A (shown in Figure 15) as the answer, and after a short discussion, he also realised that Program B could also have created the sound he heard in the recording.

6.6.4 Threading

After Gregg was introduced to the concept of threading in Activity 6, he quickly demonstrated the ability to design and build programs consisting of multiple threads without apparent difficulty. Like most of the other participants, Gregg only expressed his sense of threading through the creation of programs which involved the concept.

6.6.5 Summary

As Gregg has some limited vision, he was able to explore programs successfully without needing to employ the contour following exploratory procedure. He demonstrated his sense of sequence through his exploration of programs in order of execution and through building sequences in a logical order. Additionally, he also expressed success on many occasions and an increase in confidence over time. On one occasion Gregg also expressed his understanding of sequence in words. Gregg expressed his sense of threading through the successful creation of multi-threaded programs.

Gregg exhibited external manifestations of counting on a few occasions, and these seemed to coincide with either the introduction of a new concepts, bringing concepts together, or the introduction of a new tool. There were also two occasions when Gregg initially found it challenging to build a sequence in a logical order, and both of these occurred during activities which brought multiple concepts together.

6.7 Discussion

Perezhivanie, as discussed in Chapter 2, is the term Vygotsky used to encapsulate an experience, the processing of it and the assimilation into the personality (Blunden, 2016). In this analysis of the expressions of sequence and threading, each activity for each participant is treated as a *perezhivanie*. All the activities involving sequence and threading for a participant make up the *perezhivaniya*, which shape their individual sense of these concepts. I have examined the development of a sense of sequence and threading for each individual through the window provided by external speech, private speech (Berk, 1992) and tool use (Miller, 2011).

Although the participants completed the same activities, their *perezhivaniya* will be different, as the way in which they process their experiences are unique. Even so, there are some themes that have emerged through the analysis of each individual's development of a sense of sequence and threading. I will now explore the similarities and differences between the expression of a sense of these concepts among the participants.

6.7.1 A Sense of Sequence and Threading

All five participants expressed their sense of sequence and threading in a way which demonstrated that they understood these concepts, and could apply them to solve problems. Each of these concepts have culturally defined meanings which may differ from an individual's sense (Kravtsova, 2017). In this case the application of these concepts to computing problems by the participants, indicates that their understanding shares a strong relationship to the culturally defined meanings, even though at first glance some of their expressions may not be recognised by computer scientists.

All of the participants demonstrated the ability to design and build programs featuring sequence and threading. Some of the participants initially seemed to find building logical sequences of instructions challenging, but they did overcome this barrier in time, and this would tie in with Swidan's (2018) finding which indicated that learners may initially find it difficult to understand the order in which instructions are executed. It is possible that the physicality of the tools aided the participants in overcoming this challenge. On the other hand, none the participants seemed to have any difficulty getting to grips with the concept of threading. Meerbaum-Salant et al. (2011) did find that concurrency was a very challenging concept for younger students, however they employed Scratch as a tool which enables learners to create countless threads scattered throughout projects. In contrast, Code Jumper facilitates threading through a physical layout of threads, that are always right next to each other. It is possible that it is not the concept itself that is challenging, rather the tool which is employed, and this data provides evidence that physicality is likely to be an important factor in the development of a sense of threading.

Another form of expression, which all learners exhibited, is the exploration of a sequence in order of execution, and it is important to note that the contour following exploratory procedure (Lederman & Klatzky, 1987) seemed to play an important role here. Steven, Adam and David all relied solely on gestures to explore programs and when they neglected to employ contour following, they tended to explore sequences out of order. Sarah and Gregg, on the other hand, have some limited vision, which meant that contour following was not as important to them. Sarah initially did not use contour following but later adopted the technique. When she started to use contour following, Sarah was working with another student with some limited vision who was not part of this research. Gregg did not employ contour following at all and was able to consistently explore programs in order of execution.

Some of the participants also expressed their sense in other ways. For example, Gregg, Sarah and David were able to explain the concept of a sequence verbally, and Sarah made a sequence gesture on a couple of occasions. In regard to threading, David was the only participant who explained the concept of threading through the use of voice.

6.7.2 Affect

As noted in Chapter 2, consciousness comprises two key elements: intellect and affect (Wertsch, 1985). Vygotsky believed that both elements play an important role in learning process. Therefore affect should be taken into consideration when analysing *perezhivaniya* (Mahn & John-Steiner, 2002). Of course, we can only examine the external manifestations of affect and these will vary from participant to participant. Just because a participant did not display any external signs of affect, it does not mean that affect did not play an important part in the learning process for them.

Adam, David and Gregg all expressed engagement and a feeling of success throughout the sessions. Their expressions of confidence seemed to increase as time went on and their sense of sequence and threading developed. Sarah mainly expressed confidence during her time working with sequences and threads. She did express a lack of confidence on one occasion, and this coincided with her building a sequence out of order. Later in the same session she overcame this challenge and her confidence improved. Finally, Steven's expressions of affect seem to form a pattern, with him initially showing signs of a lack of confidence, and then as his apparent confidence with sequences grew, his expressions of confidence and success increased. Steven also seemed to find getting to grips with sequence the most challenging out of all the participants. It is therefore possible that there is a correlation between the pattern in expressions of affect, and levels of challenge and confidence.

6.7.3 One Event, Multiple Actions

In their research, Franklin et al (2017), found that young novice programmers can readily learn the concept of sequence when the sequences that they create consist of events with a single action. In Code Jumper, each play pod does in essence perform one action, it plays a sound. However, there are two parameters to set, and each sound sample can contain multiple sounds. Therefore, each play pod could perform one or multiple actions. There is evidence to suggest that Steven, Adam and Sarah all conceived of the play instruction as an event containing multiple actions at certain points.

Although, as a whole, the participants did not have great difficulty getting to grips with sequence, there were some challenges. It is possible that the way in which the play instruction affords the learner to view it as performing one or multiple actions, could have increased the challenge for some of the participants. Additionally, the design of some of the activities employed sound samples, which contained multiple words or notes, thus potentially giving the impression that the play instruction is performing multiple actions. On some occasions, a sentence was split between two sound samples. In these cases, multiple actions are being performed in each sample, but the student might conclude that the actions from both samples should be grouped into one event, as they form a whole sentence.

6.7.4 Changes in Expression of Sense

Another pattern seemed to emerge in the way in which participants expressed their sense of sequence when new concepts and tools were introduced. For all the participants, perceivable expressions of counting increased when new concepts were introduced, concepts were brought together for the first time, or new tools were introduced. The fact that these expressions of counting come and go at different points, could suggest that the practice does not disappear, they are simply not externally perceivable.

Steven, Adam and David all explored programs in an order other than the order of execution at certain points when they were introduced to new concepts or tools. Additionally, Steven, Adam, Sarah and Gregg struggled to design or build programs in a logical order at some points when new concepts or tools were introduced. These points lead me to suggest a potential relationship between changes in expression of sense, and changes in relative levels of challenge for individual students. For example, returning to previous forms of expression could indicate that the student has encountered a set of concepts or tools which increase the level of challenge for them.

Chapter 7. **Data Analysis: Repetition**

7.1 Introduction

In this chapter I will review and analyse data gathered from activities which focussed on repetition. Between Activities 10 and 25, nine of them focus on either individual loops, loops in a sequence or loops across threads. A further six of these activities focus on the use of nested loops. More details of the individual activities can be found on page 95 in Chapter 5. I will explore how the participants expressed their sense of repetition and how this sense evolved over the course of the sessions. Additionally, I will explore what these different ways of expressing repetition can tell us about the learning process in relation to sequence and threading.

As with Chapter 6, a summary table has been produced for each participant, providing an overview of their expressions of repetition. The activities which have been highlighted for each participant have been chosen to represent the way in which each participant's sense of repetition evolved throughout the sessions. Each participant's journey will be explored in turn before bringing the themes together in the discussion.

A number of types of expression which relate to repetition were identified, however some were not in evidence for all participants. An overview of each type of expression that has not already been covered in Chapter 6 is provided below:

- **Identify repetitions** - the learner identifies the number of repetitions required to solve a problem
- **Need for loop/nested loop** - the learner identifies that a loop or nested loop is needed to solve a problem.
- **Design loop in order/out of order** - the learner either creates a design for a loop in a logical order or finds it challenging to do so.
- **Build loop in order/out of order** – the learner either builds a loop in a logical order or finds it challenging to do so.

- **Loop syntax** – the learner knows what they want to achieve but finds the syntax of loops in Code Jumper to be a barrier to implementation.
- **Explain repetition** – the learner is able to explain the concept of repetition in their own words.
- **Loop = beat** – the learner is thinking of a loop like a music beat as it repeats.
- **Loop = beats** – the learner is thinking of the number of repetitions like the number of beats in a piece of music.
- **Identify total sounds** – the learner is able to identify the total sounds produced by a loop either verbally or through the use of a gesture.
- **Link between loop pod and repetition** – the physical loop pod in Code Jumper has formed a part of the learner’s sense of repetition.
- **Closes empty loop** – the learner tries to close an empty loop without any pods inside it.
- **Loop gesture** – the learner embodies a loop by performing a circular gesture.
- **Representation of a loop** – the learner creates a physical representation of a loop without using a loop pod.
- **Total sounds = no. repetitions** – the learner believes there is a one-to-one relationship between the number of sounds produced by a loop and the number of repetitions.
- **Adjacent instruction inside loop** – when a loop is part of a larger program, the learner is not sure where the demarcation is between instructions that will be inside the loop and the those that will be outside.
- **Duplicate instructions within loop** – the learner believes there is a one-to-one relationship between the number of repetitions and the number of times an instruction needs to appear within a loop.

- **Loops and multiplication** – the learner identifies the multiplicative relationship between repetitions, instructions and the number of sounds produced.
- **Nested loops and addition** – the learner believes that the relationship between the number of repetitions on both loops in a nested loop is additive.
- **Nested loops and multiplication** – the learner believes that the relationship between the number of repetitions on both loops in a nested loop is multiplicative.

As was the case in Chapter 6, the forms of expression observed will be grouped into the following categories:

- Gestures (including exploratory procedures) 🗣️
- Tool use (physical manipulation) 🖐️
- Verbal/sound (speech, noises, laughter etc.) 🗣️

The following sections will explore the development of a sense of repetition for each participant in turn.

7.2 Steven's Sense of Repetition

In Table 18 I have provided a summary of the ways in which Steven's expressions of repetition were manifested throughout the sessions. In the following sub-sections I will explore these expressions to uncover the ways in which Steven's sense evolved over time.

TABLE 18: STEVEN'S EXPRESSIONS OF REPETITION

Steven	10	11	12	13	16	17	18	19	20	A2	21	22	28
Explore in order	☺				☺			☺	☺	☺	☺		
Explore in reverse	☺	☺	☺		☺	☺		☺		☺			
Contour following	☺	☺	☺		☺	☺		☺	☺	☺	☺		
Not contour following					☺					☺			
Identify repetitions	◀		◀				☺◀	◀		◀		◀	
Need for loop					◀		◀	◀			◀	◀	
Need for nested loop									◀				
Design loop in order							☺					☺	
Design loop out of order				☺				☺					
Builds loop in order	☺	☺	☺		☺	☺	☺	☺	☺		☺	☺	☺
Build loop out of order		☺	☺		☺	☺						☺	☺
Loop syntax			☺		☺			☺	☺		☺		☺
Explain repetition													
Loop = beat								◀					◀
Repetitions = beats								◀					
Identify total sounds							☺			☺			
Link between loop pod and repetition								☺◀		☺◀			
Closes empty loop			☺		☺	☺							☺
Loop gesture										☺			
Representation of loop													
Total sounds = no. repetitions		◀	◀							◀			
Adjacent instructions inside loop													
Duplicate instructions within loop				☺						◀			
Loops and multiplication				◀									
Nested loop and addition									◀				
Nested loops and multiplication									◀				
Lack of confidence		◀			◀			◀		◀			
Confidence			◀			☺		◀			◀		
Engagement			◀					◀		◀	◀		
Success			◀		◀			◀			◀		

7.2.1 Exploration and Contour Following

By the time that repetition was introduced in Activity 10, Steven was employing the contour following exploratory procedure to explore sequences regularly, and he quickly adopted it to explore loops as well. However, despite the use of contour following, Steven would often explore loops in reverse, rather than in the order of execution. Looking at the structure of a loop built in Code Jumper, shown in Figure 21, it can be seen that the order of execution operates in a clockwise direction. However, when a learner is exploring a loop using contour following, the first wire they come across leads to the last pod to be executed rather than the first.

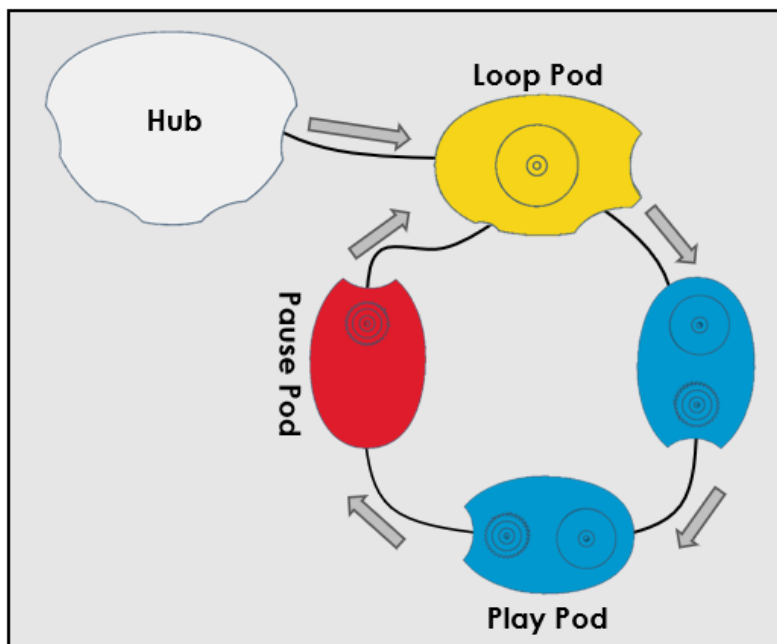


FIGURE 21: CODE JUMPER LOOP STRUCTURE

From observing Steven exploring loops, it is clear to me that when he explored a loop in reverse, he was doing so because he followed the first wire he came across while contour following. In Figure 22, Steven can be seen following loop using the first wire he encountered, causing him to explore it in reverse. For this reason, I do not believe that the exploration of a loop in reverse for Steven bears any relationship to his level of understanding in regard to repetition.

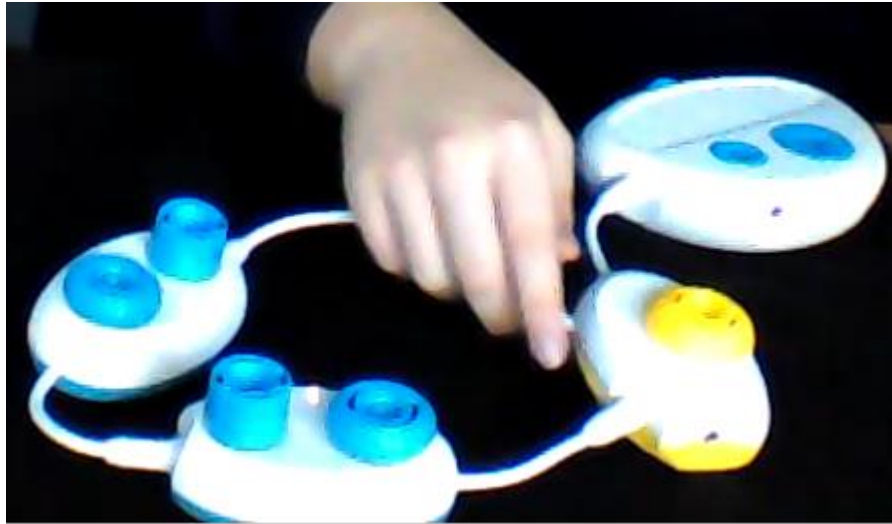


FIGURE 22: STEVEN EXPLORING A LOOP IN REVERSE

7.2.2 Designing and Building Loops

As discussed in Chapter 5, on most occasions when a new concept is introduced, the participants were initially provided with pre-made designs that they needed to use to either complete a program or build one from scratch. As the activities progressed and they became more familiar with a concept, the participants were asked to both design and build programs. The first time Steven designed a program featuring repetition was in Activity 13 and he recognised that a loop was needed to repeat the ‘Row’ instruction 3 times. However, he initially believed that he needed to duplicate the instruction within the loop, which would result in 3 copies of the instruction. He then identified that the relationship between the number of repetitions and the number of instructions was multiplicative, and therefore he only needed one copy of the instruction inside the loop.

Steven quickly demonstrated the ability to build loops in a logical order and was able to do this throughout the remaining sessions. However, there were many occasions when Steven knew what he wanted to achieve, but was not sure how to implement it using the specific syntax of loops in Code Jumper. The occurrences of Steven being unsure about the syntax of loops he wanted to implement sometimes seemed to coincide with the introduction of a new concept or tool. For example, in Activities 20 and 21, Steven had just been introduced to nested loops, and in Activity 28 he was asked to combine repetition with selection for the first time. Additionally, Steven tended to express a lack of confidence when he had just been

introduced to a new concept or tool, but also soon after would display an increase in confidence.

7.2.3 Expressing a Sense of Repetition

Steven expressed his understanding of repetition in other ways. He often would identify the number of repetitions required to solve a given problem and also identify the need for a loop in a given solution. This leads me to conclude that although he sometimes found the syntax of Code Jumper loops challenging, he did develop a good understanding of repetition. This demonstrates that just because a learner cannot construct a functional loop, it does not mean that they do not understand the concept.

7.2.4 Relationship Between Sense and the Physical Tool

For Steven, the physical loop pod in Code Jumper became an important part of his sense of repetition. For example, in Activity 19 Steven was designing a program using the design board and he knew he wanted to repeat a set of sounds eight times but could not remember how. I asked him which pod we would use if we wanted to repeat something and he suddenly exclaimed, “loop pod!” and at the same time he tried to find a loop pod on the table. He did not actually need a loop pod at this stage as he was designing the program, but he did seem to make a strong connection between the physical loop pod and repetition. In the repetition assessment activity Steven was introduced to the mini representation of the loop pod. He found it hard to make the connection between the representation and the real thing, and could not describe how we can set the number of repetitions until he held a real loop pod. Finally, on a number of occasions, when a loop pod had no pods connected inside it, Steven would try and close the empty loop on itself. It was almost as if he felt that it was not a loop unless it was closed.

7.2.5 Relationship to Non-Domain Specific Concepts

At some points Steven seemed to draw upon his experiences of music in the development of his sense of repetition, however many of the activities did involve music. In Activity 19, when referring to a loop he described it as a ‘beat’ and he used ‘beats’ to describe the number of repetitions needed. Steven also referred to a

loop as a beat in Activity 28. Another area that Steven drew upon when constructing his sense of repetition was mathematics and this is explored further in section 7.2.7.

7.2.6 Repetition Assessment Activity

In chapter 6, I described an activity that utilised miniature versions of Code Jumper pods, in order to assess the participants' understanding of sequence. I designed a similar activity to evaluate understanding of repetition. In this activity, the participants were provided with three example programs made out of the miniature pods, and they listened to a recording of a program. The participants needed to decide which of the three programs could have made the sound that they heard. The three programs can be seen in Figure 23. The recording featured two sounds repeated three times and could have been created using either Program A or Program B, depending on whether it uses a loop or a sequence.

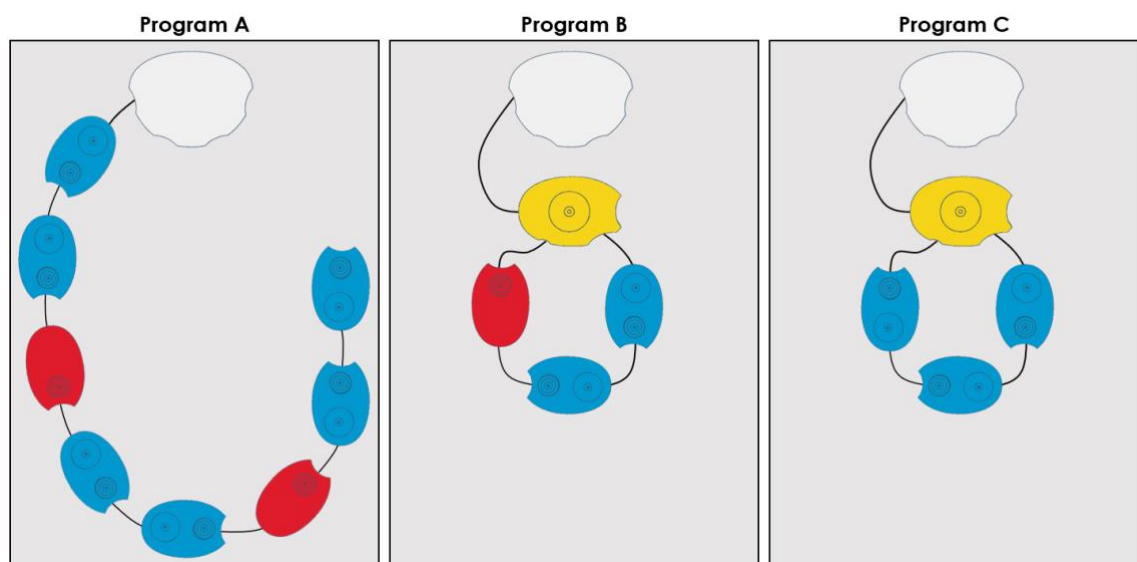


FIGURE 23: REPETITION ASSESSMENT ACTIVITY PROGRAMS

After listening to the recording, Steven quickly identified that the program featured two sounds repeated three times, and after exploring Program A he realised that it could have created the sound. It then took him a while to come to the conclusion that Program B could have also created the sound of the recording. He initially believed that it should be Program C, because the sounds were repeated three times and there were three play pods in the program.

7.2.7 Transitional Theories

Papert (1980) believed that transitional theories are an important part of the learning process and help learners bridge the gap between formal subject knowledge and personal knowledge and experience. They should not be considered as deficiencies, as is implied when the term misconception is employed. Steven demonstrated a number of transitional theories through his expressions of his sense of repetition.

7.2.7.1 Relationship Between Repetitions, Instructions and Sounds

A few times, when stating how many repetitions were required in a program, Steven would state the total number of sounds produced, rather than the actual number of repetitions needed. Of course, in a simple loop with one instruction inside, these numbers would be the same and the theory would apply. However, in Activity 13, as Steven began working with loops containing multiple instructions, he realised that this theory did not always work. His theory then evolved when he realised that there was a multiplicative relationship between the number of repetitions, the number of pods inside a loop, and the number of sounds produced.

On the other hand, early on when working with repetition, Steven would identify that a loop was required to solve a problem and was able to state how many times something needed to repeat. However, he believed that the same instruction needed to be duplicated within the loop. For example, when developing the ‘Row Your Boat’ program for Activity 13, Steven knew that ‘row’ needed to be repeated three times and so wanted to place three ‘row’ instructions inside the loop. This theory then evolved when Steven identified the multiplicative relationship between repetitions and instructions. The theory later resurfaced when he was introduced to the miniature representation of the loop pod in the repetition assessment activity.

7.2.7.2 Nested Loops, Addition and Multiplication

When Steven was first introduced to nested loops in Activity 20, he initially believed that the relationship between the inner and outer loops was additive. He had set the inner loop to five repetitions and the outer loop to two, and was asked to predict how many times the instructions inside the inner loop would play. He predicted that the instructions would play seven times as $5 + 2 = 7$. When he tested his prediction, he

realised that it was not correct, so I asked him what other relationship there could be between the two loops and he identified multiplication.

7.2.8 Summary

Steven clearly developed a strong sense of repetition throughout the activities and although he sometimes found the syntax of loops in Code Jumper challenging, this does not detract from his understanding of the concept. Steven developed a number of transitional theories during the course of the activities and drew upon personal knowledge and experience of music and mathematics, outside the subject domain of computing.

As was the case with sequence, often when new concepts or tools were introduced, Steven returned to earlier forms of expression of his sense of repetition. For instance, he seemed to find the syntax of loops in Code Jumper more challenging when a new concept had been introduced. Additionally he also tended to display a lack of confidence at these points. It is possible that this is an indication of a point of transition in his sense of a concept.

Finally, Steven made a strong connection between his sense of repetition and the physical loop pod provided in Code Jumper. At certain points he needed to hold a loop pod in order to successfully answer questions about repetition. This suggests that, for Steven, having a physical representation of a loop supported the development of his sense of repetition.

7.3 Adam's Sense of Repetition

In Table 19 I have provided a summary of the ways in which Adam's expressions of repetition were manifested throughout the sessions. In the following sub-sections I will explore these expressions to uncover the ways in which Adam's sense of these concepts evolved over time.

TABLE 19: ADAM'S EXPRESSIONS OF REPETITION

Adam	6	10	11	13	18	19	20	21	22	A2	23	27	28	29	31
Explore in order		☺	☺					☺		☺			☺	☺	
Explore in reverse			☺		☺		☺	☺							
Contour following		☺	☺		☺		☺	☺		☺			☺	☺	
Not contour following			☺					☺							
Identify repetitions				◀			◀			◀	◀				
Need for loop				◀		◀			◀	◀		◀	◀		◀
Need for nested loop											◀				
Design loop in order				☞		☞									
Design loop out of order									☞						
Build loop in order		☞	☞	☞	☞	☞	☞	☞	☞		☞	☞		☞	☞
Build loop out of order						☞					☞				
Loop syntax		☞		☞	☞	☞	☞		☞		☞	☞		☞	
Explain repetition				◀								◀			◀
Loop = beat															
Repetitions = beats															
Identify total sounds						☺	☺								
Link between loop pod and repetition															
Closes empty loop															
Loop gesture			☺	☺	☺								☺		☺
Representation of loop	☞														
Total sounds = no. repetitions															
Adjacent instructions inside loop															
Duplicate instructions within loop															
Loops and multiplication				◀											
Nested loop and addition							◀								
Nested loops and multiplication							◀								
Lack of confidence				☞	☞	◀		☞	◀						
Confidence				☞	☞	☞					◀				☞
Engagement			◀	◀		◀									
Success						◀	◀		◀		◀				◀

7.3.1 Exploration and Contour Following

From looking at Table 20, it can be seen that Adam, like Steven, explored loops both in order of execution and in reverse. On two occasions, following in reverse seemed to be triggered by an absence of contour following. However, there were other occasions in which Adam did employ the contour following exploratory procedure and also explored a loop in reverse. As time went on, Adam's exploration of loops in order of execution became more consistent.

7.3.2 Designing and Building Loops

As previously discussed, in Activity 13 they needed to design and build a program to recreate the song 'Row Your Boat', and during the design process Adam identified that they needed to start with a loop that repeated 'Row' three times. He was then able to build the loop quite confidently, seeming sure in how the problem needed to be solved, despite an initial confusion regarding loop syntax which he quickly overcame.

In Activity 19 Adam and Sarah were designing a program that featured multiple loops in a sequence and Adam was able to confidently create the design. This time, when building the program, Adam seemed less sure of himself and there were quite a few occasions where he was unsure of the loop syntax. In Activity 22, Adam and Sarah were designing and building a program which could be solved using nested loops. Sarah identified that they needed a nested loop, however, Adam found it challenging to implement the nested loop both in the design and when building the program in Code Jumper.

Throughout the other activities, Adam demonstrated the ability to build loops successfully, even though he would often initially struggle with the syntax. As time went on, Adam was able to overcome any challenges he faced with syntax more quickly and with less support from myself and other learners. Additionally, earlier on in the sessions, Adam expressed a lack of confidence in relation to repetition, however these expressions seemed to reduce as time went on and were replaced with expressions of confidence and success.

7.3.3 *Expressing a Sense of Repetition*

We have already explored how Adam expressed his sense of repetition in the form of exploration, designing and building programs. However, he did also express his sense in other ways. One example of this occurred in Activity 6, before Adam was explicitly introduced to repetition. Adam had been asked to create a sequence, and once he had done so, he decided to join the ends of the sequence together to form a loop. He held it up and said, “look at this I created a loop”. The loop he created can be seen in Figure 24. We cannot be sure that Adam was thinking in terms of repetition when creating his ‘loop’ but it does demonstrate that the design of Code Jumper and its physical nature affords the representation of repetition in the form of literal loops.



FIGURE 24: ADAM’S ‘LOOP’ FROM ACTIVITY 6

Adam also expressed his sense of repetition in the form of a looping gesture. At some points he would make the gesture above a Code Jumper loop, and at other times he traced a loop shape on the table with his finger. One example of the latter can be seen in Figure 25. He made this looping gesture on the table as he was explaining how a loop could be used to solve a problem. On all occasions, Adam made the looping gesture while talking about loops, demonstrating the connection between the gesture and his sense of repetition. On a couple of occasions, Adam utilised a counting gesture in order to keep track of the total number of sounds produced by a loop.



FIGURE 25: ADAM MAKING A LOOPING GESTURE

Another way that Adam expressed his sense of repetition was through the use of his voice. Like Steven, he was able to identify the need for a loop to solve a given problem consistently and also could often identify the number of repetitions required. For example when asked how they were going to start building the program in Activity 13, Adam replied, “I think a loop pod”. Additionally, Adam was also able to explain what repetition was in his own words on a number of occasions. For instance in Activity 27, when asked to explain repetition, Adam said, "basically if you have a sound you can loop it around... so if you want it to go on and on and on". Adam’s use of the word ‘around’ also implies an element of physicality has formed an aspect of his sense of repetition.

7.3.4 Relationship to Non-Domain Specific Concepts

Like Steven, Adam also drew upon personal experiences outside the domain of computing when developing his sense of repetition, making connections to mathematics. Early on in his work with repetition, Adam identified that the relationship between the number of repetitions, number of instructions and total sounds produced was multiplicative, just as Steven did. Additionally, he initially made the connection between nested loops and addition, before realising that the relationship was also multiplicative.

7.3.5 Repetition Assessment Activity

In the Repetition Assessment Activity (A2), Adam explored the programs in order using the contour following exploratory procedure, in addition to tapping and naming the pods as he went. When listening to the example program, he identified that sounds were repeated. However he initially believed that the two sounds were

repeated two times, but after listening again he realised it was three. Adam realised that Program A in Figure 11 could have created the sound he heard. Sarah also identified that Program B could have created the sound and Adam agreed.

7.3.6 *Transitional Theories*

Steven's transitional theory relating to the relationship between sounds and repetitions started off with the belief that the number of repetitions should equal the number of sounds produced; this then evolved to take account of the multiplicative relationship. Adam's expressions of the relationship between repetitions and sounds demonstrated that he viewed it as multiplicative early on, and there were no occurrences of him expressing a one-to-one relationship. Additionally, Adam did not demonstrate evidence of Steven's theory that linked the number of repetitions to the number of instructions.

7.3.6.1 *Nested Loops, Addition and Multiplication*

When Adam was first introduced to nested loops in Activity 20, he initially drew upon his knowledge of addition, as Steven did. When asked how they how get the instructions to repeat 10 times, Adam suggested setting one pod to 8 and another to 2 to make 10. When they tested their program, Adam realised that the sound played 16 times and when I asked what the connected between the numbers was, he replied, "8 multiplied by 2". Thus, his transitional theory had evolved to recognise the multiplicative relationship through his experiences.

7.3.7 *Summary*

Adam's expressions of repetition throughout the sessions demonstrate that he developed a clear sense and understanding of the concept. Like Steven, Adam did explore loops both in order of execution and in reverse. However, unlike Steven, the occurrences of exploration in order became more consistent as time went on. Additionally, Adam demonstrated the ability to design and build loops throughout, sometimes being hampered by the syntax of loops in Code Jumper. As time went on, Adam was able to overcome the challenges he faced with syntax more quickly.

Adam demonstrated the ability to identify the need for a loop in a given program and the number of repetitions required. Additionally, he was able to explain repetition in

his own words. On a number of occasions, Adam expressed his sense of repetition using a gesture, something that Steven was not observed to do. The appearances of these gestures occurred early on, when he had just been introduced to repetition, and then reappeared later when he was using repetition alongside other programming constructs.

Near the start of his work with repetition, Adam expressed both a lack of confidence and confidence in addition to a sense of success. Following this, there was a short period in which he expressed a lack of confidence and success, but it was not accompanied by a sense of confidence. Towards the end of the sessions, Adam mainly expressed confidence and a sense of success.

7.4 David's Sense of Repetition

In Table 20 I have provided a summary of the ways in which David's expressions of repetition were manifested throughout the sessions. In the following sub-sections I will explore these expressions, to uncover the ways in which his sense of these concepts evolved over time.

TABLE 20: DAVID'S EXPRESSIONS OF REPETITION

David	10	11	13	16	17	19	20	21	22	23	24	25	28	29	30
Explore in order	☺	☺	☺	☺	☺	☺	☺	☺	☺		☺				
Explore in reverse					☺	☺	☺	☺			☺				
Contour following	☺	☺	☺	☺	☺	☺	☺	☺	☺		☺				
Not contour following						☺					☺				
Identify repetitions	◀	◀	◀		◀				◀☺	◀	◀				◀
Need for loop				◀			◀			◀	◀		◀		
Need for nested loop									◀	◀	◀	◀			◀
Design loop in order						☞			☞						
Design loop out of order			☞												
Build loop in order	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞	☞
Build loop out of order				☞					☞	☞		☞		☞	
Loop syntax										☞					
Explain repetition							◀	◀		◀					
Loop = beat															
Repetitions = beats															
Identify total sounds								☺		☺					
Link between loop pod and repetition															
Closes empty loop															
Loop gesture								☺							
Representation of loop															
Total sounds = no. repetitions															
Adjacent instructions inside loop			☞												
Duplicate instructions within loop			☞												
Loops and multiplication															
Nested loop and addition							◀								
Nested loops and multiplication							◀			◀		◀			
Play as subroutine															
Lack of confidence			☞												
Confidence		◀				☞			☞						
Engagement		◀	◀				◀			◀	◀				
Success		◀	◀	◀		◀				◀	◀				

7.4.1 Exploration and Contour Following

When David was first introduced to repetition, he employed the contour following exploratory procedure to discover loops in order of execution. Later, he went through a period of also exploring loops out of order, whilst still contour following. Like Steven, on these occasions David seemed to follow the wire which he reached first upon exploring the loop. There were only two occasions in which David's exploration out of order coincided with a lack of contour following. There did not seem to be a relationship between exploring out of order and the nature of David's understanding of repetition.

7.4.2 Designing and Building Loops

David initially found the process of designing loops challenging, almost displaying a lack of confidence, seeming unsure and hesitant. When designing the 'Row Your Boat' program in Activity 13, David duplicated instructions within the loop and also placed instructions which needed to go outside the loop inside. Although in later activities, he seemed much more confident in the design process for loops.

David's confidence with repetition quickly built, and he soon demonstrated the ability to build loops in Code Jumper, with few difficulties with syntax, unlike Steven and Adam, who both found the Code Jumper syntax for loops challenging. There were some occasions when David initially constructed loops out of a logical order. In Activity 16, the concept of working with loops across multiple threads was relatively new to David, and he initially found it challenging to identify the correct order of pods inside each loop to make the sounds on different threads alternate. David was then able to build loops in a logical order consistently for a number of activities, until he encountered a series of activities that combined multiple nested loops. He then built out of order when variables were introduced.

7.4.3 Expressing a Sense of Repetition

We have seen how David expressed his sense of repetition through exploring, designing and building programs. We will now look at the other ways in which he expressed his sense of repetition throughout the sessions. From early in his work with repetition, David was able to identify the number of repetitions required to solve a given problem, and this continued throughout the sessions. This was mostly

expressed through his voice, however on one occasion he used a gesture by counting the repetitions on his fingers. He also employed a similar gesture to keep track of the total number of sounds produced on a couple of occasions. On one occasion, David produced a looping gesture by repeatedly circling his hand over a loop, sometimes making brief contact with it. This is illustrated in Figure 26. David also regularly demonstrated his sense of repetition by identifying the need for a loop or nested loop where appropriate. In addition to this, David was able to explain the concept of repetition in his own words. In Activity 32 he described repetition as, "er it's basically when you attach a pod that sends the instructions to repeat, and it does and the sequence repeats". I followed this up by asking why repetition is useful and David replied, "so that you could do something twice, like repeat a sound, like if it was in music you might have the same note twice".



FIGURE 26: DAVID MAKING A LOOPING GESTURE IN ACTIVITY 21

7.4.4 Relationship to Non-Domain Specific Concepts

Like Steven and Adam, David also drew upon his personal experiences outside the domain of computing when developing his sense of repetition. In particular, David drew links to mathematics, in a similar manner to Steven and Adam. In Activity 20, when first introduced to nested loops, David initially predicted an additive relationship between the number of repetitions on both loop pods before realising that the relationship was multiplicative. Unlike Steven and Adam, David did not seem to make a connection between the number of repetitions, the number of instructions and multiplication.

7.4.5 Transitional Theories

David's expressions of his sense of repetition, clearly show the presence of transitional theories during the sessions, as was also the case with Adam and Steven.

7.4.5.1 Relationship Between Repetitions, Instructions and Sounds

David quickly demonstrated the ability to identify the number of repetitions that were required to solve a problem. However, he also initially believed that the instruction needed to be duplicated within the loop, in the same way that Steven did. Like Steven, when designing the 'Row Your Boat' program for Activity 13, David identified that 'row' needed to be repeated three times, and placed three copies of the instruction inside the loop. This theory evolved during the course of the activity and did not resurface later like it did for Steven.

7.4.5.2 Adjacent Instructions Within Loop

Another transitional theory that David expressed was also expressed during Activity 13. He placed instructions that needed to occur after the loop, inside the loop instead. He did this both in the design and build stages. It seemed that it was not clear to him where the loop ended and how he could work with loops and sequences in the same program. After going through the process of building the program with all the instructions inside the loop and listening to the result, David developed an awareness of what needed to be outside the loop. This transitional theory evolved at this point and the original theory did not resurface.

7.4.5.3 Nested Loops, Addition and Multiplication

As previously mentioned, David initially believed the relationship between the two loops in a nested loop was additive, in the same way that Adam and Steven did. In Activity 20, after setting the two pods to five and two repetitions respectively, he predicted that the inner instructions would play seven times. However, before testing his prediction he revised it and said, "I think it might play ten times... because it's the outer loop playing the inner loop". Therefore, David revised his theory to identify the multiplicative relationship before hearing the program, unlike Adam and Steven. David also expressed his understanding of nested loops in terms of multiplication on two further occasions in later activities.

7.4.6 Summary

David expressed his sense of repetition in a variety of ways during the course of the sessions, and clearly demonstrates that he developed a clear understanding of repetition and how to implement it. Unlike Adam and Steven, David did not seem to find the syntax of loops in Code Jumper challenging and his confidence remained consistent. However, the design of the loop pod did seem to lead to David exploring loops out of the order of execution on some occasions, as Steven did.

David demonstrated the ability to design and build programs featuring repetition, with some instances of building out of a logical order. Some of these instances seemed to coincide with a relative increase in the level of challenge for David. Similarly, the occurrences of him producing a loop gesture or using a gesture to count the total sounds, also seem to coincide with the increase in complexity of the nested loop programs he was working with.

As was the case with Steven and Adam, connections were drawn between nested loops and mathematics, and David developed a transitional theory, initially linking nested loops to addition, before changing to multiplication. David also expressed Steven's transitional theory that suggested that instructions needed to be duplicated within a loop. Additionally, he initially found it challenging to distinguish which instructions were inside and which were outside a loop.

7.5 Sarah's Sense of Repetition

In Table 21 I have provided a summary of the ways in which Sarah's expressions of repetition were manifested throughout the sessions. In the following sub-sections I will explore these expressions to uncover the ways in which her sense evolved over time.

TABLE 21: SARAH'S EXPRESSIONS OF REPETITION

Sarah	10	11	13	14	16	17	18	19	20	22	A2	23	27	30	31
Explore in order		☺			☺				☺		☺		☺		
Explore in reverse	☺						☺								
Contour following	☺	☺			☺		☺		☺		☺				
Not contour following											☺		☺		
Identify repetitions	◀		◀					◀	☺	◀	◀				
Need for loop			◀	◀	◀				◀	◀		◀			
Need for nested loop										◀				◀	
Design loop in order								☺		☺					
Design loop out of order			☺												
Build loop in order	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺		☺	☺	☺	☺
Build loop out of order			☺		☺	☺									
Loop syntax			☺	☺	☺	☺				☺			☺	☺	☺
Explain repetition									◀	◀			◀		◀
Loop = beat															
Repetitions = beats															
Identify total sounds						☺		☺							
Link between loop pod and repetition					☺										
Closes empty loop															
Loop gesture	☺														☺
Representation of loop															
Total sounds = no. repetitions															
Adjacent instructions inside loop															
Duplicate instructions within loop			☺												
Loops and multiplication															
Nested loop and addition									◀						
Nested loops and multiplication									◀						
Play as subroutine															
Lack of confidence				◀				◀				◀			
Confidence								◀		◀	◀	◀			
Engagement	◀					◀									
Success			◀			☺		◀		☺		◀			

7.5.1 Exploration and Contour Following

When Sarah was introduced to repetition in Activity 10, she initially explored the loop out of the order of execution. However, when I informed them that the loop went clockwise, she explored in order of execution and maintained this pretty consistently throughout the remaining sessions. After I had explained that the loop went clockwise, Sarah made a loop gesture with her finger above the physical loop. This is illustrated in Figure 27. Sarah started using the contour following exploratory procedure to explore loops from the beginning. However, over time she reduced her use of this technique and relied more on her other senses which includes her limited vision. For example, in the Repetition Assessment Activity, Sarah tapped each pod in order, without using the wires to guide her between the pods.



FIGURE 27: SARAH MAKING A LOOPING GESTURE IN ACTIVITY 10

7.5.2 Designing and Building Loops

Sarah created her first program design in Activity 13, when she was designing the ‘Row Your Boat’ program. She identified that a loop could be used to repeat ‘row’ three times. However she also placed three copies of the instruction inside the loop, in much the same way that Steven and David did. When she started to build her program, she also placed three ‘row’ instructions inside the loop. In later activities, Sarah was able to create program designs in a logical order.

Sarah was able to build programs featuring repetition throughout the activities that featured loops. There were a few occasions when she initially built loops out of a logical order, however these often seemed to coincide with activities which raised the level of challenge for her. Additionally, Sarah often found the syntax of loops in Code Jumper challenging, as did Steven and Adam. However, she was always able

to overcome these difficulties and it did not seem to bear any relation to her understanding of repetition of a concept.

7.5.3 Relationship Between Sense and the Physical Tool

Like Steven, Sarah also demonstrated a close link between her sense of repetition and the physical representation provided by Code Jumper. In Activity 16 the participants were completing a program that used repetition across multiple threads to produce a beat using body percussion sounds. When she was building another thread for the program, she realised that the sound she was working with needed to repeat, and when I asked her how we could make the sound play more than once, she pointed to the loop pod and picked it up without saying anything. This demonstrates that the loop pod has become a part of her sense of repetition.

7.5.4 Expressing a Sense of Repetition

We have seen that Sarah has expressed her sense of repetition in a variety of ways and it is evident that she has developed a clear understanding of repetition. She did, however, also express her sense of repetition in other ways. Sarah was able to consistently identify the need for a loop and the number of repetitions required. On most occasions this was expressed through her voice, however on a couple of occasions she employed a gesture to keep track on the number of repetitions in a program. Sarah also employed a gesture to keep count of the total number of sounds produced on some occasions. As time went on, she was also able to express her understanding of repetition in her own words, and at the end of the sessions she produced a looping gesture that involved her circling her fingers round each other as she was explaining repetition.

7.5.5 Relationship to Non-Domain Specific Concepts

As with the other participants, Sarah drew upon her personal experiences of mathematics in the development of her sense of repetition. In particular, she initially identified an additive relationship between the inner and outer loops in nested loops, before realising that the relationship was in fact multiplicative, after she tested the program. In later activities, when working with nested loops, Sarah did not express her sense of nested loops in terms of multiplicative relationships as some participants did.

7.5.6 Repetition Assessment Activity

In the Repetition Assessment Activity (A2), Sarah explored the programs in order initially without contour following, however she then started to use contour following. She tapped each pod and said which type of pod it was. Sarah identified that Program A in Figure 11 could have produced the sound she heard, however she did not think that Program C could have produced the sound, as it had three play pods and there were only two sounds. She also believed that Program B could have produced the sound, and identified that the number of repetitions would need to be set to three.

7.5.7 Transitional Theories

In the course of the sessions, Sarah's expressions of sense provided evidence of some transitional theories:

7.5.7.1 Relationship Between Repetitions, Instructions and Sounds

Like David and Steven, Sarah initially believed that instructions needed to be duplicated within a loop. As previously mentioned, in Activity 13, she identified that 'row' needed to be repeated three times using a loop, and proceeded to place three copies of the instruction inside the loop. After this activity, this transitional theory evolved, as she realised there was not a one-to-one relationship between the number of repetitions and the number of instructions.

7.5.7.2 Nested Loops, Addition and Multiplication

As previously mentioned, Sarah started Activity 20 with the transitional theory that the relationship between the two loop pods in a nested loop was additive. She first tried setting the loop pods to eight and two respectively, and was expecting it to play ten times. When she counted sixteen, she decided that both loops must have been set to eight in error. After checking them she realised that the relationship was multiplicative and suggested setting the pods to five and two, which indicated a conceptual change in her theory.

7.5.8 Summary

Sarah employed a combination of sensory modalities in order to explore programs featuring loops. Towards the start of the sessions, she relied more on touch as she

employed the contour following exploratory procedure to gain an understanding of loops. Over time she transitioned to make more use of her available vision. There does not seem to be a relationship between the order in which she explored loops and her degree of understanding in regard to repetition. Sarah employed gestures in a number of ways, including counting repetitions and total sounds, and in her explanation of the nature of repetition itself.

After some initial challenges, Sarah was able to confidently design and build programs featuring loops. However, she did find the syntax of loops in Code Jumper challenging at certain points, but could always overcome any difficulties. Similarly, towards the start of the sessions, she had no external expressions of confidence. However these became more apparent towards the end. An example of Sarah expressing confidence can be seen in Activity 23, in which after playing their completed program, Sarah says “There we go” confidently, seemingly satisfied with what they had achieved. There is also evidence to suggest that the physical loop pod provided with Code Jumper formed a key part of her sense of repetition as it did for Steven.

Sarah also demonstrated a couple of transitional theories. For instance, she felt that there was a one-to-one relationship between the number of repetitions and the number of instructions inside a loop. Additionally, she made the connection between nested loops and mathematics as the other participants did. This transitional theory started by viewing the relationship between the loops being additive before it evolved, and she identified the multiplicative relationship.

7.6 Gregg’s Sense of Repetition

In Table 22 I have provided a summary of the ways in which Gregg’s expressions of repetition were manifested throughout the sessions. In the following sub-sections I will explore these expressions to uncover the ways in which his sense evolved over time.

TABLE 22: GREGG'S EXPRESSIONS OF REPETITION

Gregg	10	11	13	16	17	18	19	20	21	22	23	24	25	28	32
Explore in order	☺	☺				☺		☺	☺			☺			☺
Explore in reverse						☺	☺					☺			
Contour following	☺	☺													
Not contour following		☺				☺	☺	☺	☺			☺			☺
Identify repetitions		◀	◀		◀						◀	◀	◀	◀	◀
Need for loop											◀	◀		◀	
Need for nested loop								◀			◀			◀	
Design loop in order							☺			☺					
Design loop out of order			☺												
Build loop in order	☺	☺		☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
Build loop out of order				☺		☺	☺		☺	☺	☺		☺		
Loop syntax						☺	☺		☺	☺					☺
Explain repetition															◀
Loop = beat															
Repetitions = beats															
Identify total sounds	◀	☺					☺				☺	☺			
Link between loop pod and repetition															
Closes empty loop															
Loop gesture								☺							
Representation of loop															
Total sounds = no. repetitions															
Adjacent instructions inside loop															
Duplicate instructions within loop			☺												
Loops and multiplication															
Nested loop and addition								◀							
Nested loops and multiplication								◀			◀		◀		
Play as subroutine															
Lack of confidence				☺											
Confidence						☺				◀					
Engagement		◀	◀		◀			◀			◀	◀	◀		
Success											◀	◀	◀		

7.6.1 Exploration and Contour Following

At first, Gregg employed the contour following exploratory procedure in order to familiarise himself with loops in Code Jumper. However, he quickly stopped using contour following and relied more on his other senses, particularly his available vision. On the whole, Gregg mainly explored loops in order of execution, and there did not seem to be a relationship between contour following and the order of exploration. Additionally, there does not seem to be a relationship between the order of exploration and the level of understanding regarding repetition as, on most occasions, Gregg was able to build loops in order during the same activity. However, two of the occurrences of exploring out of order coincided with the introduction of sequential loops, which Gregg seemed to find challenging initially.

7.6.2 Designing and Building Loops

In Activity 13, Gregg had the first opportunity to design a program featuring a loop. Like all the other participants, apart from Adam, Gregg initially believed that instructions needed to be duplicated within a loop and placed three copies of the ‘row’ instruction in the loop. Later, in Activities 19 and 22, Gregg was much more confident with the design process, seeming sure in his actions and with little hesitation, even though he had some initial challenges when implementing his designs in Code Jumper.

Gregg was able to build loops using Code Jumper consistently through his time working with repetition. There were a few occasions when he initially struggled to build programs in a logical order, but he would quickly overcome these obstacles each time. He also found the syntax for implementing loops in Code Jumper challenging at certain points, as did all the other participants apart from Sarah. The occasions when Gregg had difficulties with the loop syntax seemed to coincide with new concepts which he found challenging. For example, in Activities 18 and 19 he was working with sequential loops for the first time, and in Activities 21 and 22 he was working with nested loops for the first time.

7.6.3 Expressing a Sense of Repetition

Gregg expressed his sense of repetition in a number of ways in addition to those discussed thus far. He was able to identify the number of repetitions required to

solve a given problem throughout the sessions, and additionally was able to identify the total number of sounds produced by a loop. He would often express the latter through the use of a gesture, which he employed to keep track of the number of sounds. Towards the end of the sessions, Gregg was also able to identify the need for a loop or nested loop to solve a problem, and he could explain repetition in his own words stating that with repetition “it repeats over and over again”. I followed this up by asking Gregg to explain how he would use it in a program and he said, "I would use it for something like a song, so like a song would go (hums a tune)", David recognised the song that Gregg was humming and Gregg explained that you would need to play that part twice. On one occasion, in Activity 20, Gregg produced a looping gesture above the loop he had created. This is illustrated in Figure 28. Gregg also expressed his sense of repetition by drawing on his personal experience of mathematics, which we will discuss further in the following section.



FIGURE 28: GREGG MAKING A LOOPING GESTURE IN ACTIVITY 20

7.6.4 Relationship to Non-Domain Specific Concepts

As all the other participants did, Gregg drew upon his personal experiences outside the subject domain of computing, and made connections between nested loops and mathematics. At first, he drew upon his knowledge of addition before realising that multiplication was more appropriate.

7.6.5 Transitional Theories

Like the other participants, Gregg demonstrated transitional theories during the development of his sense of repetition. These will be explored further below:

7.6.5.1 Relationship Between Repetitions, Instructions and Sounds

Gregg initially believed that there was a one-to-one relationship between the number of repetitions and the number of times an instruction needed to appear within a loop. All of the other participants, apart from Adam, demonstrated the same transitional theory during Activity 13. Gregg realised that there was not a direct relationship between the number of repetitions and the number of instructions needed within a loop, when Adam identified that the ‘row’ instruction would play nine times.

7.6.5.2 Nested Loops, Addition and Multiplication

All the participants, including Gregg, initially believed that there was an additive relationship between the number of repetitions in both loops in a nested loop. In Activity 20, Gregg predicted that the inner instructions would play seven times as the loops were set to five and two. When he tested his prediction by playing the program he realised that his theory was not quite right. I asked him why it was repeating ten times and he replied, “because the outside you set it to two, and the inside is five so it’s timesing five by two which is ten”. This demonstrates that his transitional theory evolved to recognise the multiplicative relationship in nested loops.

7.6.6 Summary

Throughout the sessions, Gregg expressed his sense of repetition in variety of ways and it is clear that he developed a good understanding of the concept. On the whole he did not use contour following; however, this did not seem to have an impact on his ability to explore programs in order of execution. Although on two occasions when he did explore a loop out of order, he was working with a new concept which he seemed to find challenging. Gregg was able to build loops successfully, although he did sometimes initially find it hard to construct them in a logical order. He also found the syntax for loops in Code Jumper challenging, particularly when he was working with new concepts that he was not comfortable with.

Like the other participants, Gregg drew upon his personal experiences of mathematics when developing his sense of nested loops. He developed a transitional theory, which initially identified an additive relationship between the loops within a nested loop. This then evolved, when he recognised the multiplicative relationship. Another transitional theory which Gregg developed was related to the belief that instructions needed to be duplicated within a loop, to match the number of repetitions.

Gregg was engaged throughout his work with repetition and while he briefly expressed a lack of confidence, he quickly started to display signs of confidence. As time went on, expressions of confidence seemed to evolve into expressions of success, in which he demonstrated pride and a sense of accomplishment on completing a task featuring repetition.

7.7 Discussion

In this chapter we have explored the activities that were deemed to be most relevant for each participant in terms of repetition. For this purpose, each activity for each participant has been treated as a *perezhivanie*, and together they form the *perezhivaniya*, which formed each individual's sense of repetition. Although the activities were the same, the way in which each individual processes these experiences will be different, as will their sense of the concept. However, there are some common themes among the participants which we can explore to gain further insight into the development of a sense of repetition.

7.7.1 Evolution in a Sense of Repetition

All of the participants demonstrated the ability to design and build loops, and express an understanding of repetition. They all expressed their understanding by identifying the need for a loop, and the number of repetitions required to solve a problem. They each had a different journey in their development of an individual sense of repetition. We will now explore the similarities and differences between these journeys.

The syntax for physical loops implemented using Code Jumper was a challenge for all participants at various stages, apart from David, and it is common for novice

programmers to find syntax challenging in introductory programming courses (Qian & Lehman, 2017). According to Qian and Lehman, the literature suggests that there is usually correlation between inadequate syntactic knowledge and conceptual knowledge in novice programmers' learning using text-based languages. However, this does not appear to be the case for my participants, as their difficulties with syntax was not representative of their understanding of repetition, as they were all able to describe how a problem could be solved. This could possibly suggest that the use of a physical programming language facilitated the development of a conceptual understanding of repetition in the learners.

The design of the loop pod did seem to lead some participants to explore loops in reverse, as when using contour following, the first wire that is reached on the loop pod is the return wire. Although the participants did not consistently explore loops in order of execution, this did not seem to relate to their understanding of execution order. For Adam, Gregg and Steven, occurrences of difficulties with loop syntax often seemed to coincide with the introduction of a new concept or tool.

Each participant experienced some challenges in designing and building loops in a logical order, to a greater or lesser extent. Both Adam and Sarah had relatively few problems constructing loops in a logical order. For Sarah and Gregg, the occurrences of building loops out of order often seemed to coincide with the introduction of a new concept. Additionally, for Sarah her confidence in working with repetition seemed to increase as she was constructing loops in a logical order more consistently.

For Adam, his expressions of both confidence and success seemed to increase as he became more familiar with repetition, and for Gregg his expressions of success increased. Steven went through cycles of expressing a lack of confidence when a new concept was introduced, shortly followed by an expression of confidence. Whereas David's expressions of confidence and success seemed to remain pretty consistent throughout the sessions, it is worth noting that David was the only participant that encountered almost no challenges in relation to loop syntax.

7.7.2 Relationship Between Sense and the Physical Representation

The physical representation of repetition in Code Jumper has played an important role in the development of each individual's sense. This was demonstrated in a number of ways. Firstly, both Sarah and Steven felt the need to pick up a loop pod when answering a question on repetition. Secondly, Adam, David, Sarah and Gregg all produced loop gestures, which seemed to be influenced by the physical representation of repetition as a literal loop in Code Jumper. Finally, Adam also created his own representation of a loop using Code Jumper, by connecting the pods in a circle. This occurred before Adam had been introduced to the concept of repetition, which suggests that the design of the tool affords the development of a sense of repetition. It is also worth noting that Adam often produced a loop gesture when he encountered new concepts.

7.7.3 Drawing on Personal Experiences

Another important consideration in the development of a sense of repetition, is the role that personal experiences outside the domain of computing play. All of the participants drew upon their experiences with mathematics in the development of a sense of nested loops. At first, they formed a connection between nested loops and addition, before this evolved into a connection with multiplication. It has been suggested that prior experiences of mathematics can be problematic for novice programmers, as concepts often operate differently in computing (Qian & Lehman, 2017). However, I would argue that prior mathematical knowledge should be seen as a benefit, as it provides a rich source experience to build upon when developing an understanding of programming. When there are differences between mathematics and computing, these can be explored with the learners, rather than viewing them as negative.

In addition to mathematics, Steven also drew upon his experiences with music when developing his sense of repetition. He described a loop as a beat, invoking a repeating rhythm and he described the number of repetitions as the beats, indicating the number of times the rhythm repeated. These personal experiences also played a key role in the development of transitional theories, which we will explore further in the following section.

7.7.4 *Transitional Theories*

As discussed earlier in this chapter, Papert (1980) used the term transitional theory to describe a part of the learning process that helps learners bridge the gap between formal subject knowledge and their personal knowledge and experience. We are using the term in place of misconception, which implies they are deficiencies. In this section we will explore the transitional theories that were in evidence as the participants expressed their sense of repetition during the activities.

7.7.4.1 *Relationship Between Repetitions, Instructions and Sounds*

When Steven first started working with loops, and before he started designing and building his own original loops, he believed that the number of repetitions would be the same as the number of sounds produced, and of course for simple loops with a single instruction inside this theory would apply. None of the other participants displayed evidence of this transitional theory. However, they all, apart from David expressed a belief in a one-to-one relationship between the number of repetitions and the number of times an instruction needed to appear inside a loop. They seemed to believe that if an instruction needed to be repeated three times, they needed to set the number of repetitions to three, and place three copies of the instruction inside the loop. They expected the result to be that the sound would play three times rather than nine.

$$\text{REPETITIONS} = \text{COPIES OF INSTRUCTION}$$

When they tested their theory, the participants no longer believed that there was a one-to-one relationship, however for most of them it is not clear what they believed the relationship to be. This relates to the literature which suggests that novice programmers often get confused regarding the number of times the instructions within a loop will repeat (Qian & Lehman, 2017). Steven, on the other hand, refined his earlier theory, to recognise that there was a multiplicative relationship between the number of repetitions and the number of instructions. This is the form that Adam's transitional theory took from the start.

$$\text{Total sounds} = \text{Repetitions} \times \text{Instructions}$$

Relationship Between Repetitions in Nested Loops and Total Sounds

When they were first introduced to nested loops, all of the participants believed that the relationship between the number of repetitions set on both loops was additive in nature. For instance, the loops would be set to five and two respectively, and they predicted that the instructions inside the inner loop would repeat seven times.

$$\textit{Total sounds} = \textit{Inner repetitions} + \textit{Outer repetitions}$$

Upon testing their prediction, they realised that the result did not tie in with their theory, and they made the connection to multiplication. It is important to note that although David also initially made the connection to addition, he actually realised the link to multiplication before he tested his prediction.

$$\textit{Total sounds} = \textit{Inner repetitions} \times \textit{Outer repetitions}$$

7.7.4.2 Loop Scope

When David was first working with repetition and sequences together, he initially found it challenging to distinguish where the scope of the loop ended. As such, at one point, he placed all instructions inside the loop, even the ones that should have formed a sequence that came after the loop. He had expressed verbally which instructions he wanted to repeat, but did not seem sure how to implement this. After hearing the result of the program, he managed to work out that he needed to place the instructions he did not want to repeat outside the loop. This transitional theory ties in with the misconception outlined by Swidan (2018), in which learners believe adjacent code executes within the loop, therefore demonstrating that their understanding of loop scope is not fully developed.

Chapter 8. **Data Analysis: Selection and Variables**

8.1 Introduction

In this chapter I will review and analyse data gathered from activities which focussed on selection and variables. Most of the students were introduced to selection in the second half of the eighth session, except Steven, who was introduced to it at the start of the ninth. Four activities feature selection between Activities 26 and 32 and four feature variables. More details of the individual activities can be found on page 95 in Chapter 5. I will explore how the participants expressed their sense of selection and variables, and how this sense evolved over the course of the sessions.

As with Chapters 6 and 7, a summary table has been produced for each participant, providing an overview of their expressions of selection and variables. The activities which have been highlighted for each participant, have been chosen to represent the way in which each participant's sense of repetition evolved throughout the sessions. Each participant's journey will be explored in turn, before bringing the themes together in the discussion.

A number of types of expression which relate to selection and variables were identified, however some were not in evidence for all participants. An overview of each type of expression that has not already been covered in Chapters 6 and 7 is provided below:

- **One branch initially** - the learner initially explores only one of the two branches connected to the selection pod.
- **Contour following** - the learner makes some use of contour following when exploring an individual branch or moving between branches.
- **Not contour following** - the learner makes no use of contour following to explore the branches coming out of the selection pod or when moving between branches.
- **Lost between branches** – the learner loses their place in the program when they attempt to move from one branch to another.

- **U shaped gesture** – the learner moves between the two branches by going back up one branch and down the other, almost making a u-shaped gesture.
- **Skip between branches** – the learner skips directly between branches without any contour following.
- **Explore left first** – the learner explores the left branch first.
- **Explore right first** – the learner explores the right branch first.
- **Predict the outcome of a condition** – the learner predicts which branch will execute based on a given condition.
- **Choose appropriate condition** – the learner chooses an appropriate condition so that the desired branch will execute.
- **Explain selection** – the learner explains selection in their own words.
- **Explain counters** – the learner explains counters in their own words.
- **Explain variables** – the learner explains variables in their own words.
- **Identify need for selection** – the learner successfully identifies the need for selection to solve a given problem.
- **Identify need for counters** – the learner successfully identifies the need for counters to solve a given problem.
- **Identify need for variables** – the learner successfully identifies the need for variables to solve a given problem.
- **Link variables to maths** – the learner makes a link between variables in programming and variables in mathematics.
- **Link selection to railway points** – the learner likens selection to a junction in a railway track.
- **Surprised at only one branch playing** – the learner is surprised that only one of the two branches executes.

- **Value assignment** – the learner successfully assigns a value to a variable.
- **Builds program with selection** – the learner successfully builds programs featuring selection.
- **Use counters** – the learner successfully builds programs featuring counters.

As was the case in Chapters 6 and 7, the forms of expression observed will be grouped into the following categories:

- Gestures (including exploratory procedures) 🖐
- Tool use (physical manipulation) 🖐
- Verbal/sound (speech, noises, laughter etc.) 🗣

The following sections will explore the development of a sense of selection and variables for each participant in turn.

8.2 Steven's Sense of Selection and Variables

In Table 23 I have provided a summary of the ways in which Steven's expressions of selection and variables were manifested throughout the sessions. In the following sub-sections I will explore these expressions to uncover the ways in which Steven's sense evolved over time.

TABLE 23: STEVEN'S EXPRESSIONS OF SELECTION AND VARIABLES

Steven	26	27	28	29	31
One branch initially	↺	↺	↺		
Contour Following	↺	↺	↺		
Not contour following		↺	↺		
Lost between branches		↺	↺		
Skip between branches	↺	↺			
U shape gesture			↺		
Explore left first		↺			
Explore right first	↺		↺		
Predict outcome of condition	◀	◀	◀		
Choose appropriate condition	◀✋	◀✋	✋		
Explain selection			◀		
Explain counters					◀
Explain variables					
Identify need for selection					
Identify need for counters					
Identify need for variables					
Link variables to maths					
Link selection to railway points					
Surprised at only one branch playing					
Value assignment				✋	
Builds program with selection	✋	✋	✋		
Use counters					✋
Lack of confidence	◀				
Confidence			◀		
Engagement	◀			◀	◀
Success	◀	◀	◀	◀	

8.2.1 Exploration and Contour Following

Steven employed contour following to locate the selection pod at the end of a sequence. Once he reached the selection pod, he would initially explore one branch

and this varied between the left and right. In Activity 26, he was able skip straight across to the other branch, however in Activity 27 he got lost and had to go back to the selection pod. He then explored the left branch first, before going back up and then down the right branch, making a U-shaped gesture. Steven is in the process of performing the U-shaped gesture in Figure 29.



FIGURE 29: STEVEN EXPLORING THE BRANCHES IN ACTIVITY 27

8.2.2 Expressing a Sense of Selection

Steven was introduced to the selection pod in Activity 26 and was initially confused regarding how the condition affected the flow of the program, being unable to identify which of the two branches would play given the condition he chose. He said, “I’m actually confused a bit” and I explained that the right-hand branch would play, as the answer to the question he set on the selection pod was no. I then asked him to suggest a condition which would make the other branch play and he said, “if six is bigger than four”. Following this he was able to consistently choose appropriate conditions and predict the outcome of a given condition in the following activities. Steven also expressed his sense of selection through the successful creation of program featuring the construct, often expressing a feeling of success upon completing these activities. He was also able to explain selection in his own words during Activity 28. When asked how conditions in selection worked, he said “is that number greater than the other number”.

8.2.3 Expressing a Sense of Variables and Counters

In Activity 29, Steven was introduced to variables and built a simple program featuring a sequence of two play pods. He plugged a variable into both play pods and I asked him to choose a value to plug into the first variable. He assigned 8 to the

variable and the program said “eight” twice when it played. Following this I asked Steven to change the value stored in the variable, and he had no problem choosing another value and assigning it. I then asked Steven what would have happened if we replaced the value with the random plug and he replied, “I think it might give you a random number” and identified that the generated number would play twice.

In Activity 31, Steven initially built a program that used a counter plug to increment a value, in order to play a scale of notes going up with some support from myself. Following this I gave him the countdown plug and explained that it subtracts or takes away. When asked what would happen if he used it in his program, he was able to predict that “it’s gonna go down”, seemingly referring to the notes.

8.2.4 Summary

The number of activities that covered these concepts was relatively small compared to sequence, threading, and repetition and therefore the students did not get to spend much time working on selection and variables. Steven worked at a slightly different pace to the other students, and therefore did not get the opportunity to complete as many of the activities relating to selection and variables. As a result, Steven spent even less time working on these concepts and had a reduced window in which to express his sense of them. Even so, he clearly developed a strong sense of selection, demonstrating the ability to build programs, choose conditions and predict outcomes. On the other hand, his sense of variables and counters seemed to be just starting to develop at the end of the sessions, with one instance of value assignment and expressing the function of a counter verbally.

8.3 Adam’s Sense of Selection and Variables

In Table 24 I have provided a summary of the ways in which Adam’s expressions of selection and variables were manifested throughout the sessions. In the following sub-sections I will explore these expressions, to uncover the ways in which Adam’s sense evolved over time.

TABLE 24: ADAM'S EXPRESSIONS OF SELECTION AND VARIABLES

Adam	26	27	28	29	30	31	32
One branch initially	↷		↷				↷
Contour Following	↷		↷				↷
Not contour following	↷		↷				
Lost between branches			↷				
Skip between branches	↷		↷				
U shape gesture							
Explore left first	↷		↷				↷
Explore right first							
Predict outcome of condition		◀					
Choose appropriate condition	✋	✋					◀
Explain selection						◀	
Explain counters						◀	
Explain variables							
Identify need for selection							◀
Identify need for counters							◀
Identify need for variables							◀
Link variables to maths				◀		◀	
Link selection to railway points							
Surprised at only one branch playing							
Value assignment				✋	✋	✋	✋
Builds program with selection	✋	✋	✋				✋
Lack of confidence						◀	
Confidence							
Engagement	◀	◀		◀	◀	◀	
Success		◀				◀	◀

8.3.1 Exploration and Contour Following

Like Steven, Adam used contour following to locate the selection pod and then would initially explore one branch. In Adam's case, he usually explored the left branch first. He would then skip to the other branch by lifting his hand up and

placing it directly on the branch without touching the selection pod in-between. However, on one occasion he did get lost when trying to skip to the other branch. He did not use contour following or a u-shaped gesture to move between branches.

8.3.2 Expressing a Sense of Selection

After being introduced to selection in Activity 26, Adam was able to successfully create programs featuring the construct on a number of occasions. Like Steven he demonstrated the ability to choose appropriate conditions on a number of occasions, however he only predicted the outcome of a condition on one occasion. He seemed to prefer to play the program to find out which branch would play, rather than externally expressing a prediction. It is possible that he was making predictions internally but was not confident enough to externally express them.

In Activity 27, Adam and Sarah were asked to build a story with two possible endings. They started by building a sequence for the main part of the story and then I asked them how they could have two possible endings. Adam identified that they needed to use the selection pod. Adam was also able to identify the need for selection in Activity 32, when they were given a problem which involved recreating a countdown that went from 5 to 2 and then played an explosion sound.

8.3.3 Expressing a Sense of Variables and Counters

Although Adam had a limited time to work with variables, he did seem to develop a good sense of how they operate. When he was first introduced to the concept in Activity 29, he immediately made the link to his experience of the concept in mathematics, asking “like in maths?” and stating that they are used in algebra. He then proceeded to assign the value 6 to a variable, and use the variable later in the program. Adam was introduced to counters in Activity 31, in which they created a program to produce a piano scale, and he identified that the sounds “goes one up” when the + counter is used. When asked how they could make the notes go back down again, Adam suggested using the – counter. During Activity 32 when recreating the countdown, Adam identified the need for counters and variables in order to solve the problem. Adam was also asked in Activity 31 to explain variables and he made the connection to algebra once again, commenting on the difference

between the concept in maths and computing by saying “it’s sort of unrelated, but related”.

8.3.4 Summary

In the limited time Adam had to work with selection, he developed a strong sense of the concept, as evidenced by his ability to repeatedly construct programs featuring it, and being able to choose appropriate conditions. In the development of his sense of variables, Adam drew upon his experiences of the concept in the mathematics domain, and was able to successfully assign values and identify when variables and counters were required.

8.4 David’s Sense of Selection and Variables

In Table 25 I have provided a summary of the ways in which David’s expressions of selection and variables were manifested throughout the sessions. In the following sub-sections I will explore these expressions to uncover the ways in which David’s sense evolved over time.

TABLE 25: DAVID'S EXPRESSIONS OF SELECTION AND VARIABLES

David	26	27	28	29	30	31	32	33
One branch initially	↷		↷				↷	
Contour Following	↷	↷	↷				↷	
Not contour following	↷		↷				↷	
Lost between branches								
Skip between branches	↷		↷				↷	
U shape gesture								
Explore left first	↷	↷					↷	
Explore right first			↷				↷	
Predict outcome of condition								
Choose appropriate condition		✋					◀	◀
Explain selection		◀					◀	
Explain counters						◀	◀	
Explain variables							◀	
Identify need for selection		◀✋					◀	◀
Identify need for counters						✋	◀	
Identify need for variables						◀	◀	
Link variables to maths								
Link selection to railway points	◀							
Surprised at only one branch playing	◀							
Value assignment				✋	✋	✋	✋	
Builds program with selection	✋	✋	✋				✋	
Lack of confidence						◀		
Confidence			◀					
Engagement	◀				◀	◀		◀
Success	◀	◀						◀

8.4.1 Exploration and Contour Following

Like Steven and Adam, David also employed contour following when locating the selection pod within a program and explored one branch initially. The branch he explored first varied between the left and the right. Additionally, when moving

between the branches, David did not use contour following and could skip directly between them. Unlike Adam and Steven, David did not get lost at any point when moving between branches.

8.4.2 Expressing a Sense of Selection

When David was first introduced to the selection pod in Activity 26 and he explored it, he likened it to junction on a railway track. He said, “I think it’s like a junction” and then added, “I think it's when two programs um you can now have two like on a railway track”. David and Gregg then each created a sequence and added them to either side of the selection pod. When they played their program, David was surprised when his side did not play but Gregg’s did. I explained that it asked whether one number was greater than the other number, and if it were the left branch would play, and if it was not, the right branch would play.

In Activity 27, when Gregg and David were building their dynamic story, I asked them how they could make their story have two possible endings, and David immediately went to pick up the selection pod. He also set the condition on the selection pod so that there was an even chance of each branch being chosen. At the start of Activity 32, I asked David to explain what selection was and he said, “it's when you've got two programs and you want to sort of um separate them and select which one you want... so it gives you an option like you if you want some variation in the program then you use that”. Although David demonstrated the ability to choose appropriate conditions on a number of occasions, he did not externally express predictions of the outcome of conditions as Gregg was quick to offer his own.

8.4.3 Expressing a Sense of Variables and Counters

Gregg and David were introduced to variables in Activity 29, and Gregg chose the value 1 to assign to the variable first, followed by David choosing the value 5. He was quickly able to assign values successfully. In Activity 30 they had two play pods inside a loop, which played the same note twice with the aid of a variable. I asked David what would happen if we added the random plug to the variable rather than a set value and he said, “it will choose a random variable”, and when I asked him what we would hear he replied, “different notes”. I followed this question by

asking how many times each note would play and he said, “eight... no different numbers...twice”. He seemed to recognise that a random number would be generated, and the variable would be used to play the corresponding note twice and this was before he had tried implementing it. After playing the program David reacted by saying, “that’s quite cool” and by asking “I presume you get a different tune each time?” He had identified that, in effect, the program produces random music. When David was asked to describe what a variable was later, he said, “for storing something we can use later on in the program”.

In Activity 31, when David and Gregg were introduced to counters, they started by creating a program with a loop and a single play pod inside it. They placed the minus counter into the play pod, and as the program was playing David realised “oh it goes down”, seemingly referring to the notes in the sound set. Following this I asked them how they could make the notes go back up again, and both Gregg and David initially seemed confused. After some thinking time David suggested adding another loop with the + counter in the play pod. They worked together to implement David’s idea and at the end he remarked, “it’s harder than I thought it would be”.

8.4.4 Summary

When first introduced to the selection pod, David drew upon his knowledge of railways, and likened it to the junction on a railway track. He quickly demonstrated the ability to build programs featuring selection and choose appropriate conditions for them. Additionally, he was able to identify when selection, variables and counters were required to solve a given problem and could explain these concepts in his own words. After being introduced to variables, he was able to consistently assign values to them successfully.

8.5 Sarah’s Sense of Selection and Variables

In Table 26 I have provided a summary of the ways in which Sarah’s expressions of selection and variables were manifested throughout the sessions. In the following sub-sections I will explore these expressions to uncover the ways in which Sarah’s sense evolved over time.

TABLE 26: SARAH'S EXPRESSIONS OF SELECTION AND VARIABLES

Sarah	26	27	28	29	30	31	32
One branch initially	↪		↪				
Contour Following							
Not contour following	↪		↪				
Lost between branches							
Skip between branches	↪						
U shape gesture							
Explore left first	↪		↪				
Explore right first			↪				
Predict outcome of condition	◀	◀					
Choose appropriate condition		✋					
Explain selection						◀	
Explain counters						◀	
Explain variables							
Identify need for selection							◀
Identify need for counters							
Identify need for variables							◀
Link variables to maths							
Link selection to railway points							
Surprised at only one branch playing							
Value assignment				✋	✋		✋
Builds program with selection	✋	✋	✋				✋
Lack of confidence							
Confidence	◀		✋				✋
Engagement		◀					
Success							◀

8.5.1 Exploration and Contour Following

When locating the selection pod, Sarah usually went straight to it rather than using contour following. She would then initially explore one branch, and this would vary

between the left and the right. Like David, Sarah was able to skip directly between the two branches without the use of contour following and without getting lost.

8.5.2 Expressing a Sense of Selection

Sarah and Adam were working together when Sarah was introduced to selection in Activity 26. After they played their first program featuring selection, Sarah asked what would happen if both numbers in the condition were the same, and I suggested they tried it. Adam and Sarah realised that the answer to the condition would be no and after playing the program said “oh... I get it”. Once she had been introduced to selection, Sarah seemed confident in building programs featuring it throughout the remaining activities. Sarah demonstrated the ability to predict the outcome of conditions, however she only chose a condition on one occasion. On the other occasions when a condition needed to be chosen, she seemed happy to let Adam do it.

In Activity 32, Sarah and Adam were building the countdown program and had used a loop and counter to countdown from five to one. At this point they realised they needed to find a way to make the explosion sound play rather than one, and Sarah correctly identified that they needed to use the selection pod for this. Sarah was also able to explain selection in her own words. When explaining selection she said, “it's like whichever one is the higher number, that sound goes”.

8.5.3 Expressing and Sense of Variables and Counters

When Sarah and Adam were first introduced to variables in Activity 29, they started by building the program which used variables to play the same note twice within a loop. Adam assigned the value 6 to the variable, and when I asked them what they thought would happen when the program played, Sarah said she thought the loop would play six times. After changing the value to 5, Sarah realised that the value changed the note rather than the number of repetitions.

Adam and Sarah were introduced to counters in Activity 31, and after Adam had built the structure of the program, Sarah added the + counter to the play pod inside the loop. After testing the program, I asked Sarah what the counter was doing in the program and she said, “it's adding the sound”. By this, she seemed to mean that it was moving to the next sound in the sound set, which is correct.

8.5.4 Summary

Once introduced to selection, Sarah was able to consistently and confidently build programs incorporating it, in addition to being able to successfully predict the outcomes of conditions. She only chose a condition on one occasion, letting Adam make the choice in the other activities. Sarah described selection and counters in her own words, and could identify when selection and variables were required in the solution to a given problem.

8.6 Gregg's Sense of Selection and Variables

In Table 27, I have provided a summary of the ways in which Gregg's expressions of selection and variables were manifested throughout the sessions. In the following sub-sections I will explore these expressions to uncover the ways in which Gregg's sense evolved over time.

TABLE 27: GREGG'S EXPRESSIONS OF SELECTION AND VARIABLES

Gregg	26	27	28	29	30	31	32	33
One branch initially		↷					↷	
Contour Following								
Not contour following		↷					↷	↷
Lost between branches								
Skip between branches							↷	↷
U shape gesture								
Explore left first		↷					↷	
Explore right first								↷
Predict outcome of condition	◀						◀	
Choose appropriate condition	◀	◀✋					◀	✋
Explain selection							◀	
Explain counters						◀		
Explain variables								
Identify need for selection								◀
Identify need for counters								
Identify need for variables								
Link variables to maths				◀				
Link selection to railway tracks								
Surprised at only one branch playing								
Value assignment				✋	✋			
Builds program with selection		✋	✋				✋	✋
Lack of confidence								
Confidence								
Engagement	◀		↷		◀	◀	◀	◀
Success	◀	◀				◀	◀	

8.6.1 Exploration and Contour Following

Like Sarah, Gregg also did not employ contour following in order to locate the selection pod and initially explored one branch. The branch he chose first varied between the left and the right. Gregg was also able to skip directly between the

branches without contour following or losing his place, as both David and Sarah also did.

8.6.2 Expressing a Sense of Selection

David and Gregg were introduced to selection in Activity 26, and they each created a sequence which they plugged into the selection pod. They initially set the condition so that the first value was lower than the second, and as a result, Gregg's sequence played. I then asked them how they could make David's sequence play and Gregg suggested setting the values to 8 and 8 so they were equal, and Gregg's sequence still played. I asked Gregg to think about the question "is eight bigger than eight", he realised the answer was no, and set the first value to 7, thus making David's sequence play. Following this, Gregg and David randomly set the values and Gregg was able to correctly predict which branch of the program would play.

Gregg was able to successfully build programs featuring selection on four occasions, and could also explain the concept in his own words. He said, "so like if you wanted two endings... so like Helen saw her friends in the cupboard and they yelled surprise or if you wanted, she screamed because she saw a ghost or something like that, you could have two of them, and then you would choose a number and whichever one wins it would just choose it". Additionally, in Activity 33 Gregg also demonstrated the ability to identify when selection was required in the solution for a given problem. They were building a story but decided that they wanted it to have two different endings, so Gregg suggested that they needed to use the selection pod to achieve this.

8.6.3 Expressing a Sense of Variables and Counters

Like Adam, when Gregg was first introduced to variables in Activity 29, he drew upon his experiences in mathematics. After he had assigned his first value, I had explained to them that they had stored the value 1 in x and Gregg replied, "it's like maths... algebra". Gregg proceeded to assign another value to a variable in the following activity. In Activity 31, Gregg and David were introduced to counters and created a loop with a single play pod inside. They added the minus counter to the play pod, and I asked them to predict what effect it would have on their program and Gregg replied, "it will minus it", seemingly suggesting that it would go to the

previous sound in the sound set, which is correct. When they tested the program, Gregg listened to the notes going down and likened the effect to a staircase “so it’s a staircase”.

8.6.4 Summary

Once he was introduced to selection, Gregg quickly demonstrated the ability to build programs featuring selection, in addition to choosing appropriate conditions. He also explained selection in his own words, and could identify when selection was required to solve a given problem. When Gregg was introduced to variables he drew upon his experiences in mathematics, making the connection to algebra as Adam did. He demonstrated the ability to successfully assign values to variables and explain counters in his own words.

8.7 Discussion

8.7.1 Expressing a Sense of Selection and Variables

Steven, Adam and David used contour following in order to locate the selection pod within a program, and Sarah and Gregg were able to go directly to it. Once they reached the selection pod, they would typically explore only one branch initially. The branch explored first varied between the left and the right, however the left was most common. It was explained to them that the program would take the left branch if the answer to the question was true, so this may have had some influence on the branch they chose when exploring programs featuring selection.

When moving between branches, most of the participants skipped directly between them without any contour following, the positioning of them side by side seemed to facilitate this. At one point Steven and Adam both got lost in the program when moving between branches. They both were able to relocate the selection pod through contour following, and Adam was then able to move straight to the other branch without further contour following. Steven, on the other hand, made a u-shaped gesture to move from the end of one branch to the end of the other.

We reached the activities featuring selection and variables towards the end of the 10 sessions and therefore had a limited time to work on these concepts. Despite this, all participants were able to construct programs featuring selection, choose appropriate

conditions and predicted the outcome of conditions. Furthermore, they could also explain selection in their own words. Due to time constraints, the participants spent even less time on variables than they did with selection. However, all the participants quickly demonstrated the ability to assign a value to a variable. Additionally, Adam, David and Sarah were also able to identify when a variable was required in the solution to a problem.

A number of the participants drew upon their experiences outside the domain of computing when developing their sense of selection and variables. For example, Adam and Gregg both made a connection to their experiences of variables in mathematics, when they were introduced to variables in programming. Furthermore, David drew upon his knowledge of railways when developing his sense of selection, likening the selection pod to a junction in a railway track.

8.7.2 Transitional Theories

Although David likened the selection pod to a junction in a railway track, he was still surprised that only one branch connected to a selection pod played the first time he played a program featuring selection. It has been identified in the past that some novice programmers believe that both branches of an if statement will always execute (Sorva, 2018). In text-based programming languages the learner might expect the instructions within each branch to be executed sequentially, however given the way in which branches are laid out side by side in Code Jumper it is logical that David expected them to play at the same time. Once David had heard the output of a program featuring selection, his transitional theory quickly evolved as he realised that only one branch would execute depending on the result of the condition. This would also tie in with his train-track metaphor, with the condition dictating which track the program would follow.

Mathematics has been cited as the source of a number of misconceptions in programming, with variables being highlighted as a common example (Qian & Lehman, 2017; Sorva, 2018). It is pointed out that there are subtle differences in many of the concepts shared between the two domains. Both Adam and Gregg seemed to benefit from drawing upon their knowledge of variables from the mathematics domain, however as their time working with variables in programming was limited, it is not possible to say whether or not the differences between the

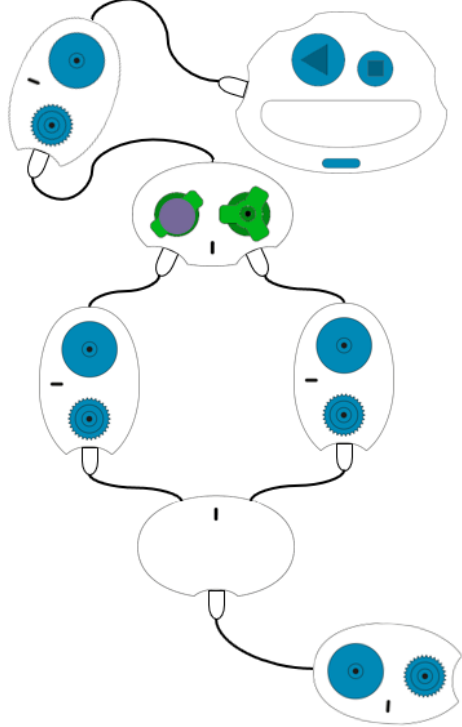
domains would have resulted in difficulties later on. Sorva (2018) highlighted a misconception relating to the direction in which assignment statements operate. In most modern programming languages, the = sign is used to indicate that the value to the right is being assigned to the value on the left. However, in mathematics the = sign indicates equality, and that the values of the expressions on either side of the = sign in equations should be equal. Sorva notes that the different usage between domains can lead to confusion regarding the direction values are assigned in.

No such confusion was in evidence when the participants were working with variables in Code Jumper; this may be because in this programming language, rather than using the = sign, Code Jumper uses plugs which are inserted into each other, making the direction of assignment clear. I would argue that the potential confusion that Sorva labels a misconception is due to the representation of variables in text-based programming languages and the choice to change the function of the = operator. It is quite reasonable for learners to assume that assignment could go in both directions given how the operator is employed in mathematics. Calling it a misconception implies that there is something wrong with a learner's conception of a variable, rather than an understandable confusion caused by the choice of representation.

8.7.3 Relationship Between Sense and the Physical Representation

The evidence presented in this chapter has demonstrated the multiple ways in which the physical representation of the concepts in Code Jumper has shaped the development of a sense of selection and variables. For example, just from exploring the selection pod, David was able to make the connection to his experiences of railways, in likening the pod to a railway junction. This is before he had used selection in a program, which suggests the physical design of the selection pod lends itself the development of a sense of selection in some learners.

TABLE 28: EXAMPLE SELECTION PROGRAMS 1

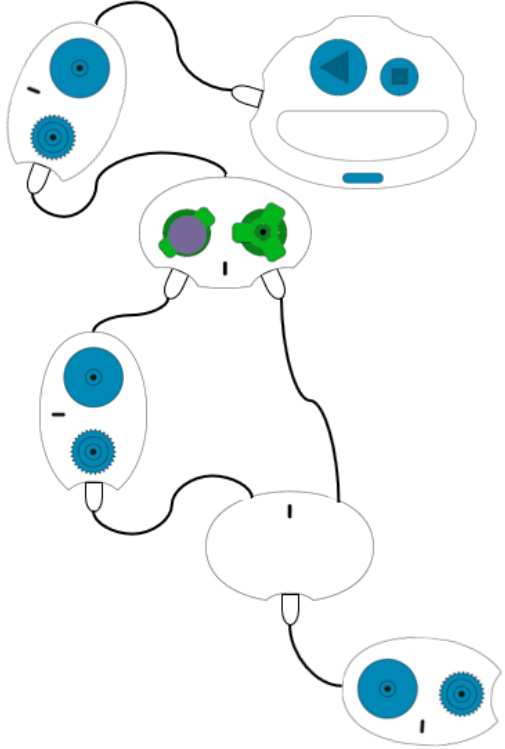
Program A	Program B
	<pre data-bbox="863 324 1209 573"> PLAY SAMPLE 1 IF RANDOM > 4 THEN PLAY SAMPLE 2 ELSE PLAY SAMPLE 3 END IF PLAY SAMPLE 4 </pre>

From looking at the example programs featuring selection shown in Table 28, it can be seen that the physical and text-based representations of selection are very different in structure. In Program B the text is linear, and the flow of control would skip certain lines depending on the outcome of the condition. Whereas in Program A the instructions are placed side by side and the condition selects which side to follow. It could be argued that the way that the flow of control operates in selection is much clearer in Program A. However, it could also lead learners to believe that both branches will execute simultaneously as David did initially.

A review of the literature by Swidan et al. (2018) identified that some novice programmers may believe that the flow of the program returns to the start if the result of a condition is false and no else branch exists. Swidan et al. also found that some learners may think that if the outcome of a condition is false and no else branch is present then the program will end. Unfortunately, we did not work with any programs which featured only one branch, and therefore cannot comment as to whether this belief was present. However, looking at the example programs shown in Table 29, it can be seen that when there are no instructions that are executed if a

condition is false, then this is represented using an empty branch. This makes it clear the program will continue onto the next instruction and could potentially avoid these misconceptions.

TABLE 29: EXAMPLE SELECTION PROGRAMS 2

Program A	Program B
	<pre data-bbox="863 517 1209 696"> PLAY SAMPLE 1 IF RANDOM > 4 THEN PLAY SAMPLE 2 END IF PLAY SAMPLE 4 </pre>

It has been suggested that drawing on experiences of variables in mathematics, can lead to some learners getting confused regarding the order of assignment. This potential confusion was not evident in the participants of my study, and it is possible that the design of variables in Code Jumper, that requires learners to physically plug a value into a variable, makes the direction of assignment clear and avoids this potential confusion.

Chapter 9. Discussion: Developing a Sense of Programming

9.1 Introduction

This study set out to develop an understanding of the processes by which blind and partially sighted learners develop their sense of programming concepts. In Chapters 6 to 8 we examined the data collected during the intervention, to gain an insight into the development of a sense of specific programming concepts. In this chapter we will explore the development of a sense of programming more broadly, by discussing the factors which played a role in the process. These aspects will be examined in relation to the literature introduced in Chapters 2 to 4, and implications drawn.

9.2 The Role of Tools

In Chapter 2 we learnt that *perezhivaniya* play an important role in the development of sense. A *perezhivanie* forms at the intersection between the personality and the environment, encompassing both an experience and the processing of it (Vygotsky, 1987). An individual's numerous *perezhivaniya* form a unity, which contribute to the development of sense (Blunden, 2016; Mattosinho Bernardes, 2018). Tools shape the *perezhivaniya*, from which sense is appropriated, and therefore tools can be seen as playing an important role in mediating its development (Wertsch, 2007).

This study employed Vygotsky's (1987) double stimulation method in order to elicit the development of a sense of programming in the learners. In the context of this method, tools can be considered stimuli, and two types of stimuli were utilised in this research: the 'stimulus-end' which is the problem the learner was asked to solve; and the 'stimulus-means' which are the tools that the learner may draw upon to support them in solving the problem. Through observation of the way in which the learners interacted with the stimuli, we were able to gain an insight into the development of their sense of programming, which, as the research progressed, led to the offering of new stimulus-means in the form of student activity, that appeared to support the appropriation of sense. We analysed interactions by examining external speech, private speech (Berk, 1992) and tool use (Miller, 2011). This section will now examine the role which stimulus-means and stimulus-ends played in the

development of a sense of programming. The stimulus-means include physical representations of programming, gestures, and the spoken word. The stimulus-ends are the problems learners were asked to solve and how they were designed.

9.2.1 Stimulus-Means

9.2.1.1 Physical Representations of Programming

One of the challenges that blind and partially sighted learners face when working with a text or block-based programming language, is the lack of a global overview of the program. Using a screen reader, they can only have one line read out to them at a time, which makes it hard to appreciate the overall structure of a program and the relationships between different instructions (Morrison et al., 2019). Physical representations help to address this challenge, by facilitating free exploration with the hands, and enabling the learners to appreciate the relationships tangibly. My research utilised two main physical representations - a physical programming language in the form of Code Jumper, and a design tool in the form of a board with magnetic strips of text. The role of these representations in the development of sense will be explored in the following sections.

Code Jumper

The Code Jumper physical programming language was chosen as the primary programming tool and stimulus-means for use during the intervention, as it is accessible to blind and partially sighted learners. The design of Code Jumper itself, and the way it represents different concepts, has shaped the sense of programming for each learner, and can be viewed as being a part of their sense. This relationship was demonstrated most clearly in the learners' use of gestures. For example, the looping gestures produced by the learners seemed to be influenced by the physical representation of a loop in Code Jumper. These gestures mostly occurred when the learners were describing an aspect of repetition, indicating the relationship between the gesture and their sense. Therefore, it is more than a gesture, it is embodiment of their sense of repetition. The connection between the physical pods and the concepts was also in evidence when the learners needed to hold or touch a relevant pod, when answering questions relating to that construct. It seemed that the act of feeling the pod was an important part of their triggering and drawing upon previous

perezhivaniya, perhaps through imagined re-enaction of their past experiences with the pod in question.

The physical way in which programming concepts are represented in Code Jumper seemed to also influence the accessibility of aspects of the concept itself. Some concepts, which have been demonstrated to be particularly challenging in past research, appeared to be appropriated relatively easily for the learners in this study. In particular, threading seemed to pose few challenges for the learners. The way Code Jumper represents the threading could be a key factor here, as it employs a physical layout of threads that are side-by-side, almost embodying the concept in contrast to text-based languages, which force a linearity in their presentation of threading. Another concept which is often perceived as challenging is variables. The direction of assignment has been cited as a source of confusion, as the = operator is usually employed for this purpose. In programming it denotes the assignment of a value to a variable, however in mathematics the same operator is used to indicate equality, thus leading to confusion in learners. In Code Jumper, the process of assignment is a physical, or embodied action, in which the learner places a value plug into a variable, thus making the direction of assignment clear.

These examples imply that for some concepts that are perceived as challenging, the challenge is not only related to the concept, but also by the particular form through which it is represented. This could be seen as evidence of how the tools become an integral part of thinking about programming concepts – they do not just offer access to knowledge of programming, they become part of it. This in itself could be an important justification for including a range of ways of representing concepts in computing education.

Design Board

The design board is another stimulus-means that was introduced to enable learners to plan and design their programs, before implementing them using Code Jumper. In the pilot study, it was found that when the learners were given a problem to solve which involved building a program from nothing, they often struggled to know where to start, or found it challenging to remember the order the instructions needed to go in. These findings tied in with research carried out by Waite et al. (2018), which highlighted the importance of the design or planning stage when learning to

program. Design in programming is often thought of as having to take the shape of formal notation, such as pseudocode or flowcharts, however they are not very accessible to blind learners, and it is important to provide forms which are tailored to supporting the particular learners involved. As I was working with blind and partially sighted learners, a braille representation seemed most appropriate. I placed the braille representation of each instruction on magnetic strips, which could be arranged on the design board to produce designs. The wording used was based on the way in which code is read out in Code Jumper, and provided another way of representing the concepts, although in this form they are not actually executable.

The ways in which the learners employed the design board demonstrated that they were able to move between and apply concepts to different representations. They could take an existing design and turn it into a Code Jumper program, and also design programs themselves, before turning them into Code Jumper programs. When working with the design board, the learners also demonstrated that Code Jumper had become an integral part of their sense of programming concepts. For example, some learners needed to hold a relevant pod when working out how to approach a particular part of the design. Additionally, when deciding which instructions to add next, they would often make reference to the form of representation provided in Code Jumper. For instance, they may say they needed to add a loop pod next when working on the design. This suggests that the learners were drawing on *perezhivaniya* from other experiences working with Code Jumper, and the occasional need to reach out and hold the pods potentially indicating that these experiences were internally re-enacted, imagining the form the Code Jumper version of the program would take as they were designing it.

There is a close correspondence between the representation of sequence in Code Jumper and the design board, as they both feature a set of instructions which are executed in the order they appear. The correspondence between the representations of repetition is not as close, as the design board does not feature a physical loop, however it still encloses the instructions which are repeated. When it comes to selection, the representation provided by Code Jumper takes on a totally different form to what is seen in text-based representations. This is because Code Jumper provides a physical representation of branching, in which the instructions literally

split into two branches, making the flow of control explicit. On the other hand, text-based representations depict selection in a linear fashion, which requires lines to be skipped. Taking into consideration the unique nature of selection as represented in Code Jumper, I decided against implementing an analogue with the design board. Despite not having to a tool to facilitate the design of programs featuring selection, the learners seemed to have no problems building such programs. This reinforces the suggestion that the design of selection in Code Jumper lends itself to the development of a strong sense of the concept.

Overall, the design board certainly seemed to facilitate the planning process, and enable learners to overcome the challenges which were observed in the pilot study. It helped them to develop their sense of sequence, by reinforcing the importance of order and enabling counting. Counting as a tool was an important aspect of this process and will be explored further in the following section. As time went on, planning became part of how the learners thought about programming and as such the design board was less important to them. Planning took on other forms, such as explaining how they were going to approach a problem verbally. For example, in Activity 2, David and Gregg started by discussing how they were going to approach the problem. Gregg said, “the first one is finger snap”, identifying the first sound in the program and David added, “and it’s twice”, pointing out the number of repetitions. Following this, Gregg started planning, “so we’re going to have to add a loop pod for the finger snap and the shutting door one once”. David added, “I think is has to be a nested loop because you have the finger snaps in one loop, and you have the shutting door in...” and Gregg finished the sentence with, “another loop”.

Interrelationship Between Representations

This study has demonstrated physical representations can be a powerful tool for blind and partially sighted learners, enabling them to overcome some of the barriers that screen-based representations impose. The data has also shown that physical representations not only shape, but also become a part of the learner’s sense of a concept. Even though the tool forms a key part of sense, this does not mean that learners cannot apply their sense to different tools, as the learners were able to readily switch between Code Jumper and the design board.

There is an implication in the literature, relating to learning programming with block-based languages, that they are an easier form of programming which facilitates the transition to text-based languages (Bau, Gray, Kelleher, Sheldon, & Turbak, 2017). This implies that their role is to introduce programming, and that they are not serious tools in their own right. This is a potentially dangerous perspective, as it could convey the impression that it is not ‘proper’ programming. Additionally, the use of the term transition seems to imply that block-based languages are temporary scaffolds (Weintrop, Hansen, Harlow, & Franklin, 2018), which are to be taken away once the concepts have been appropriated and block-based – or by analogy physical programming tools – are replaced by the text-based language that are associated with ‘proper’ programming. The interactions of the learners with the different representations available in the activities in this study indicate that particular representations are not replaced or superseded by others; they remain a core part of sense. It would seem more appropriate to think of the use of different representations as enriching sense, rather than in terms of replacement.

Different programming tools do have different strengths, and the programming challenges that can be solved with Code Jumper certainly have limitations. This does not mean that a particular tool should be considered as more advanced than another, rather that they have different affordances that makes them more suitable for a certain type of problem. Additionally, the form a language takes, be it physical, block or text-based should not denote its sophistication. Given the observed benefits of physical programming for blind and partially sighted learners, it would seem sensible to develop additional physical programming tools with different affordances to complement Code Jumper. This would provide these learners with a choice of representations to draw from, when they wish to tackle problems which cannot be solved using Code Jumper given its affordances. Such a tool could focus on other aspects of programming, such as handling inputs and working with subroutines.

9.2.1.2 Gestures and the Spoken Word

The learners also employed private speech as a tool in multiple ways throughout the intervention. As discussed in Chapter 5, gestures, and exploratory procedures for oneself are considered as manifestations of private speech for the purposes of this study. As discussed in the previous section, many learners created a looping gesture

when talking about repetition, almost re-enacting the representation in Code Jumper. Additionally, the contour following exploratory procedure (Lederman & Klatzky, 1987) acted as an important tool for the learners. It played an important role in the development of a sense of the order of execution for the blind learners. These exploratory procedures went on to become part of the social structuring of the activity. What started as private speech, became a stimulus means that could be employed as a sense-making activity for other students. Indeed, as the sessions developed and new concepts were introduced, there were cases in which students did not use the procedure spontaneously and needed to be prompted.

Counting was another important tool, which was expressed using a mixture of gestures and voice. It seemed to help the learners to orient themselves within a sequence of instructions, enabling them to locate a specific instruction. From my participation in the various activities, I had observed students adopting this strategy and could therefore offer it as a stimulus-means for others. For example, in Activity 3, Steven was turning a program design into a Code Jumper program, and while doing this he repeatedly struggled to identify the correct pod to set within his Code Jumper sequence. I facilitated the introduction of counting as a stimulus-means for Steven, by referring to instructions by their numerical position in a sequence. Steven then started using a counting gesture, first by pausing briefly on each pod and later by counting out loud. This seemed to be particularly helpful for him and there were occasions later on when he encountered similar challenges, and I suggested he employ counting once again. For most of the learners, externally perceivable counting was more in evidence when new concepts were introduced. Although the external manifestations of counting decreased, this does not mean that the tool was not present for the learners, as they may have been utilising inner speech.

Interestingly, the external signs of counting often returned when learners were introduced to new concepts. This could suggest that the enactment of the tool had a greater supportive effect in the learning process, and as such the learners employed this form of the tool when they encountered problems that challenged their current conceptions. This ties in with Vygotsky's assertion that private speech may reappear as task difficulty increases (Fernihough & Fradley, 2005).

As previously discussed, text and block-based programming languages are challenging for blind and partially sighted learners to work with, even with the aid of a screen reader, as there is a lack of a global view of the program. Physical languages go some way to addressing this challenge. However, while they are a step forward, blind and partially sighted learners may still face challenges when navigating their way around a physical program. This study has demonstrated that contour following and counting are effective techniques which emerged in the learners, and were transformed into stimulus-means to enable the successful navigation of physical programs.

9.2.2 Stimulus-End

9.2.2.1 Problem Design

For Vygotsky consciousness consists of two elements, affect and intellect, (Wertsch, 1985) and both play vital roles in the learning process. Accordingly, it is important to take affect into consideration during the analysis of *perezhivaniya* (Mahn & John-Steiner, 2002). When reflecting on the activities employed as the stimulus-end in the pilot study, I came to the conclusion that some of them may not have been as motivating as they could have been, as they were aimed at younger learners. For example, many activities featured nursery rhymes. As a result, I redesigned many of the activities for the main study in line with the design-based research approach. For example, I employed amusing limericks for the sequence activities and instrumental extracts from pop songs for nested loops. These activities were particularly engaging for the learners. Their engagement was manifested in a number of forms, including laughter, saying how much they liked the output of their program, or being disappointed when the activity ended. During these activities their motivation seemed to increase, as did their confidence. They seemed more assured of their actions when building programs, and were not discouraged when they encountered challenges.

An example can be found in Steven's negotiations of Activity 3. He initially displayed a lack of confidence when he was trying to find the correct sound for the next instruction in a limerick, saying, "oh dear, this is a bit of a difficult task for me to do, isn't it?". However, when he tried playing the program with the wrong sound, he realised that it sounded funny and laughed. This gave him the motivation to carry

on and locate the correct sound. The pattern of not locating the correct sound, finding the result amusing, and then persevering occurred a couple more times during this activity. The proceeding two activities also featured limericks and his confidence in building sequences continued to build.

As the emotions that learners experience during an activity form a core part of the associated *perezhivanie* and therefore the resulting sense, it can be concluded that these activities will have a significant influence on the shape of the learners' sense, and form an integral part of sense itself. It has been suggested that when tackling a new problem, the experience is refracted through existing *perezhivaniya*. Therefore, it could be argued, that in addition to refracting aspects of the concepts, it is also the case that aspects of their affective performance are also recalled. While it has been reported that students did seem to associate these experiences with positive feelings of success, amusement and growing confidence, it is important to note that the data collected focused mainly on the cognitive aspects of sense, and were not directed towards capturing external expressions of affect. This limits the conclusions that can be drawn from the data relating to affect, and this may be an area to be developed in future research.

9.3 The Role of Transitional Theories

9.3.1 Revisiting Misconceptions

The idea of misconceptions has always permeated computing education, however there is currently renewed attention to the notion that novice programmers hold misconceptions (Sorva, 2012, 2018; Swidan et al., 2018). Such research has played an important role in increasing our understanding of the learning processes associated with programming. It has enabled us to identify potential challenges in the process of learning to program and begin to develop strategies for addressing them. However, in Chapter 3 reservations about the use of the term 'misconceptions' were discussed and the term 'transitional theory' (originally employed by Papert (1980) was offered as an alternative. I will now revisit this discussion, summarising the main reservations and examining them in light of the data gathered from the intervention.

9.3.2 *Relationship Between the Expert and the Novice*

In their critique of state of misconception research at the time, Smith et al. (1993) suggested that the literature sought to identify how learners' conceptions conflicted with those of experts in the field, with the view that they must be replaced. Such framing of expert and novices is also in evidence in programming education literature. Kolikant and Mussai (2008) investigated learners' conceptions of program correctness and portrayed a program with any kind of error as totally incorrect, as that would be how an expert would look at it. Smith et al. (1993) criticised this kind of binary view of the distinction between the novice and the expert, as it implies that the conceptions of novices are replaced by expert concepts. Such a view is incompatible with constructivist theories of learning, which sees conceptions as being continually refined through experiences and are never replaced. It would perhaps be more appropriate to view the conceptions of novices and experts as being on a continuum.

When the literature discusses 'expert' conceptions of a topic, it seems to be referring to the external, culturally defined meaning. As such it could be argued that, from this perspective, someone that is considered an expert in a field would express their sense of a concept in a manner which closely matches the culturally defined meaning. In contrast the way a novice expresses their sense of a concept may differ considerably from the culturally defined meaning, at least at first glance. Looking at the *perezhivaniya* of my learners has enabled me to further reflect on my choice to adopt transitional theory in place of misconception. When Steven first started working with loops, he would conflate the number of sounds produced with the number of repetitions required. This conception served him well while he was working with simple loops containing a single instruction; however as he encountered loops which contained multiple instructions, he realised that his theory did not always work. As a result, his theory evolved, drawing upon *perezhivaniya* related to mathematics, to identify the multiplicative relationship between the number of repetitions, the number of instructions and the total number of sounds produced. This demonstrates that sense emerges and continually evolves through the interaction of *perezhivaniya*; it cannot be said that there is an abrupt switch between the sense of a novice and that of an expert. The sense of a novice will evolve over time and be refined and through this process their external expressions may seem to

match the meaning more closely. Of course, individual sense of any individual, be they expert or novice, will always be unique due to the specific combination of *perezhivaniya* through which it formed, but the data from this study indicates that aspects of the learners' sense of programming did approximate to the culturally shared meaning over the course of the activities.

When the expert is discussed in literature, it usually seems to refer to professionals in the field, for example professional programmers. However, I would argue that the role of experts in relation to novices in the learning process is more dynamic than that. From a Vygotskian perspective, the expert could be seen as the 'more knowledgeable other' which would often be the teacher, but could equally be another learner (Sentance et al., 2019). When working in pairs or groups, the role of the 'more knowledgeable other' may switch, even within the same activity. One learner may be more slightly more confident with a certain concept than the other learner, but not necessarily to the level of a professional programmer. Acting as the 'more knowledgeable other', a learner facilitates an experience which forms a *perezhivanie* for the other learner and further refines their sense. Although their sense may have been refined through this process, it is unlikely to suddenly reflect that of a professional programmer. It is a dynamic process which facilitates the evolution of sense.

The dynamic nature of the expert in the learning process was demonstrated during Activity 18, in which Sarah and Adam were building a program that featured two loops in a sequence. Although they both had built programs which featured instructions following a loop before, they had not created one with another loop connected in sequence. Adam had successfully added the first loop, but was unsure of the syntax required to add a second loop following it. Sarah used a gesture to indicate to Adam where to connect the additional loop. It appears that Sarah drew upon the *perezhivaniya* which shaped her sense of repetition and sequence, refining them further to meet the needs of this new scenario. For this moment, Sarah became the expert or more knowledgeable other, facilitating the further refinement of Adam's sense of sequence and repetition, enabling him to make the same connection she had. Although Sarah's sense of these concepts is unlikely to have closely reflected that of a professional programmer at that point in time, the state of her

sense enabled her to support Adam in the refinement of his sense to accomplish the given task.

9.3.3 Role of Experiences Across Subject Domains

Another area in the misconception literature which Smith et al. (1993) identified as problematic, was how the role of experiences outside the subject domain are treated. Computing education literature seems to have a tendency to identify such experiences as being the source of many of what they refer to as misconceptions. It has also been suggested that metaphors should be employed with caution in programming education as there are usually many incompatibilities between the concept and the metaphor (Swidan et al., 2018). The use of a box metaphor to explain how variables work is often cited as a particularly problematic example, as there are so many differences between the concepts. However, drawing upon *perezhivaniya* from a variety of sources is an important part of the learning process and the development of a sense of a concept. This study has demonstrated the importance of drawing on *perezhivaniya* across subject domains in the development of their sense of programming.

The learners in this study came into the sessions with *perezhivaniya* relating to other disciplines and experiences outside formal education, and they drew upon many of these during the development of their sense of programming. Experiences of mathematics seemed to be particularly helpful to many learners. When first working with nested loops, some developed a theory that the relationship between the two loops was additive. However, after testing their theory they realised that it did not quite match the results they observed, so they refined it and made the connection to multiplication. Additionally, some of them also made the connection to their knowledge of variables from algebra when developing their sense of the concept.

It is understandable that learners made connections between programming and mathematics, considering the number of cross overs between the disciplines. However, learners also drew upon *perezhivaniya* from further afield when developing their sense of programming concepts. When David was introduced to the selection pod, he likened it to a junction in a railway track before being told what selection was, drawing on his knowledge of railways. Additionally, when Steven was working with loops, he likened a loop to a beat and described the number of

repetitions as the beats. Apparently, he was drawing upon his *perezhivaniya* relating to music when developing his sense of repetition. It could be argued that the musical nature of many of the Code Jumper activities may have encouraged this connection.

Rather than being wary of experiences from other domains, the data collected during this project suggests that we should embrace them as a core part of developing one's sense of a concept. This study confirms the view of Smith et al. (1993), that we should recognise that the learners will encounter situations when these comparisons break down, and treat them as learning opportunities. In fact, it may be beneficial to present the learners with problems that may not be entirely compatible with their current sense of the concept, therefore prompting the further refinement of their sense, as happened with Steven when he was conflating the number of repetitions and the number of sounds produced.

9.3.4 Representations of Programming

This study has also demonstrated that some of what have been identified in the literature as misconceptions may not be directly related to the concept itself, but rather the representation. As discussed earlier in this chapter, the direction of assignment and threading are cited as concepts which can be problematic for learners. However, this was not the case for Code Jumper in this study, and this could potentially be due to the way in which Code Jumper embodies those concepts. It is therefore possible that some transitional theories stem from design of the representation rather than the nature of the concept itself.

9.3.5 Transitional Theories

The examples given above highlight the need to move beyond the framing of misconceptions as deficits in novices, which need to be replaced by expert conceptions. This study has demonstrated how an individual's sense of a concept is shaped and refined through their *perezhivaniya* and is not simply replaced. We also need to consider the use of the term misconception itself, as I would argue that for many it implies a deficit in the learner. This is demonstrated in Sorva's definition of misconceptions as "understandings that are deficient or inadequate for many practical programming contexts" (Sorva, 2013, p. 85). When some of the learners developed the theory that the relationship between loops in a nested loop was

additive, it was not a deficit, it was the result of logical reasoning. They took the number of repetitions from each loop and added them together, which provided them with an initial theory which they could test. Upon testing their theory, they discovered that it needed to be refined and made the connection to multiplication. The initial theory was an essential part of the learning process, as it provided them with a starting point for discussions and experimentation to identify the limitations, and therefore further refine them, providing further backing for the use of the term transitional theory in place of misconception.

For Vygotsky the meaning of a concept which would be recognised by experts forms a part of an individual's sense of that concept. The meaning is internalised into an individual's sense through *perezhivaniya*, resulting from interactions with their environment. As each individual's sense of a concept is unique, we require a means to discuss our sense of a concept with others. Transitional theories provide this means, enabling learners to discuss a concept even though their individual senses are different. Through these discussions with others, an individual's transitional theory relating to a concept will be refined and continue to evolve. Additionally, as sense is constantly evolving in response to new *perezhivaniya*, even the sense of an expert cannot be seen as fixed and is in fact transitional.

9.4 The Value of Perezhivanie

9.4.1 Introduction

In undertaking this research, I sought a theoretical lens which would embrace the perspective that both intellect and affect are fundamental to the process of learning, and should not be treated as totally independent factors. Additionally, I required an approach which would value the unique ways that individuals may perceive a concept. Vygotsky made it clear that the intellectual side of our consciousness should not be separated from the affective side (Wertz, 1985), and he also made the distinction between sense as the unique way an individual perceives a concept and the meaning which is external and culturally defined (Kravtsova, 2017).

Perezhivanie, a Russian word which Vygotsky utilized in his cultural-historical theory, met these requirements. It is usually translated as 'an experience' or 'emotional experience', but these translations may not adequately capture his view

of perezhivanie as the unity of experience and the processing of it. From this perspective, sense can be thought of as emerging from the interaction of various perezhivaniya, and viewing a learning experience as a perezhivanie allowed me to consider the role that numerous factors play in the development of an individual's sense.

The emotional experience [perezhivanie] arising from any situation or from any aspect of his [sic] environment, determines what kind of influence this situation or this environment will have on the child. Therefore, it is not any of the factors in themselves (if taken without reference to the child) which determines how they will influence the future course of his [sic] development, but the same factors refracted through the prism of the child's emotional experience [perezhivanie] (Vygotsky, 1994, p. 339).

In his work, Vygotsky also considered how a researcher can gain a window into the development of sense through the interaction of perezhivaniya using his double stimulation method. This approach utilizes two types of stimuli; a problem that the learners are asked to solve, which he described as the stimulus-end, and the tools the learners use to solve the problem, described as the stimuli-means. In the double stimulation method, it is important not to see the stimuli as fixed, as would be the case in traditional controlled experiments, as learners may introduce their own novel stimuli-means. With the focus on learning to program, both the Code Jumper physical programming language and the design board were central stimuli-means, but not the only ones, as the students' individual and shared strategies, as well as their interactions with the researcher, also contributed to their sense-making activities. That is to say, taking perezhivanie as the unit of analysis allows a focus on how the student's individual sense and emotional experience of programming are inextricably linked with, and formed through, the collective experience of the classroom environment.

9.4.2 A Personal Example

Perezhivanie can be exemplified through an exploration of my own personal journey in learning to program. I first encountered programming when I started my undergraduate degree. The module featured lectures which introduced abstract principles of programming, along with seminars which included abstract programming challenges. This approach to learning programming did not work for me and left me feeling that I was simply not capable of learning to program. The sense that emerged from the interaction of these perezhivaniya was dominated by affect. The thought of programming conjured feelings of inadequacy. Soon after I switched from a pure computer science degree to a programme which focused on information technology more broadly, along with education. This enabled me to explore the more creative applications of technology. Through this exploration I rediscovered programming when I wanted to add interactivity to my multimedia projects, although I was not consciously trying to learn programming concepts. During this period my sense of programming continued to evolve through the interaction of new and existing perezhivaniya. I did not have a sense of myself as a programmer, more as someone who knew enough to write basic scripts to achieve simple goals. As an ICT teacher, I taught multimedia units which involved the creation of interactive multimedia products. Looking back, I can see that my sense of programming may have impacted on the students' own sense as I did not convey to them that what they were doing was programming.

As I began to teach GCSE Computing and attended training events, I encountered people who did not view the kind of programming I had been doing up till that point as 'proper' programming. The resulting perezhivaniya fed into my sense of programming and further reinforced a feeling of inadequacy. However, I found that I was able to easily apply the principles I had learned while working with multimedia to other languages such as Python. While the multimedia programming was still a core part of my sense, it was further enriched by experiences with other tools. At this stage I began to see myself as someone who could program, but not a programmer. I felt that the route I had taken to learn how to program somehow made my understanding less sophisticated than that of people who learnt using more traditional means. Although this feeling has become less pronounced with time, it is still a part of my sense of programming and always will be. This highlights the importance of considering how tools are framed to learners. My collective

experience of the research activities carried out during this study has been refracted through the prism of my existing perezhivaniya. The process of exploring the learners' multiple expressions of sense contributed to an increased awareness of the value of my previous programming experience, and observing their responses to their struggles and their successes further convinced me of the importance of seeing intellect and emotion as part of the same whole. This was illustrated when examining Steven's journey, as early on he stated, "it's a bit difficult for me", seemingly drawing on the affect aspects of previous perezhivaniya. This changed as he developed an increasing awareness of evolution in his sense as he commented "I think I'm starting to get it" progressing to "that's amazing!".

9.4.3 Reflecting on This Study

In the process of designing the stimuli-ends for this study, the choice of perezhivanie as a theoretical lens, led me to seek to ensure that the individuality of the learners was respected, and valuing the fact that they would be employing their hands to see, rather than their eyes. Given my own learning trajectory, it was important to me to facilitate inclusive perezhivaniya, which would elicit a positive sense of programming concepts both in terms of intellect and affect. To achieve this end, I sought to offer a social and material environment which, in interaction with the student's personal characteristics, would contribute to their awarenesses of the programming concepts in question, whilst also recognising that a unique perezhivanie would be elicited in each student.

Through the examination of learners' expressions of sense during this study, it was possible to gain an insight into the way that the stimuli-ends and stimuli-means contributed to, and also became part of, the students' developing sense of key programming constructs. For example, the contour following exploratory procedure was an important stimulus-means, which enabled the learners to navigate and explore programs. It acted on multiple levels, as a tool for navigating programs, but also as an expression of their sense of sequence. More evidence to suggest that the stimuli-means that learners employed not only influenced their sense of a concept, but in fact become an integral part of it, can be seen in the way the physical form that loops take in Code Jumper influenced the gestures that learners produced when talking about repetition. There were also occasions, when learners needed to hold a

loop pod in order to answer a question about repetition, that the tool seemed to have become part of the thinking process. It became apparent that, although it may be useful to distinguish between the stimulus-end and stimuli-means for research design purposes, from the perspective of perezhivanie they are both part of the thinking process. Therefore a core part of the resulting sense, at which point the distinction between them becomes meaningless – the programs constructed by the students becoming as much expressions of their senses as the tools that enabled their construction.

This research also highlighted the important role that existing perezhivaniya play in the processing of new perezhivaniya, and that they form an important part of the sense that emerges as a result. This was illustrated when the learners were introduced to nested loops and initially drew upon their existing perezhivaniya relating to addition, when considering the relationship between the inner and outer loop. They developed a theory that considered the relationship as additive. After some investigation the learners' theory evolved to recognise the multiplicative relationship between the loops, drawing upon their existing perezhivaniya relating to multiplication.

The use of perezhivanie shows a great deal of potential as a theoretical lens through which to examine learning processes in computing education and in education more broadly. The following sections will potential implications for different aspects of education.

9.4.4 Implications for Pedagogy

Awareness of the role perezhivaniya plays in the development of sense could lead to a more inclusive learning experience for students. This research has highlighted a number of aspects, which influence the shape of sense and perhaps also become part of sense. For instance, if tools become a part of a student's sense of a programming concept they cannot really be considered as temporary scaffolds, useful on the student's journey, but which can be removed once a student seems to have grasped the learning objective. Tools should be expected to constitute a core part of the resulting sense. That is not to imply that an individual's sense will forever be constrained by the nature of the tools offered as stimuli means. Evidence from this study indicates that while material tools, and the expressions they informed, became

a core part of the student's sense, these tools and expressions could also be applied to other means of expressing programming concepts, and that these can in fact further enrich sense.

Another aspect that plays an important role in the development of sense is existing *perezhivaniya*. It has been suggested that drawing on experiences from other domains can be a potential source of what are usually referred to as misconceptions (Sorva, 2018). However, as existing *perezhivaniya* are inevitably part of any sense making process, attempts to disallow the important role they play in the process will send the message that they are not valued, and this too may become an integral part of their sense. This also leads to the implication that the framing of misconceptions should also be reconsidered given the negative connotations associated with them.

The role of affect in the development of sense should not be underestimated. It is not just about motivation, the feelings we experience during the development of our sense of a concept becomes part of sense itself and hence they may be re-experienced whenever we draw upon that sense again. This has implications for activity design and also the framing of tools. Tangible learning tools are often framed as being a stepping-stone to more abstract forms of expression, implying that the sense developed using them is somehow less sophisticated. This is a particularly important consideration for teachers of visually impaired learners, as they employ tangible tools extensively and the framing of such tools could have a significant impact on their development of sense. This research has also demonstrated that learners can express their sense of a concept in numerous ways, such as gestures, and there may be a tendency not to see these as valuable as verbal expressions or even as sophisticated. Ignoring the value of these less recognised forms of expression could be seen as a lost learning opportunity.

9.4.5 Implications for Professional Development

As previously noted, providing alternative ways of engaging with programming concepts can lead to a more inclusive learning experience. Therefore, it is recommended that teachers be given the opportunity to develop their pedagogical skills with a variety of programming tools, particularly physical representations.

As noted earlier, an individual's sense of a concept is always under development, and this includes that of experts. This also applies to teachers; their unique sense of a concept will have emerged from the interaction of various *perezhivaniya*. It is recommended that teachers reflect on their own individual sense of different concepts and how it may impact their delivery. For example, considering affect in the development of sense. If teachers were encouraged to reflect on the affective side of their own sense of a concept, they may develop a greater awareness of how aspects of the negative and positive experiences that accompanied their own conceptual development, are re-felt as they teach the concept in question. This could, perhaps, influence their delivery and the resulting sense of each individual learner.

9.4.6 Implications for Resource and Tool Development

For the purposes of inclusion, it is important to design learning environments which include multiple ways of engaging in programming, including physical. However, when framing these tools, we need to be careful not to create a hierarchy, or to frame physical tools as less sophisticated, as this could have negative implications on affect and hence the resulting sense. It would be better to think of different programming tools in terms of affordances rather than in sophistication. For example, some programming tools may be more suitable for solving certain types of problem. If learners are given access to a variety of different forms of programming tool with different affordances, they could select the tool that they are both comfortable using and is appropriate for solving the given problem.

Physical programming languages have been shown to be an accessible modality for visually impaired learners to engage with programming concepts. However, there are currently few physical programming languages available, and most of these are not fully accessible. Therefore, it is recommended that additional physical languages are developed with different affordances to expand the choice available to learners with visual impairments when solving different types of problem.

Chapter 10. Conclusion

This thesis set out to address the dual concern of understanding the processes by which blind and partially sighted learners develop their sense of programming concepts, while building learning ecologies that would support engagement in these processes. Design-based research was employed, to facilitate the development and refinement of effective learning ecologies, with a Vygotskian lens being taken to the analysis of resulting learning processes. This concluding chapter will start by exploring how each of the following research questions have been addressed, and close with a discussion of the contribution that this thesis has made to field of computing education.

1. How do blind, and partially sighted learners express their sense of sequence, threading, repetition, selection and variables?
2. What do these expressions reveal about the learning processes by which sense of programming develops?
3. How do the design structures embedded in the learning ecology support these learning processes?

10.1 Research Question 1

Through carrying out the planned intervention and carefully analysing video recordings of the sessions, it was possible to identify a variety of means of expressing a sense of programming. Many of these have not been considered or examined in existing programming education literature. The following sub-sections outline the forms of expression that were observed relating to the concepts under investigation.

10.1.1 Sequence and Threading

The contour following exploratory procedure, identified by Lederman & Klatzky (1987), was important for the blind learners, as it enabled them to explore the structure of sequences and locate specific instructions through touch as a primary sense. The partially sighted learners, on the other hand, were less reliant on contour following, as they were able to supplement touch with some vision. When the learners were working with sequences in Code Jumper for the first time, they

initially struggled to build logical sequences, which ties in with the findings of Swidan (2018). They found this task less challenging when they began using counting in the form of gestures and voice to locate specific instructions within their design or Code Jumper program. As their confidence grew, their external manifestations of counting reduced. However, they often increased when the level of challenge increased.

A sense of sequence was also expressed in other ways, for example through the use of a linear gesture and by explaining the concept verbally. Additionally, there were expressions of affect associated with the sequence and threading problems they were tackling. For instance, many learners expressed engagement and a feeling of success, and this seemed to increase as their sense of sequence developed. Some learners also initially expressed a lack of confidence in the early activities, and this gradually transformed into expressions of confidence.

10.1.2 Repetition

The learners expressed their sense of repetition in numerous ways, from the identification of the need for a loop to solve a problem to the use of gestures to represent the concept. Although they found the syntax of the physical loop in Code Jumper challenging at times, they were still able to demonstrate a clear understanding of repetition. The need for some learners to hold the loop pod when talking about repetition, coupled with the use of looping gestures, demonstrates that the physical representation facilitates the development of a conceptual understanding of repetition, even if they were not always sure of the syntax. This evidence counters Qian & Lehman's (Qian & Lehman, 2017) suggestion that there is usually a correlation between inadequate syntactic knowledge and a lack of conceptual knowledge.

Some expressions of repetition suggested that the learners were drawing upon *perezhivaniya* relating to other domains such as mathematics and music. For instance, when developing their sense of nested loops, some learners initially believed that the relationship between the number of repetitions in both loops was additive. However, through experimentation they realised that the relationship was in fact multiplicative.

As was the case with sequence and threading, affect played an important role in the development of a sense of repetition. The learners' expressions of confidence and a feeling of success increased as time went on. For some learners, there were expressions of a lack of confidence when the level of challenge increased before the expressions of confidence returned once again.

10.1.3 Selection and Variables

A sense of selection was expressed in a variety of ways, including the choosing of appropriate conditions for a given problem, and by successfully predicting the outcome of conditions. The learners did not have any major difficulties when working with the selection pod. Before being introduced to the concept of selection, David made the connection between the selection pod and a junction in a railway track. Seemingly, the design of the selection pod lends itself to the development of a sense of selection. This could be due to the way in which Code Jumper makes the flow of control in selection clear in its layout, whereas in text-based languages, the linear representation of selection involves the skipping of lines of code.

On the other hand, David was initially surprised that both branches which emerge from the selection pod do not play simultaneously. Given that the branches were placed right next to each other in similar manner to threads, this is a reasonable assumption to make. It indicates that although the representation of selection in Code Jumper may facilitate the development of a sense of the concept, it may also present other challenges that are not experienced when working with text-based languages.

When they were first introduced to variables in Code Jumper, some learners drew upon their knowledge of variables in mathematics. In the past, research has indicated that the discrepancies between the concepts across the two domains is problematic for the development of an understanding of concept in computing (Sorva, 2018). However, the learners did not encounter challenges when working with variables in Code Jumper, and were able to assign values in addition to identifying when variables were needed. Assignment is often cited as being a concept which confuses novice programmers. It is possible that this is due to the application of the = operator in computing in a different capacity than it is used in mathematics. Code Jumper

avoids this confusion, by using the physical action of plugging a value into a variable without involving the use of the = operator.

10.2 Research Question 2

Although it is not possible to gain a direct understanding of how an individual's sense of a concept develops, through examining their expressions of sense we can gain an insight into the learning process. When the learners are in the learning situations which I designed, they bring *perezhivaniya* with them and as they are challenged, they re-enact and reprocess what they already know, they are not simply recalling a memory. As they re-enact these *perezhivaniya* they may simulate them using gestures or even want to hold the tools that are a part of their sense.

Additionally, they will also recall how they felt during those experiences, whether it was hard or whether they felt good about themselves, and these feelings form a part of the re-enactment. Therefore, as the learners bring their previous experiences into the present, they reprocess them which then becomes a part of their sense.

From the analysis of the learners' expressions of a sense of programming, we have established that tools, transitional theories and affect all play an important role in the development of sense. These aspects are processed in the *perezhivaniya* and not only influence the shape of an individual's sense but form a core part of it. This is not a one-way process, these aspects of sense can be externally manifested, enabling social interaction, resulting in further refinement of the sense of all learners involved. Therefore, each of these aspects are present both within sense but also within the learning ecology. The culturally defined meaning of a concept is another important part of the learning ecology and also, according to Vygotsky (1987), forms a small part of an individual's sense. These relationships are illustrated in Figure 30.

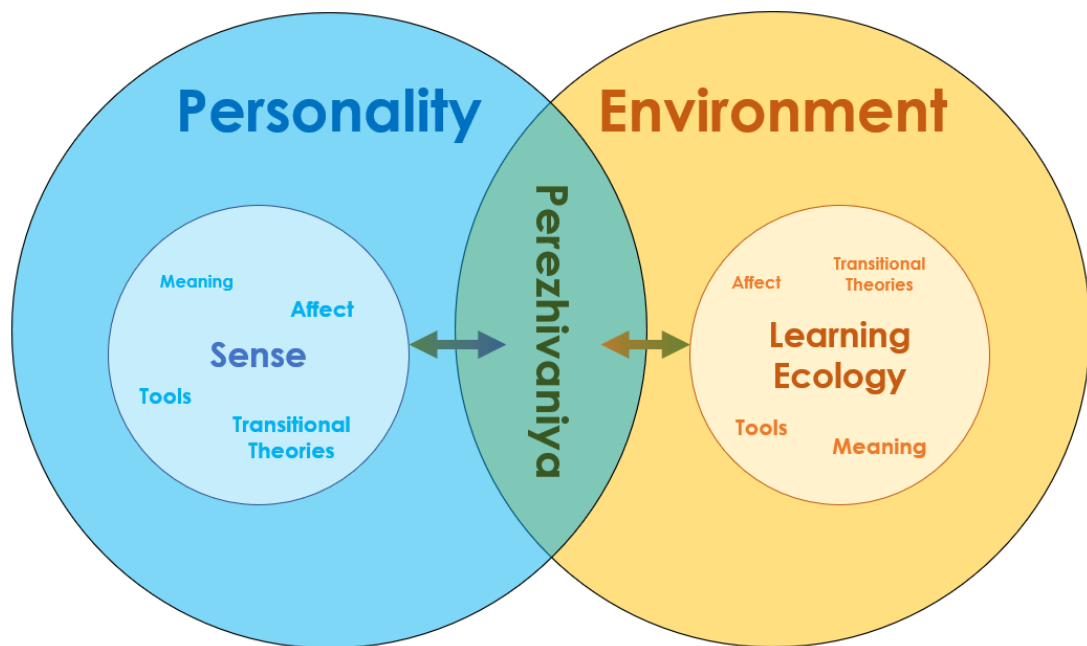


FIGURE 30: ASPECTS OF SENSE

The following sections will explore tools, affect and transitional theories as aspects of sense and the role they play in the learning process.

10.2.1 Tools

This study has demonstrated that tools play a significant role in the development of sense and in fact become part of it. This includes physical programming representations such as Code Jumper and the design board. This is highlighted in the gestures which were influenced by the form of Code Jumper, and in the need for some learners to hold a particular pod in order to answer a question regarding that concept. These actions suggest an internal re-enactment and reprocessing of experiences with these tools. Additionally, the gestures suggest the embodiment of Code Jumper. This demonstrates that representations form an integral part of sense, and therefore different representations enable learners to develop different aspects of their sense and connecting between the representations offers another means to reflect on the nuances of the concepts in question. As such, tools are not replaced or superseded, rather they enrich sense and their presence continues in their tangible absence.

Additional tools which learners introduced themselves included the use of counting and contour following. For many learners, counting using gestures and voice was an

important tool when developing a sense of sequence and became part of it. The act of counting often became internal as learners became more comfortable with a concept, however it usually returned when new concepts were introduced. Counting gestures were also employed in order to keep track of the number of repetitions in a loop, therefore counting is also an aspect of a sense of repetition for some learners. The contour following exploratory procedure was an essential tool for blind learners, enabling them to gain an understanding of the flow of control in a program. It was particularly important when working with sequences and formed a part of their sense of the concept.

10.2.2 Affect

We have seen how the affect associated with an activity forms a core part of the resulting *perezhivanie* and therefore feeds directly into sense. As discussed earlier, the tools employed in the development of sense of a concept can be re-enacted internally when the learner encounters the concept in the future. In a similar manner, the affect associated with a concept can also be re-enacted, impacting on the learner's motivation and confidence.

A key influence on affect in learning situations is the problem design. If problems are designed in such a way that the learner can relate to it and it engages them, they are more likely to have a positive relationship with that concept going forward. The role that affect plays in sense is also continually evolving as the learner engages in new activities.

10.2.3 Transitional Theories

In contemporary computing education literature, there is an implication that the expert and novice programmer are binary positions. It has been suggested that novices often hold misconceptions and that these often stem from learners drawing upon experiences outside the domain of computing. Additionally, there is an implication that misconceptions are replaced with correct conceptions, or those of expert programmers. In this study we have challenged this perspective of the learning process by demonstrating that the conceptions of learners are continually evolving through the interaction of *perezhivaniya*, and as such even an expert's conceptions will continue to change. Transitional theory is an alternative term which

was proposed by Papert, and it is suggested that it be employed in place of misconceptions, as it embraces the dynamic nature of conceptions. Based on the results of this study, we suggest that transitional theories are formed by and are part of sense.

Vygotsky stressed the importance of social interactions in the learning process and transitional theories can be seen as playing an important role in facilitating this process. Transitional theories enable learners to discuss a concept even though their individual sense will be unique. Through these interactions, the transitional theories of the learners will continue to evolve. From this perspective we can reframe the role of the expert in the learning process as the more knowledgeable other, rather than a professional. In this configuration, the roles of the expert and novice are dynamic. In many situations the expert may be the teacher, but in other circumstances it could be another learner, who through the current state of their sense of a concept is able to facilitate the further development of the sense of another learner.

10.3 Research Question 3

Through the use of the design-based research method outlined in Chapter 5, I developed and refined a microworld which facilitated and revealed the learning processes described in the previous section. The microworld incorporates the stimulus-end and the stimuli-means. The stimulus-end is the problem the learners are being asked to solve and the stimuli-means are the tools which the learners employ to facilitate the process of solving the problem. One of the stimuli-means chosen was the programming language; Code Jumper was chosen as it is inclusive of blind and partially sighted learners. Additionally, it enabled learners to obtain a global overview of the program through its physicality, something that would not be possible with a text-based language coupled with a screen reader. A pilot study was conducted, which found that some learners found it challenging to implement their ideas directly through Code Jumper. Therefore, an additional stimulus-means was added to main study, which facilitated the planning and design process for the learners. This stimulus-means took the form of the design board, which featured magnetic braille pieces which could be arranged to produce designs.

The stimuli-ends took the form of the activities which made up the curriculum. The development of these started before the research covered in this thesis. I developed the original Code Jumper curriculum, and was actively involved in its evaluation (Morrison et al., 2019). This evaluation did find that some learners struggled with problems which relied on being able to differentiate between the pitch of different musical notes. For this reason, the curriculums for the pilot and main studies avoided such tasks where possible.

The activities employed in the pilot study were largely based on the original Code Jumper curriculum, which was designed for learners aged between 7 and 11. Upon reflection, it was decided that it was necessary to adapt some of the activities to make them more engaging for the participants in the main study who were aged between 11 and 15. This was particularly important, because as the previous section identified, activity design influences affect, which is a key aspect of sense.

In addition to the stimuli-means that I designed and introduced, there were also stimulus-means which were originated in the learners. As discussed in Chapter 5, it is important, when employing Vygotsky's method of double stimulation, to recognise that learners may introduce their own novel stimulus-means. The novel stimuli-means which were introduced by the learners in this study included the use of the contour following exploratory procedure, in addition to counting using gestures and voice. As teacher-researcher, I recognised that these novel stimuli-means supported the learners that employed them when working with sequences in Code Jumper. I therefore chose to introduce these as stimuli-means for the other learners. Contour following was most beneficial to the blind learners and counting was beneficial to all the participants in their sense making process.

10.4 Contribution

This research makes a number of contributions to the field of computing education. Firstly, it has demonstrated that the tools, in the Vygotskian sense, employed by blind and partially sighted learners when learning to program become an integral part of their sense of programming. As a result, the nature of these tools has an influence on the way in which the learners express their sense of programming. For instance, the physicality of the tools employed in this research was conveyed in the

representational gestures and exploratory procedures that the learners were observed to use, and this thesis demonstrates the need to recognise these forms of expression which have not been given sufficient recognition in past research.

Secondly, this research has not only highlighted the important role previous experiences play in the learning process, but also provided further insights into the nature of this role. The data demonstrates that when learners tackled the problems presented in the teaching intervention, the experiences were refracted through existing *perezhivaniya*, enabling them to be re-enacted and re-processed. This process enabled the learners to review their developing knowledge of programming and to refine their ideas.

A third contribution is the demonstration of the role transitional theories play in the learning journey. It is suggested that the use of ‘misconception’ as a term can imply a deficit in the learner and lead to an oversimplification of the learning process. It frames understanding of programming concepts, in terms of expert and novice conceptions, as binary positions. Transitional theories recognise that our sense of a concept is always developing as we assimilate new experiences. They also recognise that they play an important role in enabling the communication and transformation of our ideas through our interaction with our environment. Therefore, a move away from the emphasis on misconceptions to transitional theories could enable researchers to enhance their understanding of progression in programming.

Finally, this research has demonstrated the value of *perezhivanie* as a theoretical lens through which to examine the unique learning processes of individuals. This novel approach to the investigation of sense views both intellect and affect as being fundamental to the process of learning, and should not be considered as independent factors. Using *perezhivanie* as the unit of analysis, facilitated the exploration of how the learner’s individual sense and emotional experience of programming are interwoven and formed through, the *perezhivaniya* emerging from the classroom environment. *Perezhivanie* shows a great deal of promise as a theoretical lens to explore learning processes in computing education beyond programming and in other disciplines. Using *perezhivanie* as a theoretical lens has highlighted the importance of recognising embodied ways of accessing, engaging with, and understanding programming for the inclusion of blind and partially sighted learners.

In the light of this finding, it would be interesting for further research to identify the diversity of different embodiments of programming, as the recognition of these may be fundamental to the inclusion of other groups of learners that have traditionally experienced barriers to programming education.

10.5 Implications

This research has a number of implications for various members of the computing education community in addition to other researchers. The implications for each group will be explored in turn.

10.5.1 Computing Pedagogy

Educators having an awareness of the role *perezhivaniya* play in the development of sense, can inform their pedagogy in such a way that leads to a more inclusive learning experience. This awareness includes an appreciation of the different factors which not only shape, but also become an integral part of sense. One such factor is tools, which have been demonstrated to become a core part of sense. Therefore, they should not be thought of as temporary scaffolds, and additional tools can be seen as enriching sense, rather than replacing existing tools.

Existing *perezhivaniya* is another factor that shapes an individual's sense of a concept. Drawing on experiences from other domains is an essential part of the learning process, and not recognising the important role they play could lead learners to believe that these experiences are not valued, further shaping their sense. This has implications for affect, which is another important factor in the development of sense. The feelings we experience during the development of sense, form a core part of it and therefore these feelings may be re-lived when we draw upon that sense in the future.

This research has demonstrated that, with the aid of a physical programming language, blind and partially sighted children can successfully learn key programming concepts without needing to use text-based languages and screen readers. Considering this, it is important to ensure that different forms of programming are portrayed as valid or legitimized. Framing physical and visual programming tools as being a gateway to text-based languages could result in

learners feeling as if they are not doing ‘proper’ programming. This could have a lasting impact on a learners’ sense of themselves as a programmer.

In terms of the professional development of computing teachers, it is important for them to be aware that their own sense of each programming concept is unique and will have emerged from the interplay between numerous *perezhivaniya*. Reflecting on the affective side of their sense may enable them to consider how their positive and negative experiences may influence their delivery, and the resulting sense of their learners.

10.5.2 Curriculum Designers

In order to create learning environments which are as inclusive as possible, it is important to provide multiple ways of engaging in programming. We should also recognise that learners can express their sense of programming in a variety of forms, which should all be valued. As such, care should be taken not link forms of expression with the level of progress. Additionally, programming tools should not be seen in terms of a hierarchy, giving this impression may have a negative impact on the affective side of a learner’s sense. It would be better to see them in terms of having different affordances which make them more suitable for solving certain types of problem over others. For instance, Code Jumper has certain affordances which make it particularly suitable for solving certain types of problem. It could be beneficial to incorporate a variety of programming tools within curricula, to provide learners with a toolbox from which they can select a tool with which they are comfortable and is suitable to solve the problem in question.

The value of employing an approach that views developing conceptualisations as transitional theories, rather than misconceptions, has been argued to enable a richer understanding of the process of learning programming. This also suggests that structuring a curriculum around identifying and addressing misconceptions is likely to have limitations. Particularly as certain ‘misconceptions’ have been shown to have their origins in the type of programming representation, and may not be observed when other forms of representation are employed. Furthermore, others, while they may have a limited domain of validity, are still important for the learning process and with progress involving their refinement rather than replacement.

10.5.3 Tool Designers

Physical programming languages such as Code Jumper have been shown to be effective in making programming accessible to learners with visual impairments. Additionally, as discussed previously, programming tools have different affordances. As such, it would be beneficial for a broader range of physical programming languages to be made available for educators and curriculum designers. For example, a new tool could focus on subroutines and input handling, as Code Jumper does not have these affordances.

The physical representation provided by Code Jumper has been shown to be particularly effective in supporting the development of a sense of certain concepts, such as threading, selection and variables. The nature of the physical representation of these concepts should be taken into consideration in the development of new tools. Additionally, further research is required to identify effective methods of representation for other programming concepts.

10.6 Further Research

This study has enabled the identification of a number of insights into the development of a sense of programming concepts among novice programmers with visual impairments. Considering both the blind and partially sighted learners expressed their sense in the form of gestures, it could be argued that sighted learners are also likely to produce gestures when working with a physical language. Additionally, despite blind and sighted learners relying on different senses, they may still employ similar learning processes to refine their transitional theories. Therefore, I recommend additional research be conducted into how these insights apply to a wider population with a broader range of participants, employing a range of different programming tools.

References

- Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference? In *Constructivism: Uses and Perspectives in Education* (pp. 85–94). Geneva.
- Alderson-Day, B., & Fernyhough, C. (2015). Inner speech: Development, cognitive functions, phenomenology, and neurobiology. *Psychological Bulletin*, *141*(5), 931–965. <https://doi.org/10.1037/bul0000021>
- Alfieri, L., Brooks, P. J., Aldrich, N. J., & Tenenbaum, H. R. (2011). Does discovery-based instruction enhance learning? *Journal of Educational Psychology*, *103*(1), 1–18. <https://doi.org/10.1037/a0021017>
- Amineh, R. J., & Asl, H. D. (2015). Review of constructivism and social constructivism. *Journal of Social Sciences, Literature and Languages*, *1*(1), 9–16.
- Armstrong, D. F., Stokoe, W. C., & Wilcox, S. E. (1995). *Gesture and the nature of language*. Cambridge: Cambridge University Press.
- Asakawa, C., & Leporini, B. (2009). Screen readers. In C. Stephanidis (Ed.), *The Universal Access Handbook* (pp. 28–1 to 28–17). Boca Raton: CRC Press.
- Australian Institute for Teaching and School Leadership. (2013). *Literature review: Student-centred schools make the difference*. Melbourne, Australia.
- Baker, C. M., Milne, L. R., & Ladner, R. E. (2015). StructJumper. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15* (pp. 3043–3052). New York, New York, USA: ACM Press. <https://doi.org/10.1145/2702123.2702589>
- Ball, D. L. (2000). Working on the inside: Using one's own practice as a site for studying teaching and learning. In A. E. Kelly & R. A. Lesh (Eds.), *Handbook of Research Design in Mathematics and Science Education* (pp. 365–402). Mahwah, New Jersey: Lawrence Erlbaum Associates.
- Ball, S. j. (1990). Self-doubt and soft data: social and technical trajectories in ethnographic fieldwork. *International Journal of Qualitative Studies in Education*, *3*(2), 157–171. <https://doi.org/10.1080/0951839900030204>
- Barron, B. (2007). Video as a tool to advance understanding of learning and development in peer, family and other informal learning contexts. In Ricki Goldman, R. D. Pea, B. Barron, & S. J. Derry (Eds.), *Video Research in the Learning Sciences* (pp. 159–187). Mahwah, N.J.: Lawrence Erlbaum Associates.
- Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM*, *60*(6), 72–80. <https://doi.org/10.1145/3015455>

- Beck, K. (1999). *Extreme programming explained: Embrace change. XP Series*. Boston, MA: Addison-Wesley Longman Publishing Co.
<https://doi.org/10.1136/adc.2005.076794>
- Berger, P. L., & Luckmann, T. (1966). *The social construction of reality*. London: Penguin Books.
- Berila, E. P., Butterfield, L. H., & Murr, M. J. (1976). Tactual reading of political maps by blind students: A videomatic behavioral analysis. *Journal of Special Education, 10*(3), 265–276.
- Berk, L. (1992). Children's private speech: An overview of theory and the status of research. In R. M. Diaz & L. Berk (Eds.), *Private speech: From social interaction to self-regulation* (pp. 17–53). Hillsdale, NJ: Erlbaum, Inc.
- Bernstein, R. (1983). *Beyond objectivism and relativism: Science, hermeneutics, and praxis*. Philadelphia: University of Pennsylvania Press.
- Bevan, J., Werner, L., & McDowell, C. (2002). Guidelines for the use of pair programming in a freshman programming class. In *Proceedings of the 15th Conference on Software Engineering Education and Training* (p. 100). Washington, DC, USA.
- Bigham, J. P., Aller, M. B., Brudvik, J. T., Leung, J. O., Yazzolino, L. a., & Ladner, R. E. (2008). Inspiring blind high school students to pursue computer science with instant messaging chatbots. *ACM SIGCSE Bulletin, 40*(1), 449.
<https://doi.org/10.1145/1352322.1352287>
- Blackwell, A. F. (2002). What is programming? In *Proc. of 14th Workshop of the Psychology of Programming Interest Group* (pp. 204–218). London.
- Blaikie, N. (2010). *Designing social research* (2nd Edition). Cambridge: Polity Press.
- Blair, A. (2017). *Learning mathematics through inquiry: the relationship between induction and deduction in inquiry maths*. King's College London, University of London.
- Blunden, A. (2016). Translating perezhivanie into English. *Mind, Culture, and Activity, 23*(4), 274–283. <https://doi.org/10.1080/10749039.2016.1186193>
- Bradley, B. A., & Reinking, D. (2011a). Enhancing research and practice in early childhood through formative and design experiments. *Early Child Development and Care, 181*(3), 305–319. <https://doi.org/10.1080/03004430903357894>
- Bradley, B. A., & Reinking, D. (2011b). Revisiting the connection between research and practice using design research and formative experiments. In N. Duke & M. Mallette (Eds.), *Literacy research methodologies, 2nd Edition* (2nd Edition, pp. 188–212). New York: Guilford Press.
- Burr, V. (2015). *Social constructionism*. London: Taylor and Francis.

- Calder, M., Cohen, R. F., Lanzoni, J., Landry, N., Skaff, J., Calder, M., ... Skaff, J. (2007). Teaching data structures to students who are blind. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education - ITiCSE '07* (Vol. 39, p. 87). New York, New York, USA: ACM Press. <https://doi.org/10.1145/1268784.1268811>
- Calder, M., Cohen, R. F., Lanzoni, J., & Xu, Y. (2006). PLUMB: An interface for users who are blind to display, create, and modify graphs. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility - Assets '06* (p. 263). New York, New York, USA: ACM Press. <https://doi.org/10.1145/1168987.1169046>
- Capovilla, D., Krugel, J., & Hubwieser, P. (2013). Teaching algorithmic thinking using haptic models for visually impaired students. In *2013 Learning and Teaching in Computing and Engineering* (pp. 167–171). IEEE. <https://doi.org/10.1109/LaTiCE.2013.14>
- Cartmill, E. A., Beilock, S., & Goldin-Meadow, S. (2012). A word in the hand: action, gesture and mental representation in humans and non-human primates. *Philosophical Transactions: Biological Sciences*, 367, 129–143. <https://doi.org/10.2307/41433521>
- Charlie McDowell, Brian Hanks, & Linda Werner. (2003). Experimenting with pair programming in the classroom. *Proceeding-Innovation and Technology in Computer Science Education(ITiCSE 2003)*, 35, 60–64. <https://doi.org/10.1145/961290.961531>
- Cheong, C. (2010). Coding without sight: Teaching object-oriented java programming to a blind student. In *Eighth Annual Hawaii International Conference on Education* (pp. 1–12). Hawaii International Conference on Education.
- Chu, M., & Kita, S. (2008). Spontaneous gestures during mental rotation tasks: Insights into the microdevelopment of the motor strategy. *Journal of Experimental Psychology*, 137(4), 706–723. <https://doi.org/10.1037/a0013157>
- Chu, M., & Kita, S. (2011). The nature of gestures' beneficial role in spatial problem solving. *Journal of Experimental Psychology: General*, 140(1), 102–116. <https://doi.org/10.1037/a0021790>
- Cobb, P., Confrey, J., Disessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher*, 32(1), 9–13.
- Cohen, L., Manion, L., & Morrison, K. (2017). *Research methods in education* (8th ed.). Oxon: Routledge.
- Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design research: Theoretical and methodological issues. *Journal of the Learning Sciences*, 13(1), 15–42. https://doi.org/10.1207/s15327809jls1301_2
- Crotty, M. (1998). *The foundations of social research*. London: Sage Publications.

- Crowder, E. M. (1996). Gestures at work in sense-making science talk. *The Journal of the Learning Sciences*, 5(3), 173–208.
- Cutts, Q., Connor, R., Michaelson, G., & Donaldson, P. (2014). Code or (not code): Separating formal and natural language in CS education. In *Proceeding of 9th Workshop in Primary and Secondary Computing Education* (pp. 20–28). Association for Computing Machinery (ACM).
<https://doi.org/10.1145/2670757.2670780>
- Daniels, H., Cole, M., & Wertsch, J. V. (2007). Editors' introduction. In *The Cambridge Companion to Vygotsky* (pp. 1–18). Cambridge University Press.
<https://doi.org/10.1017/CCOL0521831040.001>
- Department for Education. (2014). *The national curriculum in England - Framework document*. Department for Education.
- Dorsey, R., Chung, H. P., & Howard, A. (2014). Developing the capabilities of blind and visually impaired youth to build and program robots. In *28th Annual International Technology and Persons with Disabilities Conference*. San Diego: California State University, Northridge.
- Du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1), 57–73. <https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9>
- Edwards, L. D. (1991). Children's learning in a computer microworld for transformation geometry. *Source: Journal for Research in Mathematics Education*, 22(2), 122–137.
- Ellis, V. (2010). Studying the process of change: The double stimulation strategy in teacher education research. In V. Ellis, A. Edwards, & P. Smagorinsky (Eds.), *Cultural-Historical Perspectives on Teacher Education and Development: Learning Teaching* (pp. 95–114). Abingdon UK: Routledge.
<https://doi.org/10.4324/9780203860106>
- Engeström, Y. (2011). From design experiments to formative interventions. *Theory & Psychology*, 21(5), 598–628. <https://doi.org/10.1177/0959354311419252>
- Engle, R. A., Conant, F. R., & Greeno, J. G. (2007). Progressive refinement of hypotheses in video-supported research. In R. Goldman, R. Pea, B. Barron, & S. J. Derry (Eds.), *Video research in the learning sciences* (pp. 239–254). Mahwah, N.J: Erlbaum.
- Erickson, F. (2006). Definition and analysis of data from videotape: Some research procedures and their rationales. In *Handbook of Complementary Methods in Education Research* (pp. 177–205).
- Exton, C. (2002). Constructivism and program comprehension strategies. In *Proceedings - IEEE Workshop on Program Comprehension* (pp. 281–284). IEEE Computer Society. <https://doi.org/10.1109/WPC.2002.1021349>
- Federici, S., & Stern, L. (2011). A constructionist approach to computer science. In

Global Learn 2011. Melbourne, Australia: Association for the Advancement of Computing in Education.

- Fernyhough, C., & Fradley, E. (2005). Private speech on an executive task: relations with task difficulty and task performance. *Cognitive Development, 20*, 103–120. <https://doi.org/10.1016/j.cogdev.2004.11.002>
- Francioni, J. M., & Smith, A. C. (2002). Computer science accessibility for students with visual disabilities. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education - SIGCSE '02* (Vol. 34, p. 91). New York, New York, USA: ACM Press. <https://doi.org/10.1145/563340.563372>
- Franklin, D., Skifstad, G., Rolock, R., Mehrotra, I., Ding, V., Hansen, A., ... Harlow, D. (2017). Using upper-elementary student performance to understand conceptual sequencing in a blocks-based curriculum. In *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE* (pp. 231–236). Association for Computing Machinery. <https://doi.org/10.1145/3017680.3017760>
- Goldin-Meadow, S. (2014). How gesture works to change our minds. *Trends in Neuroscience and Education, 3*(1), 4–6. <https://doi.org/10.1016/J.TINE.2014.01.002>
- Goldin-Meadow, S., & Beilock, S. L. (2010). Action's influence on thought: The case of gesture. *Perspectives on Psychological Science, 5*(6), 664–674. <https://doi.org/10.1177/1745691610388764>
- Gray, D. E. (2014). *Doing research in the real world* (2nd ed.). London: Sage Publications.
- Grover, S., & Basu, S. (2017). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and Boolean logic. In *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE* (pp. 267–272). Association for Computing Machinery. <https://doi.org/10.1145/3017680.3017723>
- Gujberova, M., & Kalas, I. (2013). Designing productive gradations of tasks in primary programming education. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education on - WiPSE '13* (pp. 108–117). New York, New York, USA: Association for Computing Machinery. <https://doi.org/10.1145/2532748.2532750>
- Hadwen-Bennett, A., & Thieme, A. (2018). Inclusivity of computing education for learners with visuali. In *CHI2018 Workshop on: Emerging Opportunities in Inclusive Education for People with Visual Impairments*. Montréal, Canada.
- Hasan, R. (1992). Speech genre, semiotic mediation and the development of higher mental functions. *Language Sciences, 14*(4), 489–528. [https://doi.org/10.1016/0388-0001\(92\)90027-C](https://doi.org/10.1016/0388-0001(92)90027-C)
- Healy, L., Hassan, S., & Fernandes, A. A. (2011). The role of gestures in the mathematical practices of those who do not see with their eyes. *Education*

Studies in Mathematics, 77(2–3), 157–174. <https://doi.org/10.1007/s>

- Healy, L., Ramos, E. B., Fernandes, S. H. A. A., & Peixoto, J. L. B. (2016). Mathematics in the hands of deaf learners and blind learners: Visual–gestural–somatic means of doing and expressing mathematics (pp. 141–162). Springer, Cham. https://doi.org/10.1007/978-3-319-14511-2_8
- Horn, M. S., & Jacob, R. J. K. (2007a). Designing tangible programming languages for classroom use. In *Proceedings of the 1st international conference on Tangible and embedded interaction - TEI '07* (p. 159). New York, New York, USA: ACM Press. <https://doi.org/10.1145/1226969.1227003>
- Horn, M. S., & Jacob, R. J. K. (2007b). Tangible programming in the classroom with tern. In *CHI '07 extended abstracts on Human factors in computing systems - CHI '07* (p. 1965). New York, New York, USA: ACM Press. <https://doi.org/10.1145/1240866.1240933>
- Hoyles, C., & Noss, R. (1987). Children working in a structured logo environment: from doing to understanding. *Recherches En Didactique Des Mathématiques*, 8(1.2), 131–174.
- Hoyles, C., & Noss, R. (1992). A pedagogy for mathematical microworlds. *Educational Studies in Mathematics*, 23(1), 31–57. <https://doi.org/10.1007/BF00302313>
- Iverson, J. M., & Goldin-Meadow, S. (2001). The resilience of gesture in talk: Gesture in blind speakers and listeners. *Developmental Science*, 4(4), 416–422. <https://doi.org/10.1111/1467-7687.00183>
- Jacob, E. (1997). Context and Cognition: Implications for educational innovators and anthropologists. *Anthropology & Education Quarterly*, 28(1), 3–21. <https://doi.org/10.1525/aeq.1997.28.1.3>
- Jansen, S. E. M., Bergmann Tiest, W. M., & Kappers, A. M. L. (2013). Identifying haptic exploratory procedures by analyzing hand dynamics and contact force. *IEEE Transactions on Haptics*, 6(4), 464–472. <https://doi.org/10.1109/TOH.2013.22>
- Jašková, Ľ., & Kaliaková, M. (2014). Programming microworlds for visually impaired pupils. In G. Futschek & C. Kynigos (Eds.), *Proceedings of the 3rd international constructionism conference*. Vienna.
- Jelec, A., & Jaworska, D. (2015). Thoughts on the table. Gesture as a tool for thinking in blind and visually impaired children. *Yearbook of the Poznan Linguistic Meeting*, 1(1), 73–88. <https://doi.org/10.2478/yplm-2014-0004>
- Jones, P. E. (2008). Language in cultural-historical perspective. In B. Van Oers, W. Wardekker, E. Elbers, & R. van der Veer (Eds.), *The Transformation of Learning* (pp. 76–99). Cambridge University Press.
- Kalagher, H., & Jones, S. S. (2011). Young children's haptic exploratory procedures. *Journal of Experimental Child Psychology*, 110(4), 592–602.

<https://doi.org/10.1016/J.JECP.2011.06.007>

Kane, S. K., & Bigham, J. P. (2014). Tracking @stemxcomet. In *Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE '14* (pp. 247–252). New York, New York, USA: ACM Press.
<https://doi.org/10.1145/2538862.2538975>

Kenton, A. (1980). Gesticulation and speech: Two aspects of the process of utterance. In M. R. Key (Ed.), *The Relationship of Verbal and Nonverbal Communication*. Great Britain: Moutoun Publishers.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2010). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Https://Doi.Org/10.1207/S15326985ep4102_1*, 41(2), 75–86.
https://doi.org/10.1207/S15326985EP4102_1

Klatzky, R. L., Lederman, S. J., & Mankinen, J. M. (2005). Visual and haptic exploratory procedures in children's judgments about tool function. *Infant Behavior and Development*, 28(3), 240–249.
<https://doi.org/10.1016/J.INFBEH.2005.05.002>

Kolikant, Y. B. D., & Mussai, M. (2008). “So my program doesn't run!” Definition, origins, and practical expressions of students' (mis)conceptions of correctness. *Computer Science Education*, 18(2), 135–151.
<https://doi.org/10.1080/08993400802156400>

Kölling, M. (2010). The Greenfoot programming environment. *ACM Transactions of Computing Education*, 10(4). <https://doi.org/10.1145/1868358.1868361>

Konecki, M. (2014). GUIDL as an aiding technology in programming education of visually impaired. *Journal of Computers*, 9(12), 2816–2821.
<https://doi.org/10.4304/jcp.9.12.2816-2821>

Koushik, V., & Lewis, C. (2016). An accessible blocks language. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS '16* (pp. 317–318). New York, New York, USA: ACM Press. <https://doi.org/10.1145/2982142.2982150>

Kozulin, A. (1995). The learning process: Vygotsky's theory in the mirror of its interpretations. *School Psychology International*, 16(2), 117–129.
<https://doi.org/10.1177/0143034395162003>

Kravtsova, E. (2017). The sense and the meaning of cultural-historical theory of L. S. Vygotsky. *Revue Internationale Du CRIRES: Innover Dans La Tradition de Vygotsky*, 4(1), 35–47.

Lederman, S. J., & Klatzky, R. L. (1987). Hand movements: A window into haptic object recognition. *Cognitive Psychology*, 19(3), 342–368.
[https://doi.org/10.1016/0010-0285\(87\)90008-9](https://doi.org/10.1016/0010-0285(87)90008-9)

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... Werner, L.

- (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1), 32.
<https://doi.org/10.1145/1929887.1929902>
- Lee, J. (2008). Gesture and private speech in second language acquisition. *Studies in Second Language Acquisition*, 30(2), 169–190.
<https://doi.org/10.1017/S0272263108080303>
- Lewis, C. (2014). Work in progress report: Nonvisual visual programming. In *Proceedings of the 25th Psychology of Programming Annual Conference (PPIG 2014)*.
- Lister, R., Seppälä, O., Simon, B., Thomas, L., Adams, E. S., Fitzgerald, S., ... Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. In *Working group reports from ITiCSE on Innovation and technology in computer science education - ITiCSE-WGR '04* (Vol. 36, p. 119). New York, New York, USA: ACM Press.
<https://doi.org/10.1145/1044550.1041673>
- Ludi, S. (2013). Robotics programming tools for blind students. In *28th Annual International Technology and Persons with Disabilities Conference*. San Diego: California State University, Northridge.
- Ludi, S. (2015). Position paper: Towards making block-based programming accessible for blind users. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)* (pp. 67–69). IEEE.
<https://doi.org/10.1109/BLOCKS.2015.7369005>
- Ludi, S., Ellis, L., & Jordan, S. (2014). An accessible robotics programming environment for visually impaired users. In *Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility - ASSETS '14* (pp. 237–238). New York, New York, USA: ACM Press.
<https://doi.org/10.1145/2661334.2661385>
- Ludi, S., & Reichlmayr, T. (2011). The use of robotics to promote computing to pre-college students with visual impairments. *ACM Transactions on Computing Education*, 11(3), 1–20. <https://doi.org/10.1145/2037276.2037284>
- Lynch, M. (1998). Towards a constructivist genealogy of social constructivism. In I. Velody & R. Williams (Eds.), *The Politics of Constructionism* (pp. 13–32). London: SAGE Publications.
- Mackenzie, N., & Veresov, N. (2013). How drawing can support writing acquisition: Text construction in early writing from a Vygotskian perspective. *Australasian Journal of Early Childhood*, 38(4), 22–29.
<https://doi.org/10.1177/183693911303800404>
- Mahn, H. (2012). Vygotsky's analysis of children's meaning making processes. *International Journal of Educational Psychology*, 1(2), 100–126.
<https://doi.org/10.4471/ijep.2012.07>
- Mahn, H., & John-Steiner, V. (2002). The gift of confidence: A Vygotskian view of emotions. In G. Wells & G. Claxton (Eds.), *Learning for Life in the 21st*

Century : Sociocultural Perspectives on the Future of Education (pp. 46–58). Oxford: John Wiley & Sons.

- Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education*, *10*(4), 1–15. <https://doi.org/10.1145/1868358.1868363>
- Margulieux, L., Denny, P., Cunningham, K., Deutsch, M., & Shapiro, B. R. (2021). When wrong is right : The instructional power of multiple conceptions. In *International Computing Education Research conference (ICER '21), August 16-19, 2021*.
- Mason, J. (2002). *Qualitative Researching* (2nd ed.). Thousand Oaks, CA: Sage Publications.
- Mattosinho Bernardes, M. E. (2018). The appropriation of culture and psychological development: contributions of historical-cultural psychology for child education. *Revista de Estudios y Experiencias En Educación*, *17*(35), 77–89. <https://doi.org/10.21703/rexe.20181735mattosinho5>
- McLaughlin, B. (1981). *An experimental comparison of discovery and didactic computerized instructional strategies in the learning of computer programming*. The Catholic University of America, Ann Arbor.
- McNeill, D. (1992). *Hand and mind: what gestures reveal about thought*. London: Cambridge University Press.
- Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011). Habits of programming in Scratch (pp. 168–172).
- Miller, R. (2011). *Vygotsky in perspective*. Cambridge: Cambridge University Press.
- Morrison, C., Villar, N., Hadwen-Bennett, A., Regan, T., Cletheroe, D., Thieme, A., ... Sentance, S. (2019). Physical programming for blind and low vision children at scale. *Human-Computer Interaction*. <https://doi.org/10.1080/07370024.2019.1621175>
- Morrison, C., Villar, N., Thieme, A., Ashktorab, Z., Taysom, E., Salandin, O., ... Zhang, H. (2018). Torino: A tangible programming language inclusive of children with visual disabilities. *Human Computer Interaction*.
- Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers*. London: Springer. <https://doi.org/10.1007/978-94-009-1696-8>
- Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2017). Reviewing the affordances of tangible programming languages: Implications for design and practice. In *2017 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1811–1816). IEEE. <https://doi.org/10.1109/EDUCON.2017.7943096>
- Papazafiropulos, N., Fanucci, L., Leporini, B., Pelagatti, S., & Roncella, R. (2016). Haptic models of arrays through 3D printing for computer science education.

Computers Helping People with Special Needs, 491–498.
https://doi.org/10.1007/978-3-319-41264-1_67

- Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*. New York: Basic Books.
- Papert, S. (1987). Microworlds: Transforming education. In R. Walter Lawler & M. Yazdani (Eds.), *Artificial Intelligence and Education: Learning environments and tutoring systems* (pp. 79–94). Norwood, NJ: Intellect Books.
- Papert, S. (1992). *The children's machine: Rethinking school in the age of the computer*. New York: Basic Books.
- Papert, S., & Harel, I. (1991). Situating constructionism. In S. Papert & I. Harel (Eds.), *Constructionism*. Norwood, NJ: Ablex Publishing Corporation.
- Penuel, W. R., & Wertsch, J. V. (1995). Vygotsky and identity formation: A sociocultural approach. *Educational Psychologist*, 30(2), 83–92.
https://doi.org/10.1207/s15326985ep3002_5
- Piaget, J. (1982). *The essential Piaget*. (H. E. Gruber & J. J. Vonèche, Eds.). London: Routledge & Kegan Paul.
- Popper, K. (1963). *Conjectures and refutations: The growth of scientific knowledge*. London, UK: Routledge.
- Powell, A. B., Francisco, J. M., & Maher, C. A. (2003). An analytical model for studying the development of learners' mathematical ideas and reasoning using videotape data. *Journal of Mathematical Behavior*, 22(4), 405–435.
<https://doi.org/10.1016/j.jmathb.2003.09.002>
- Przybylla, M., & Romeike, R. (2014a). Key competences with physical computing. In *Proceedings of Key Competencies in Informatics and ICT 2014* (pp. 351–361). Potsdam.
- Przybylla, M., & Romeike, R. (2014b). Physical computing and its scope - towards a constructionist computer science curriculum with physical computing. *Informatics in Education*, 13(2), 241–254.
<https://doi.org/10.15388/infedu.2014.05>
- Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A Literature Review. *ACM Transactions of Computing Education*, 18(1). <https://doi.org/10.1145/3077618>
- Rajich, V. (2002). Program comprehension as a learning process. In *Proceedings - 1st IEEE International Conference on Cognitive Informatics, ICCI 2002* (pp. 343–347). Institute of Electrical and Electronics Engineers Inc.
<https://doi.org/10.1109/COGINF.2002.1039316>
- Raman, T. V. (1996). Emacspeak---direct speech access. In *Proceedings of the second annual ACM conference on Assistive technologies - Assets '96* (pp. 32–36). New York: ACM Press. <https://doi.org/10.1145/228347.228354>

- Rosborough, A. (2014). Gesture, meaning-making, and embodiment: Second language learning in an elementary classroom. *Journal of Pedagogy*, 5(2), 227–250. <https://doi.org/10.2478/jped-2014-0011>
- Roth, W. M. (2001). Situating cognition. *Journal of the Learning Sciences*, 10(1–2), 27–61. https://doi.org/10.1207/S15327809JLS10-1-2_4
- Salleh, N., Mendes, E., & Grundy, J. (2011). Empirical studies of pair programming for CS / SE teaching in higher education : A systematic literature review. *IEEE Transactions on Software Engineering*, 37(4), 509–525.
- Sánchez, J., & Aguayo, F. (2006). APL: Audio programming language for blind learners. In K. Miesenberger, J. Klaus, W. L. Zagler, & A. I. Karshmer (Eds.), *Computers Helping People with Special Needs. ICCHP 2006. Lecture Notes in Computer Science* (4061st ed., pp. 1334–1341). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11788713_192
- Sandoval, W. A. (2013). Educational design research in the 21st Century. In R. Luckin, S. Puntambekar, P. Goodyear, B. Grabowski, J. Underwood, & N. Winters (Eds.), *Handbook of Design in Educational Technology*. Abingdon UK: Routledge. https://doi.org/10.1057/9780230593305_8
- Sannino, A., Engeström, Y., & Lemos, M. (2016). Formative interventions for expansive learning and transformative agency. *Journal of the Learning Sciences*, 25(4), 599–633. <https://doi.org/10.1080/10508406.2016.1204547>
- Sasso, B. A., & Morais, A. de. (2014). Egocentric speech in the works of Vygotsky and Piaget: educational implications and representations by teachers. *International Journal of Humanities and Social Science*, 4(8), 133–143.
- Scherer, R., Siddiq, F., & SánchezViveros, B. (2020). A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Computers in Human Behavior*, 109. <https://doi.org/10.1016/j.chb.2020.106349>
- Schulte, C., Clear, T., Taherkhani, A., Busjahn, T., & Paterson, J. H. (2010). An introduction to program comprehension for computer science educators. In *Proceedings of the 2010 ITiCSE working group reports on Working group reports - ITiCSE-WGR '10* (p. 65). New York: ACM Press. <https://doi.org/10.1145/1971681.1971687>
- Scott, J., & Marshall, G. (2009). *Oxford Dictionary of Sociology* (4th ed.). Oxford: Oxford University Press.
- Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, 29(2–3), 136–176. <https://doi.org/10.1080/08993408.2019.1608781>
- Siegfried, R. M. (2006). Visual programming and the blind: The challenge and the opportunity. In *SIGCSE '06 Proceedings of the 37th SIGCSE technical symposium on Computer science education* (Vol. 38, pp. 275–278). Houston, Texas: ACM. <https://doi.org/10.1145/1124706.1121427>

- Smith, A. C., Francioni, J. M., & Matzek, S. D. (2000). A Java programming tool for students with visual disabilities. In *Proceedings of the fourth international ACM conference on Assistive technologies - Assets '00* (pp. 142–148). New York, New York, USA: ACM Press. <https://doi.org/10.1145/354324.354356>
- Smith, J. P., DiSessa, A. A., & Roschelle, J. (1993). Misconceptions reconceived: A constructivist analysis of knowledge in transition. *Http://Dx.Doi.Org/10.1207/S15327809jls0302_1*, 3(2), 115–163. https://doi.org/10.1207/S15327809JLS0302_1
- Sorva, J. (2012). *Visual program simulation in introductory programming education*. Aalto University.
- Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education (TOCE)*, 13(2), 31. <https://doi.org/10.1145/2483710.2483713>
- Sorva, J. (2018). Misconceptions and the beginner programmer. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer Science Education: Perspectives on Teaching and Learning in School* (pp. 171–205). London: Bloomsbury Publishing.
- Stefik, A., Alexander, R., Patterson, R., & Brown, J. (2007). WAD: A Feasibility study using the Wicked Audio Debugger. In *15th IEEE International Conference on Program Comprehension (ICPC '07)* (pp. 69–80). IEEE. <https://doi.org/10.1109/ICPC.2007.42>
- Stefik, A., Hundhausen, C., & Smith, D. (2011). On the design of an educational infrastructure for the blind and visually impaired in computer science. In *Proceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE '11* (p. 571). New York: ACM Press. <https://doi.org/10.1145/1953163.1953323>
- Stefik, A., & Siebert, S. (2013). An empirical investigation into programming language syntax. *ACM Transactions on Computing Education*, 13(4), 1–40. <https://doi.org/10.1145/2534973>
- Stefik, A., Siebert, S., Stefik, M., & Slattery, K. (2011). An empirical comparison of the accuracy rates of novices using the quorum, perl, and random programming languages. In *Proceedings of the 3rd ACM SIGPLAN workshop on Evaluation and usability of programming languages and tools - PLATEAU '11* (p. 3). New York: ACM Press. <https://doi.org/10.1145/2089155.2089159>
- Streeck, J. (2009a). *Gesturecraft: The manufacture of meaning*. Philadelphia: John Benjamins Publishing.
- Streeck, J. (2009b). Gestures: pragmatic aspects. In J. L. Mey (Ed.), *Concise Encyclopedia of Pragmatics*. Oxford: Elsevier.
- Subbotsky, E. (1996). Vygotsky's distinction between lower and higher mental functions and recent studies on infant cognitive development. *Journal of Russian & East European Psychology*, 34(2), 61–66.

<https://doi.org/10.2753/RPO1061-0405340261>

- Swanson, D., & Williams, J. (2014). Making abstract mathematics concrete in and out of school. *Educational Studies in Mathematics*, 86(2), 193–209.
<https://doi.org/10.1007/s10649-014-9536-4>
- Swidan, A., Hermans, F., & Smit, M. (2018). Programming misconceptions for school students. In *ICER '18: Proceedings of the 2018 ACM Conference on International Computing Education Research* (pp. 151–159). Espoo, Finland.
<https://doi.org/10.1145/3230977.3230995>
- Tenenberg, J., & Knobelsdorf, M. (2014). Out of our minds: A review of sociocultural cognition theory. *Computer Science Education*, 24(1), 1–24.
<https://doi.org/10.1080/08993408.2013.869396>
- The Design-Based Research Collective. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, 32(1), 5–8.
- The Royal Society. (2017). *After the reboot: computing education in UK schools*.
- Thieme, A., Morrison, C., Villar, N., Grayson, M., & Lindley, S. (2017). Enabling collaboration in learning computer programming inclusive of children with vision impairments. In *Proceedings of the 2017 Conference on Designing Interactive Systems - DIS '17* (pp. 739–752). New York: ACM Press.
<https://doi.org/10.1145/3064663.3064689>
- Tudge, J. (2012). Vygotsky, the zone of proximal development, and peer collaboration: Implications for classroom practice. In *Vygotsky and Education* (pp. 155–172). Cambridge: Cambridge University Press.
<https://doi.org/10.1017/cbo9781139173674.008>
- Ungar, S., Blades, M., & Spencer, C. (1995). Visually impaired children's strategies for memorising a map. *British Journal of Visual Impairment*, 13(1), 27–32.
- van der Veer, René. (2007). Vygotsky in context: 1900-1935. In *The Cambridge Companion to Vygotsky* (pp. 21–49). Cambridge University Press.
<https://doi.org/10.1017/CCOL0521831040.002>
- Van der Veer, Rene, & Valsiner, J. (1993). *Understanding Vygotsky: A quest for synthesis*. Oxford: Blackwell.
- Veerasamy, A. K., D'Souza, D., & Laakso, M.-J. (2016). Identifying novice student programming misconceptions and errors from summative assessments. *Journal of Educational Technology Systems*, 45(1), 50–73.
<https://doi.org/10.1177/0047239515627263>
- Velody, I., & Williams, R. (1998). *The politics of constructionism*. London: SAGE Publications.
- Veresov, N. (2016). Perezhivanie as a phenomenon and a concept: Questions on clarification and methodological meditations. *Cultural-Historical Psychology*,

12(3), 129–148. <https://doi.org/10.17759/chp.2016120308>

- Veresov, N., & Fleer, M. (2016). Perezhivanie as a theoretical concept for researching young children's development. *Mind, Culture, and Activity*, 23(4), 325–335. <https://doi.org/10.1080/10749039.2016.1186198>
- Vinter, A., Fernandes, V., Orlandi, O., & Morgan, P. (2012). Exploratory procedures of tactile images in visually impaired and blindfolded sighted children: How they relate to their consequent performance in drawing. *Research in Developmental Disabilities*, 33, 1819–1831. <https://doi.org/10.1016/j.ridd.2012.05.001>
- Vygotsky, L. S. (1978). *Mind in society*. (M. Cole, V. John-Steiner, S. Scribner, & E. Souberman, Eds.). Harvard, Massachusetts: Harvard University Press.
- Vygotsky, L. S. (1987). *The collected works of L.S. Vygotsky / Vol.1, Problems of general psychology*. (R. W. (Robert W. . Rieber & A. S. Carton, Eds.). book, New York ; London: Plenum.
- Vygotsky, L. S. (1994). The problem of the environment. In Rene Van der Veer & J. Valsiner (Eds.), *The Vygotsky Reader*. Oxford: Blackwell.
- Vygotsky, L. S. (1997). *The collected works of L.S. Vygotsky / Vol.3, Problems of the theory and history of psychology*. (R. W. (Robert W. . Rieber & J. L. Wollock, Eds.). book, New York ; London: Plenum.
- Waite, J., Curzon, P., Marsh, W., Sentance, S., & Hawden-Bennett, A. (2018). Abstraction in action: K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming. *International Journal Of Computer Science Education In Schools*, 2(1).
- Weintrop, D., Hansen, A. K., Harlow, D. B., & Franklin, D. (2018). Starting from scratch: Outcomes of early computer science learning experiences implications for what comes next. In *ICER 2018 - Proceedings of the 2018 ACM Conference on International Computing Education Research* (Vol. 18, pp. 142–150). Association for Computing Machinery, Inc. <https://doi.org/10.1145/3230977.3230988>
- Wertsch, J. V. (1985). *Vygotsky and the social formation of the mind*. London, England: Harvard University Press.
- Wertsch, J. V. (2007). Mediation. In *The Cambridge Companion to Vygotsky* (pp. 178–192). Cambridge: Cambridge University Press. <https://doi.org/10.1017/CCOL0521831040.008>
- Wertsch, J. V. (2000). Vygotsky's two minds on the nature of meaning. In C. D. Lee & P. Smagorinsky (Eds.), *Vygotskian Perspectives on Literary Research* (pp. 19–29). Cambridge: Cambridge University Press.
- Zavershneva, E. (2016). Vygotsky the unpublished: an overview of the personal archive (1912–1934). In A. Yasnitsky & R. Van der Veer (Eds.), *Revisionist revolution in Vygotsky studies* (pp. 94–126). London: Routledge.

Appendix 1 Pilot Study Ethical Approval

Research Ethics
Office

Franklin Wilkins Building
5.9 Waterloo Bridge Wing
Waterloo Road
London SE1 9NH
Telephone 020 7848 4020/4070/4077
reo@kcl.ac.uk



Alexander Hadwen-Bennet

22 February 2018

Dear Alexander

LRS-17/18-5607 - An Investigation of Exploratory Procedures used by Visually Impaired Children when Working with Physical Programming Languages

Thank you for submitting your application for the above project. I am pleased to inform you that your application has now be approved with the provisos indicated at the end of this letter. All changes must be made before data collection commences. The Committee does not need to see evidence of these changes, however supervisors are responsible for ensuring that students implement any requested changes before data collection commences.

Ethical approval has been granted for a period of **three years** from 22 February 2018. You will not be sent a reminder when your approval has lapsed and if you require an extension you should complete a modification request, details of which can be found here: <http://www.kcl.ac.uk/innovation/research/support/ethics/applications/modifications.aspx>

Please ensure that you follow the guidelines for good research practice as laid out in UKRIO's Code of Practice for research: <http://www.kcl.ac.uk/innovation/research/support/conduct/cop/index.aspx>

Any unforeseen ethical problems arising during the course of the project should be reported to the panel Chair, via the Research Ethics Office.

Please note that we may, for the purposes of audit, contact you to ascertain the status of your research.

We wish you every success with your research.

Yours sincerely,
Miss Annah Whyton

Senior Research Ethics Officer

For and on behalf of:
E&M Research Ethics Panel

Final Dual Review Decision: Approved with Provisos

Major Issues (will require substantial consideration by the applicant before approval can be granted)

Minor Issues related to application (the reviewer should identify the relevant section number before each comment)

Given that the individuals in the unit will be approaching the students, please discuss with your supervisor how possible pressure to participate will be mitigated (for example, an administrator from the team and not the faculty approaching students?)

Minor Issues related to recruitment documents

Information sheet: please make it clear to participants and their parents/guardians that they will not be disadvantaged in any way if they decide not to take part.

Please indicate where the workshop will take place and whether this will be conducted during school hours.

Remove reference to the ethics committee approval under the funding statement.

In the information sheet for students also explain that their data will be removed if they change their decision to participate.

Please ensure that a suitable way of communicating the information and consent forms is provided for visually impaired child participants. Please ensure that an adequate means of securing visually impaired child participants' consent is in place.

Advice and Comments (do not have to be adhered to, but may help to improve the research)

Appendix 2 Pilot Study Participant Information Sheet (Parent)

Version Number 22/02/18

INFORMATION SHEET FOR PARTICIPANTS

REC Reference Number: LRS-17/18-5607

YOU WILL BE GIVEN A COPY OF THIS INFORMATION SHEET



Title of study

An Investigation of Exploratory Procedures used by Visually Impaired Children when Working with Physical Programming Languages

Invitation Paragraph

I would like to invite your child to participate in this research project which forms part of my PhD research. You should only give permission for your child to participate if you want to; choosing not to take part will not disadvantage you or your child in anyway. Before you decide whether you want your child to take part, it is important for you to understand why the research is being done and what your child's participation will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask me if there is anything that is not clear or if you would like more information.

What is the purpose of the study?

Most children in primary school are currently taught how to program using visual programming languages that use puzzle-like pieces that can be snapped together. The visual nature of these languages makes them inaccessible to most visually impaired learners.

Physical programming languages are an alternative to visual languages, they use physical blocks that can be joined together to create programs. The aim of this study is to identify the ways in which learners develop an understanding of physical programs by exploring them with their hands.

In this study we will be using Torino, a physical programming language that is designed to be inclusive of learners with visual impairments. It can be used to build programs that produce music, stories and poetry.

Why has my child been invited to take part?

I am inviting visually impaired children aged between 7 and 11 to take part in this study.

Do I have to take part?

Participation is voluntary. You do not have to agree for your child take part. Your child will not be disadvantaged in any way if they do not participate. You should read this information sheet and if you have any questions you should ask the researcher.

What will happen to me if I take part?

If you decide to give permission for your child to take part, you will be given this information sheet to keep and will be asked to sign a consent form. Your child will be invited to participate in a one-hour coding workshop. During the workshop your child will work with another visually impaired child to solve challenges using the Torino physical programming language.

The workshop will be video recorded to enable analysis of the use of hands to explore physical programs. The video recordings will be stored securely and only be accessible to researchers directly involved in the study.

Even if you have decided to take part, you are still free to stop your child's participation at any time during the workshop and to have research data relating to them withdrawn without giving any reason up to 4 weeks after the workshop.

What are the possible benefits and risks of taking part?

The information I get from the study will help to develop an understanding of how visually impaired children learn to program with physical programming languages. If your child takes part in this study, they will get the opportunity to learn about coding using a physical programming language. Furthermore, I will provide you with a summary of a final report describing the main findings. The main disadvantage to taking part in the study is that your child will be donating an hour of their time to take part.

There are no foreseeable risks in participating in the study.

Will my taking part be kept confidential?

What is recorded during the workshop is strictly confidential and will be held securely. All data for analysis will be anonymised. In reporting on the research findings, I will not reveal the names of any participants or the organisation involved. At all times there will be no possibility of your child as an individual being linked with the data.

The UK Data Protection Act 1998 will apply to all information gathered within the workshops and held on password-locked computer files and locked cabinets within King's College London. No data will be accessed by anyone other than the researchers directly involved with this study; and anonymity of the material will be protected by using false names. No data will be able to be linked back to any individual taking part in the workshops.

How is the project being funded?

The researcher is funded by an EPSRC studentship.

What will happen to the results of the study?

I will produce a final report summarising the main findings, which will be sent to you. The research findings will form part of my PhD thesis. I also plan to disseminate the research findings through publication and conferences.

Who should I contact for further information?

If you have any questions or require more information about this study, please contact me using the following contact details:

Alex Hadwen-Bennett
School of Education, Communication and Society | King's College London
Telephone: 020 7848 3945
E-mail: alex.hadwen-bennett@kcl.ac.uk

What if I have further questions, or if something goes wrong?

If this study has harmed you in any way or if you wish to make a complaint about the conduct of the study you can contact King's College London using the details below for further advice and information:

Dr Sue Sentance
School of Education, Communication and Society, King's College London
Telephone: 020 7848 3117
E-mail: sue.sentance@kcl.ac.uk

Thank you for reading this information sheet and for considering taking part in this research.

Appendix 3 Pilot Study Participant Information Sheet (Child)

Version Number 22/02/18

INFORMATION SHEET FOR PARTICIPANTS

REC Reference Number: LRS-17/18-5607

YOU WILL BE GIVEN A COPY OF THIS INFORMATION SHEET



Title of study

An Investigation of Exploratory Procedures used by Visually Impaired Children when Working with Physical Programming Languages

Introduction

I'm inviting you to take part in this research project which is part of my course at university. You should only agree to take part if you want to, you are free to decide not to take part.

Before you decide if you want to take part, I need to tell you about the research. Please read the following information and discuss it with others if you wish. Ask me if you have any questions or want me to explain something to you.

What is the purpose of the study?

In school, most children are taught how to code by snapping together puzzle pieces on a computer screen. These systems can be hard for visually impaired children to use.

Another way of learning to program a computer is to use physical blocks that can be connected. I want to find out about how blind children use their hands to explore a physical program.

In this study we will be using Torino, which can be used to build programs that create music, stories and poetry.

Why have I been asked to take part?

I am inviting visually impaired children aged between 7 and 11 to take part in this study.

Do I have to take part?

Taking part is voluntary. You do not have to take part. You will not be disadvantaged in any way if you decide not to take part. You should read this information sheet and if you have any questions you should ask me.

What will happen to me if I take part?

- If you tell us that you want to take part, you will be given this information sheet to keep.
- You will be invited to take part in a 1-hour coding workshop. During the workshop you will work with another blind child to solve challenges using Torino.
- The workshop will be video recorded, so I can see how you use your hands when working with Torino.
- The video recordings will be stored safely, only the people working on this project will be able to watch them.
- If you decide to take part, you are still free to change your mind at any time up to 4 weeks after the workshop.

What are the possible benefits and risks of taking part?

- What I find out from the study will help me to understand how visually impaired children learn to code with products like Torino.
- If you take part in this study, you will get to learn about coding using Torino.
- I will also give you with a summary of what I find out during the study.
- The main downside to taking part in the study is that you will be giving up an hour of your time.
- There are no predictable risks in participating in the study.

Will my taking part be kept confidential?

- The video recording of the workshop is private and will be stored securely.
- When I write or talk about what I find out, I won't tell anyone your name.
- No one will be able to identify you.
- I will follow all the rules in UK law about collecting and storing data.
- Only people working directly on this project will be able to see your information.
- Your information will be protected with a password.
- Fake names will be used so no one will be able to identify you from your information.

How is the project being funded?

I am funded by the EPSRC research council.

What will happen to the results of the study?

- I will give you a summary of what I find out.
- The research findings will be a part of my course at university.
- I will write about my research findings and give talks about them.

Who should I contact for further information?

If you have any questions or require more information about this study, please contact me using the following contact details:

Alex Hadwen-Bennett
School of Education, Communication and Society | King's College London
Telephone: 020 7848 3945
E-mail: alex.hadwen-bennett@kcl.ac.uk

What if I have further questions, or if something goes wrong?

If this study has harmed you in any way or if you wish to make a complaint about how this study has been carried out you can contact King's College London using the details below for further advice and information:

Dr Sue Sentance
School of Education, Communication and Society, King's College London
Telephone: 020 7848 3117
E-mail: sue.sentance@kcl.ac.uk

Thank you for reading this information sheet and for considering taking part in this research.

Appendix 4 Pilot Study Consent Form (Parent)

Version Number 23/01/18

CONSENT FORM FOR PARTICIPANTS IN RESEARCH STUDIES

Please complete this form after you have read the Information Sheet and/or listened to an explanation about the research.



Title of Study: An Investigation of Exploratory Procedures used by Visually Impaired Children when Working with Physical Programming Languages

King's College Research Ethics Committee Ref: *LRS-17/18-5607*

Thank you for considering taking part in this research. The person organising the research must explain the project to you before you agree to take part. If you have any questions arising from the Information Sheet or explanation already given to you, please ask the researcher before you decide whether to join in. You will be given a copy of this Consent Form to keep and refer to at any time.

I confirm that I understand that by ticking/initialling each box I am consenting to this element of the study. I understand that it will be assumed that unticked/initialled boxes mean that I DO NOT consent to that part of the study. I understand that by not giving consent for any one element I may be deemed ineligible for the study.

Please tick or initial

1. *I confirm that I have read and understood the information sheet dated 22/02/18 for the above study. I have had the opportunity to consider the information and asked questions which have been answered satisfactorily.
2. *I understand that I will be able to withdraw my child's data up to 4 weeks after the workshop.
3. *I consent to the processing of my child's personal information for the purposes explained to me. I understand that such information will be handled in accordance with the terms of the UK Data Protection Act 1998.
4. *I understand that my child's information may be subject to review by responsible individuals from the College for monitoring and audit purposes.
5. *I understand that confidentiality and anonymity will be maintained and it will not be possible to identify my child in any publications.

Please tick or initial

- 6. I agree to be contacted in the future by King's College London researchers who would like to invite my child to participate in follow up studies to this project, or in future studies of a similar nature.
- 7. I agree that the research team may use my child's data for future research and understand that any such use of identifiable data would be reviewed and approved by a research ethics committee. (In such cases, as with this project, data would not be identifiable in any report).
- 8. I understand that the information I have submitted will be published as a report and I wish to receive a copy of it.
- 9. I consent to the workshop being video recorded.

Name of Child

Name of Parent/Guardian

Date

Signature

Appendix 5 Pilot Study Consent Form (Child)

Version Number 23/01/18

CONSENT FORM FOR PARTICIPANTS IN RESEARCH STUDIES

Please complete this form after you have read the Information Sheet and/or listened to an explanation about the research.



Title of Study: An Investigation of Exploratory Procedures used by Visually Impaired Children when Working with Physical Programming Languages

King's College Research Ethics Committee Ref: LRS-17/18-5607

Thank you for thinking of taking part in this research. The person organising the research must explain the project to you before you agree to take part. If you have any questions after reading the Information Sheet please ask the researcher before you decide whether to take part. You will be given a copy of this Consent Form to keep and refer to at any time.

I confirm that I understand that by ticking/initialling each box I am agreeing to this part of the study. I understand that we will think that unticked/initialled boxes mean that I DO NOT agree to that part of the study. I understand that if I don't agree to one part I may not be able to take part.

Please tick or initial

Please tick or initial

1. *I confirm that I have read and understood the information sheet dated 22/02/18 for the above study. I have had the chance to think about the information and asked questions which have been answered .
2. *I understand that I will be able to ask for my data to be removed up to 4 weeks after the workshop.
3. *I agree to the use of my personal information for the uses explained to me. I understand that my information will be looked after according to UK law.
4. *I understand that my information may checked by responsible people from the College to make sure it is being collected and stored properly.
5. *I understand that my personal information will be kept secret and it will not be possible to identify me in the results of the research.
6. I agree to be contacted in the future by King's College London researchers who would like to ask me to take part in similar research.
7. I agree that the research team may use my data for future research and understand that any such use of my personal data would be checked and approved by a research ethics committee.

8. I understand that the information I have given will be put in a report and I would like to be told the what the researchers find out.

9. I agree to the workshop being video recorded.

Name

Date

Signature*

*If the child has given verbal consent to all aspects of this research and is unable to sign this form, please write "Verbal Consent Given" in the signature space.

Appendix 6 Main Study Ethical Approval

Research Ethics
Office

Franklin Wilkins Building
5,9 Waterloo Bridge Wing
Waterloo Road
London SE1 9NH
Telephone 020 7848 4020/4070/4077
rec@kcl.ac.uk



Alexander Hadwen-Bennett

31 August 2018

Dear Alexander

LRS-17/18-7757 An exploration of strategies for teaching computer science concepts to children with visual impairments

LRS-17/18-7757 Thank you for submitting your application for the above project. I am pleased to inform you that full approval has been granted by Committee Name. LRS-17/18-7757

Ethical approval has been granted for a period of **three years** from 31 August 2018. You will not be sent a reminder when your approval has lapsed and if you require an extension you should complete a modification request, details of which can be found here:

<https://internal.kcl.ac.uk/innovation/research/ethics/applications/modifications.aspx>

Please ensure that you follow the guidelines for good research practice as laid out in UKRIO's Code of Practice for research: <http://www.kcl.ac.uk/innovation/research/support/conduct/cop/index.aspx>.

Any unforeseen ethical problems arising during the course of the project should be reported to the panel Chair, via the Research Ethics Office.

Please note that we may, for the purposes of audit, contact you to ascertain the status of your research.

We wish you every success with your research.

Yours sincerely,

Miss Annah Whyton

Senior Research Ethics Officer

For and on behalf of:

E&M Research Ethics Panel

E&M Research Ethics Panel

Appendix 7 Main Study Participant Information Sheet (Parent)

Version Number 21/06/18

INFORMATION SHEET FOR PARTICIPANTS

REC Reference Number: LRS-17/18-7757

YOU WILL BE GIVEN A COPY OF THIS INFORMATION SHEET



Title of study

An exploration of how visually impaired learners can develop an understanding of key programming concepts using physical programming languages

Invitation Paragraph

I would like to invite your child to participate in this research project which forms part of my PhD research. You should only give permission for your child to participate if you want to; choosing not to take part will not disadvantage you or your child in anyway. Before you decide whether you want your child to take part, it is important for you to understand why the research is being done and what your child's participation will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask me if there is anything that is not clear or if you would like more information.

What is the purpose of the study?

Most children in primary school are currently taught how to program using visual programming languages that use puzzle-like pieces that can be snapped together. The visual nature of these languages makes them inaccessible to most visually impaired learners.

Physical programming languages are an alternative to visual languages, they use physical blocks that can be joined together to create programs. The aim of this study is to identify the ways in which learners develop an understanding of physical programs by exploring them with their hands.

In this study we will be using Torino, a physical programming language that is designed to be inclusive of learners with visual impairments. It can be used to build programs that produce music, stories and poetry.

Why has my child been invited to take part?

I am inviting visually impaired children aged between 7 and 15 to take part in this study.

What will happen to my child if they take part?

If you decide to give permission for your child to take part, you will be given this information sheet to keep and will be asked to sign a consent form. Your child will be invited to participate in a series of coding workshops. During the workshops your child will work with another visually impaired child to solve challenges using the Torino physical programming language.

The workshops will be video recorded to enable analysis of the use of hands to explore physical programs. The video recordings will be stored securely. Even if you have decided to take part, you are still free to stop your child's participation at any time during the workshops.

Does my child have to take part?

Participation is voluntary. You do not have to agree for your child take part. Your child will not be disadvantaged in any way if they do not participate. You should read this information sheet and if you have any questions you should ask the researcher.

What are the possible risks of taking part?

The main disadvantage to taking part in the study is that your child will be donating some of their time take part. There are no foreseeable risks in participating in the study.

What are the possible benefits of taking part?

The information I get from the study will help to develop an understanding of how visually impaired children learn to program with physical programming languages. If your child takes part in this study, they will get the opportunity to learn about coding using a physical programming language. Furthermore, I will provide you with a summary of a final report describing the main findings.

Data handling and confidentiality

Your data will be processed in accordance with the General Data Protection Regulation 2016 (GDPR). What is recorded during the workshop is strictly confidential and will be held securely. All data for analysis will be anonymised. In reporting on the research findings, I will not reveal the names of any participants or the organisation involved. Images or short video clips showing only your child's hands may be used in reports I write or presentations I give, their face will never be shown. At all times there will be no possibility of your child as an individual being linked with the data.

All information gathered within the workshops and held on password-locked computer files and locked cabinets within King's College London. No data will be accessed by anyone other than the researchers directly involved with this study at King's College London; and anonymity of the material will be protected by using false names. The video recordings will be deleted once my PhD thesis is submitted. An anonymised written summary of the data will be kept for future research, your child will be not identifiable from this data.

Data Protection Statement

The data controller for this project will be King's College London (KCL). The University will process your personal data for the purpose of the research outlined above. The legal basis for processing your personal data for research purposes under GDPR is a 'task in the public interest' You can provide your consent for the use of your personal data in this study by completing the consent form that has been provided to you.

You have the right to access information held about you. Your right of access can be exercised in accordance with the General Data Protection Regulation. You also have other rights including rights of correction, erasure, objection, and data portability. Questions, comments and requests about your personal data can also be sent to the King's College London Data Protection Officer Mr Albert Chan info-compliance@kcl.ac.uk. If you wish to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk.

What if I change my mind about my child taking part?

You are free to withdraw your child from the study at any point, without having to give a reason. Withdrawing your child from the study will not affect you in any way. You are able to withdraw your child's data from the study up until 31/10/18, after which withdrawal of their data will no longer be possible due to inclusion of their data in written reports. If you withdraw your child from the study after 31/10/18, we will keep the information about them that we have already obtained. To safeguard their rights, we will use the minimum personally-identifiable information possible.

How is the project being funded?

The researcher is funded by an EPSRC studentship.

What will happen to the results of the study?

I will produce a final report summarising the main findings, which will be sent to you. The research findings will form part of my PhD thesis. I also plan to disseminate the research findings through publication and conferences.

Who should I contact for further information?

If you have any questions or require more information about this study, please contact me using the following contact details:

Alex Hadwen-Bennett
School of Education, Communication and Society | King's College London
Telephone: 020 7848 3945
E-mail: alex.hadwen-bennett@kcl.ac.uk

What if I have further questions, or if something goes wrong?

If this study has harmed you in any way or if you wish to make a complaint about the conduct of the study you can contact King's College London using the details below for further advice and information:

Dr Sue Sentance
School of Education, Communication and Society, King's College London
Telephone: 020 7848 3117
E-mail: sue.sentance@kcl.ac.uk

Thank you for reading this information sheet and for considering taking part in this research.

Appendix 8 Main Study Participant Information Sheet (Child)

Version Number 21/06/18

INFORMATION SHEET FOR PARTICIPANTS

REC Reference Number: LRS-17/18-7757

YOU WILL BE GIVEN A COPY OF THIS INFORMATION SHEET



Title of study

An exploration of how visually impaired learners can develop an understanding of key programming concepts using physical programming languages

Introduction

I'm inviting you to take part in this research project which is part of my course at university. You should only agree to take part if you want to, you are free to decide not to take part.

Before you decide if you want to take part, I need to tell you about the research. Please read the following information and discuss it with others if you wish. Ask me if you have any questions or want me to explain something to you.

What is the purpose of the study?

In school, most children are taught how to code by snapping together puzzle pieces on a computer screen. These systems can be hard for visually impaired children to use.

Another way of learning to program a computer is to use physical blocks that can be connected. I want to find out about how blind children use their hands to explore a physical program.

In this study we will be using Torino, which can be used to build programs that create music, stories and poetry.

Why have I been asked to take part?

I am inviting visually impaired children aged between 7 and 15 to take part in this study.

What will happen to me if I take part?

- If you tell us that you want to take part, you will be given this information sheet to keep.
- You will be invited to take part in a series of regular coding workshops. During the workshops you will work with another blind child to solve challenges using Torino and learn about programming.
- The workshop will be video recorded, so I can see how you use your hands when working with Torino.
- The video recordings will be stored safely.
- If you decide to take part, you are still free to change your mind at any time.

Do I have to take part?

Taking part is voluntary. You do not have to take part. You will not be disadvantaged in any way if you decide not to take part. You should read this information sheet and if you have any questions you should ask me.

What are the possible risks of taking part?

- The main downside to taking part in the study is that you will be giving up some of your time to take part in the workshops.
- There are no predictable risks in participating in the study.

What are the possible benefits of taking part?

- What I find out from the study will help me to understand how visually impaired children learn to code with products like Torino.
- If you take part in this study, you will get to learn about coding using Torino.
- I will also give you with a summary of what I find out during the study.

Data handling and confidentiality

Your data will be processed in accordance with the General Data Protection Regulation 2016 (GDPR).

- The video recordings of the workshops are private and will be stored securely.
- Images or short video clips showing only your hands may be used in reports I write or presentations I give, your face will never be shown.
- When I write or talk about what I find out, I won't tell anyone your name.
- No one will be able to identify you.
- Only people working directly on this project at King's College London will have access to your data.
- Your information will be protected with a password.
- Fake names will be used so no one will be able to identify you from your information.
- The video recordings will be deleted once my university course is complete. A written summary of the data will be kept for future research, you will not be identifiable from this data.

Data Protection Statement

The data controller for this project will be King's College London (KCL). The University will process your personal data for the purpose of the research outlined above. The legal basis for processing your personal data for research purposes under GDPR is a 'task in the public interest' You can provide your consent for the use of your personal data in this study by completing the consent form that has been provided to you.

You have the right to access information held about you. Your right of access can be exercised in accordance with the General Data Protection Regulation. You also have other rights including rights of correction, erasure, objection, and data portability. Questions, comments and requests about your personal data can also be sent to the King's College London Data Protection Officer Mr Albert Chan info-compliance@kcl.ac.uk. If you wish to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk.

What if I change my mind about taking part?

You are free to withdraw from the study at any point, without having to give a reason. Withdrawing from the study will not affect you in any way. You are able to withdraw your data from the study up until 31/10/18, after which withdrawal of your data will no longer be possible due to inclusion of the data in written reports. If you withdraw from the study after

31/10/18, we will keep the information about you that we have already obtained. To safeguard your rights, we will use the minimum personally-identifiable information possible.

How is the project being funded?

I am funded by the EPSRC research council.

What will happen to the results of the study?

- I will give you a summary of what I find out.
- The research findings will be a part of my course at university.
- I will publish my research findings and give talks about them.

Who should I contact for further information?

If you have any questions or require more information about this study, please contact me using the following contact details:

Alex Hadwen-Bennett
School of Education, Communication and Society | King's College London
Telephone: 020 7848 3945
E-mail: alex.hadwen-bennett@kcl.ac.uk

What if I have further questions, or if something goes wrong?

If this study has harmed you in any way or if you wish to make a complaint about how this study has been carried out you can contact King's College London using the details below for further advice and information:

Dr Sue Sentance
School of Education, Communication and Society, King's College London
Telephone: 020 7848 3117
E-mail: sue.sentance@kcl.ac.uk

Thank you for reading this information sheet and for considering taking part in this research.

Appendix 9 Main Study Consent Form (Parent)

Version Number – 21/06/18

CONSENT FORM FOR PARTICIPANTS IN RESEARCH STUDIES

Please complete this form after you have read the Information Sheet and/or listened to an explanation about the research.



Title of Study: An exploration of how visually impaired learners can develop an understanding of key programming concepts using physical programming languages

King's College Research Ethics Committee Ref: LRS-17/18-7757

Thank you for considering taking part in this research. The person organising the research must explain the project to you before you agree to take part. If you have any questions arising from the Information Sheet or explanation already given to you, please ask the researcher before you decide whether to join in. You will be given a copy of this Consent Form to keep and refer to at any time.

I confirm that I understand that by ticking/initialling each box I am consenting to this element of the study. I understand that it will be assumed that unticked/initialled boxes mean that I DO NOT consent to that part of the study. I understand that by not giving consent for any one element I may be deemed ineligible for the study.

Please tick or initial

1. *I confirm that I have read and understood the information sheet dated 21/06/18 for the above study. I have had the opportunity to consider the information and asked questions which have been answered to my satisfaction.
2. I consent voluntarily for my child to be a participant in this study and I can withdraw them from the study at any time, without having to give a reason.
3. *I consent to the processing of my child's personal information for the purposes explained to me in the Information Sheet. I understand that such information will be handled in accordance with the terms of the General Data Protection Regulation.
4. *I understand that my child's information may be subject to review by responsible individuals from the College for monitoring and audit purposes.
5. I understand that confidentiality and anonymity will be maintained and it will not be possible to identify my child in any research outputs

Please tick or initial

- 6. I agree to be contacted in the future by King's College London researchers who would like to invite my child to participate in follow up studies to this project, or in future studies of a similar nature.
- 7. I agree that the research team may use my child's data for future research and understand that any such use of identifiable data would be reviewed and approved by a research ethics committee. (In such cases, as with this project, data would/would not be identifiable in any report).
- 8. I understand that the information gathered will be published as a report and I wish to receive a copy of it.
- 9. I consent to the workshops being video recorded.
- 10. I agree to images or short video clips that do not show my child's face being used in research reports and presentations.

Name of Child

Name of Parent/Guardian

Date

Signature

Appendix 10 Main Study Consent Form (Child)

Version Number – 21/06/18

CONSENT FORM FOR PARTICIPANTS IN RESEARCH STUDIES

Please complete this form after you have read the Information Sheet and/or listened to an explanation about the research.



Title of Study: An exploration of how visually impaired learners can develop an understanding of key programming concepts using physical programming languages

King's College Research Ethics Committee Ref: LRS-17/18-7757

Thank you for considering taking part in this research. The person organising the research must explain the project to you before you agree to take part. If you have any questions arising from the Information Sheet or explanation already given to you, please ask the researcher before you decide whether to join in. You will be given a copy of this Consent Form to keep and refer to at any time.

I confirm that I understand that by ticking/initialling each box I am consenting to this element of the study. I understand that it will be assumed that unticked/initialled boxes mean that I DO NOT consent to that part of the study. I understand that by not giving consent for any one element I may be deemed ineligible for the study.

Please tick or initial

Please tick or initial

1. *I confirm that I have read and understood the information sheet dated 21/06/18 for the above study. I have had the chance to think about the information and asked questions which have been answered.
2. I consent voluntarily to take part in this study and understand and I can withdraw from the study at any time, without having to give a reason.
3. * I agree to the use of my personal information for the uses explained to me. I understand that my information will be looked after according to UK law.
4. *I understand that my information may checked by responsible people from the College to make sure it is being collected and stored properly.
5. I understand that confidentiality and anonymity will be maintained and it will not be possible to identify me in any research outputs
6. I agree to be contacted in the future by King's College London researchers who would like to ask me to take part in similar research.
7. I agree that the research team may use my data for future research and understand that any such use of my personal data would be checked and approved by a research ethics committee.

- 8. I understand that the information I have given will be put in a report and I would like to be told the what the researchers find out.
- 9. I consent to the workshops being video recorded.
- 10. I agree to images or short video clips that do not include my face being used in research reports and presentations.

Name **Date** **Signature***

*If the child has given verbal consent to all aspects of this research and is unable to sign this form, please write "Verbal Consent Given" in the signature space.

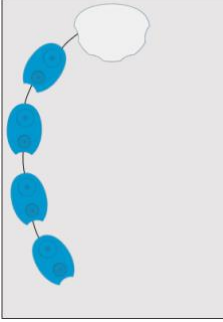
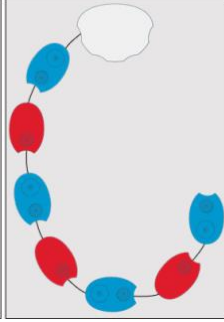
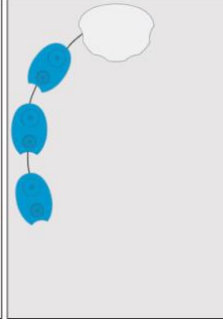
Appendix 11 Main Study Curriculum

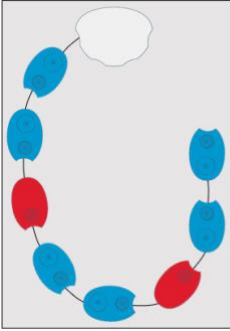
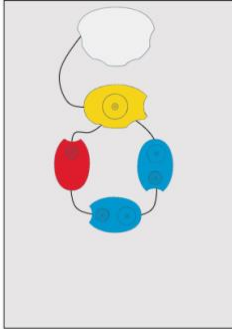
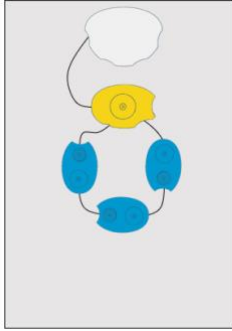
Activity	Learning Objective(s)	Details
1	Become familiar with Torino (including the Play pod and hub).	<ul style="list-style-type: none"> • Introduce Torino • Ask students to describe the play pod. Explain that each pod is a command in a program and can be connected together to form a sequence. • Get the students to put their hands on the hub and ask them to describe it and what they think it does. Explain that all programs start at the hub.
2	Be able to create a simple program that features a sequence of play instructions. Know that a sequence is a set of instructions that are carried out in order.	<ul style="list-style-type: none"> • Get the students to create their first program by plugging one play pod into connector 1 on the hub. • Point out the sounds that are played when pods are plugged in and unplugged. • The students should play their program by pressing the play button on the hub. • Students add three more play pods to their program and run it. • Ask the students if they can work out how to change which sound is played. • Get them to customise their programs. • Explain that this type of program is a sequence.
3	Be able to use an algorithm to finish a partially complete program (sequencing). Be able to state the difference between an algorithm and a program. Be able to identify programs and algorithms that feature sequencing.	<p>Explore, Predict and Test:</p> <ul style="list-style-type: none"> ○ PLAY “There was a young man from” ○ PLAY “seeds” ○ PLAY “Who ate a whole packet of” <p>Introduce algorithm for Limerick 1</p> <ul style="list-style-type: none"> ○ There was a young man from ○ Leeds ○ Who ate a whole packet of ○ seeds ○ In less than an hour ○ His nose was a flower ○ And his head was a garden of ○ weeds! <p>Explain what an algorithm is Dry run Limerick 1 algorithm Complete program based on algorithm Test and follow program</p>
4	Be able to create a new program based on an existing algorithm (sequencing).	<ul style="list-style-type: none"> • Dry run Limerick 2 algorithm <ul style="list-style-type: none"> ○ There once was a Thingamajig ○ Like a Whatsit, ○ but three times as big. ○ When it first came in view ○ It looked something like you ○ But it stayed ○ and turned into a pig • Create program based on Limerick 2 algorithm • Test and follow program
5	Be able to design an algorithm for a given task and turn it into a program (sequencing).	<ul style="list-style-type: none"> • Play the complete Limerick 3 program • Design algorithm for Limerick 3 <ul style="list-style-type: none"> ○ A funny young fellow named Perkins

Activity	Learning Objective(s)	Details
		<ul style="list-style-type: none"> ○ Was terribly fond of small gherkins. ○ One day after tea ○ He ate ninety-three ○ And pickled his internal workings. ● Create program based on algorithm ● Test and follow program
6	<p>Become familiar with the concept of threading with Torino.</p> <p>Know that threading allows multiple sets of instructions to be carried out at the same time.</p>	<ul style="list-style-type: none"> ● Ask each student to create a small sequence with play pods and plug their sequences into the hub. ● Ask them to predict what they think will happen when the program plays.
7	<p>Be able to use an algorithm to finish a partially complete program (sequencing and threading).</p> <p>Be able to identify programs and algorithms that feature threading.</p>	<p>Explore, Predict and Test:</p> <ul style="list-style-type: none"> ○ THREAD 1 <ul style="list-style-type: none"> ▪ PLAY “Dr Foster” ▪ PLAY “Went to Gloucester” ▪ PLAY “In a shower of rain” ○ THREAD 2 <ul style="list-style-type: none"> ▪ PAUSE 2 ▪ PAUSE 1 ▪ PLAY Rain x2 speed <p>Introduce algorithm Dr Foster</p> <ul style="list-style-type: none"> ○ THREAD 1 <ul style="list-style-type: none"> ▪ Dr Foster ▪ Went to Gloucester ▪ In a shower of rain ▪ He stepped in a puddle ▪ Right up to his middle ▪ And never went there again ○ THREAD 2 <ul style="list-style-type: none"> ▪ PAUSE 2 ▪ PAUSE 1 ▪ Rain x2 speed ▪ PAUSE 2 ▪ Splash <p>Dry run Dr Foster algorithm</p> <p>Complete program based on algorithm</p> <p>Test and follow program</p>
8	<p>Be able to create a new program based on an existing algorithm (threading).</p>	<ul style="list-style-type: none"> ● Dry run Story algorithm <ul style="list-style-type: none"> ○ THREAD 1 <ul style="list-style-type: none"> ▪ Helen is looking for her friends ▪ She thinks they might be in the Cupboard ▪ She opens the door ▪ Her friends jump out and yell surprise ○ THREAD 2 <ul style="list-style-type: none"> ▪ PAUSE 2 ▪ PAUSE 2 ▪ PAUSE 2 ▪ Cheer ● Create program based on Story algorithm ● Test and follow program

Activity	Learning Objective(s)	Details
9	Be able to design an original algorithm and turn it into a program (threading).	<ul style="list-style-type: none"> • Design an algorithm for a poem based on the sounds from the “Mashed potatoes on the ceiling” poem. It should feature a separate thread for sound effects • Create program based on algorithm • Test and follow program
10	Become familiar with the loop pod in Torino and the concept of Torino.	<ul style="list-style-type: none"> • Introduce the loop pod, explain that it can be used to repeat a set of commands. The explain that this is known as repetition in programming • Get the students to take turns holding the loop pod, point out the dial and where the loop starts and ends. Ask the students to feel the groove that connects the connectors at the start and the end of the loop, point out that the start of the loop is curved inwards • Get students to create a sequence and play it inside the loop • Introduce the concept of repetition
11	Be able to use an algorithm to finish a partially complete program (repetition). Be able to identify programs and algorithms that feature repetition.	<p>Explore, Predict and Test:</p> <ul style="list-style-type: none"> ○ Loop 2 times <ul style="list-style-type: none"> ▪ PLAY “One” ▪ PLAY “Two” <p>Introduce Counting algorithm</p> <ul style="list-style-type: none"> ○ Loop 3 times <ul style="list-style-type: none"> ▪ PLAY “One” ▪ PLAY “Two” ▪ PLAY “Three” <p>Dry run Counting algorithm Complete program based on algorithm Test and follow program</p>
12	Be able to create a new program based on an existing algorithm (repetition and sequencing).	<ul style="list-style-type: none"> • Dry run Jingle Bells algorithm <ul style="list-style-type: none"> ○ LOOP 2 times <ul style="list-style-type: none"> ▪ Jingle ▪ Bells ○ Jingle ○ All ○ The ○ Way • Create program based on Jingle Bells algorithm • Test and follow program
13	Be able to design an original algorithm and turn it into a program (repetition and sequencing).	<ul style="list-style-type: none"> • Play the song ‘Row Your Boat’ • Design an algorithm to recreate the song <ul style="list-style-type: none"> ○ Loop 3 times <ul style="list-style-type: none"> ▪ Row ○ Your Boat ○ Gently Down the stream ○ Merrily 1 ○ Merrily 2 ○ Merrily 3 ○ Merrily 4 ○ Life is but a dream • Create program based on algorithm • Test and follow program

Activity	Learning Objective(s)	Details
14	Be able to create an original program without the aid of an algorithm (repetition).	<ul style="list-style-type: none"> • Play the song ‘Frere Jacques’ • Create program to recreate the song <ul style="list-style-type: none"> ○ Loop 2 times <ul style="list-style-type: none"> ▪ C5 ▪ D5 ▪ E5 ▪ C5 ○ Loop 2 times <ul style="list-style-type: none"> ▪ E5 ▪ F5 ▪ G5 • Test and follow program • Extension – complete the program <ul style="list-style-type: none"> ○ Loop 2 times <ul style="list-style-type: none"> ▪ G5 ▪ A5 ▪ G5 ▪ F5 ▪ E5 ▪ C5 ○ Loop 2 times <ul style="list-style-type: none"> ▪ C5 ▪ G5 ▪ C5
15	Be able to design an algorithm for an original story and turn the algorithm into a program.	<ul style="list-style-type: none"> • Set students the homework to plan an original story containing two threads. • These stories are turned into sound samples in Code Jumper. • The students are asked to build a program to create their story.
16	Be able to complete a program that uses repetition on two threads.	<ul style="list-style-type: none"> • Explore, Predict and Test: <ul style="list-style-type: none"> ○ Loop 8 times <ul style="list-style-type: none"> ▪ PLAY “Cymbal Low” • Play the ‘Percussion Loop’ rhythm • Ask the students to adapt the program to match the completed program. <ul style="list-style-type: none"> ○ Loop 8 times <ul style="list-style-type: none"> ▪ PLAY “Cymbal Low” ▪ PAUSE ½ beat • Play the ‘Threads and Loops 1’ rhythm • Ask the students to adapt their existing program to match the completed program. <ul style="list-style-type: none"> ○ THREAD 1 Bass <ul style="list-style-type: none"> ▪ Loop 8 times <ul style="list-style-type: none"> • PAUSE ½ beat • PLAY “Bass Low” ○ THREAD 2 Cymbal <ul style="list-style-type: none"> ▪ Loop 8 times <ul style="list-style-type: none"> • PLAY “Cymbal Low” • PAUSE ½ beat

Activity	Learning Objective(s)	Details
17	Be able to add a third thread to an existing two-threaded program.	<ul style="list-style-type: none"> Ask the students to extend their activity 16 program by adding another beat to it using another thread and loop.
18	Be able to build a program that uses multiple loops on one thread	<ul style="list-style-type: none"> Give the students the algorithm below and ask them to turn it into a Torino program. <ul style="list-style-type: none"> THREAD 1 Body Percussion <ul style="list-style-type: none"> Loop 4 times <ul style="list-style-type: none"> PLAY “Clap” PAUSE ½ beat Loop 4 times <ul style="list-style-type: none"> PLAY “Finger Click” PAUSE ½ beat
19	Be able to design and build a program that uses multiple loops on one thread	<ul style="list-style-type: none"> Play the “Multiple Loops” rhythm and ask the students to design the algorithm for it. Ask the students to build the program based on their algorithm. <ul style="list-style-type: none"> THREAD 1 Body Percussion <ul style="list-style-type: none"> Loop 8 times <ul style="list-style-type: none"> PLAY “Clap” at 0.5 times speed PLAY “TongueClick” at 0.5 times speed PAUSE ½ beat PLAY “Stomp” at 0.5 times speed PAUSE ½ beat Loop 8 times <ul style="list-style-type: none"> PLAY “FingerSnap” at 0.5 times speed PLAY “TongueClick” at 0.5 times speed
20	Assess sense of sequence	<ul style="list-style-type: none"> The students are presented with the following mini-programs and asked to explore them. They are then played an example program and asked to select which of the three mini-programs could have created the sound they heard. <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;"> <p>Program A</p>  </div> <div style="text-align: center;"> <p>Program B</p>  </div> <div style="text-align: center;"> <p>Program C</p>  </div> </div>
21	Assess sense of repetition	<ul style="list-style-type: none"> The students are presented with the following mini-programs and asked to explore them. They are then played an example program and asked to select which of the three mini-programs could have created the sound they heard.

Activity	Learning Objective(s)	Details
		<div style="display: flex; justify-content: space-around; text-align: center;"> <div data-bbox="815 237 1046 577"> <p>Program A</p>  </div> <div data-bbox="1051 237 1283 577"> <p>Program B</p>  </div> <div data-bbox="1287 237 1519 577"> <p>Program C</p>  </div> </div>
20	To develop an understanding of nested loops.	<ul style="list-style-type: none"> • Get the students to create a simple program which contains a loop that repeats 8 times. • Ask the students how they could repeat their set of commands more than 8 times (they could put their loop inside another loop, known as a nested loop). • The students need to place their loop inside another loop, the outer loop should be set to 2 and the inner one to 5. Ask them to predict how many times it will repeat their set of commands (10 times and $2 \times 5 = 10$).
21	To be able to create a program that features a nested loop	<ul style="list-style-type: none"> • Give the students the following algorithm and ask them to turn it into a program <ul style="list-style-type: none"> ○ THREAD 3 Body Percussion <ul style="list-style-type: none"> ▪ LOOP 3 times <ul style="list-style-type: none"> ▪ LOOP 3 times <ul style="list-style-type: none"> ○ PLAY Finger Snap ○ PAUSE ½ ○ PLAY Tongue Click ○ PAUSE ½ ▪ END LOOP ▪ PLAY Stomp ▪ END LOOP ○ END THREAD
22	To be able to design and create a program that features a nested loop	<ul style="list-style-type: none"> • Play the students the example program and ask them to design the algorithm for it. • Ask the students to create the program based on their algorithm. <ul style="list-style-type: none"> ○ THREAD 3 Body Percussion <ul style="list-style-type: none"> ▪ LOOP 4 times <ul style="list-style-type: none"> • LOOP 2 times <ul style="list-style-type: none"> ○ PLAY Clap 1.5 ○ PAUSE ¼ • END LOOP ▪ PLAY Stomp 0.5 ○ END LOOP
23	Be able to create a program that features nested loops on multiple threads	<ul style="list-style-type: none"> • Play the students the example program and ask them to recreate it. <ul style="list-style-type: none"> ○ THREAD 1 Eye of the Tiger <ul style="list-style-type: none"> ▪ LOOP 2 times <ul style="list-style-type: none"> • Play Sample 1 • LOOP 2 times <ul style="list-style-type: none"> ○ PLAY Sample 2 • END LOOP

Activity	Learning Objective(s)	Details
		<ul style="list-style-type: none"> • Play Sample 3 • • Play the students the example program with backing and ask them to adapt their program to recreate it. <ul style="list-style-type: none"> ○ THREAD 1 Eye of the Tiger <ul style="list-style-type: none"> ▪ PAUSE 2 ▪ PAUSE 2 ▪ PAUSE ½ ▪ LOOP 2 times <ul style="list-style-type: none"> • Play Sample 1 • LOOP 2 times <ul style="list-style-type: none"> ○ PLAY Sample 2 • END LOOP • Play Sample 3 ▪ END LOOP ○ THREAD 4 Backing <ul style="list-style-type: none"> ▪ LOOP 3 times <ul style="list-style-type: none"> • LOOP 6 times <ul style="list-style-type: none"> ○ PLAY Backing • END LOOP ▪ END LOOP
24	Be able to create a program that features a combination of loops and nested loops.	<ul style="list-style-type: none"> • Play the students the example program and ask them to recreate it. <ul style="list-style-type: none"> ○ THREAD 1 Gimme Gimme Gimme <ul style="list-style-type: none"> ▪ LOOP 2 times <ul style="list-style-type: none"> • LOOP 2 times <ul style="list-style-type: none"> ○ PLAY Sample 1 • END LOOP • Play Sample 2 ▪ END LOOP ▪ Play Sample 3 ▪ • Play the students the example program with backing and ask them to adapt their program to recreate it. <ul style="list-style-type: none"> ○ THREAD 1 Gimme Gimme Gimme <ul style="list-style-type: none"> ▪ LOOP 2 times <ul style="list-style-type: none"> • Play Sample 4 ▪ END LOOP ▪ LOOP 2 times <ul style="list-style-type: none"> • LOOP 2 times <ul style="list-style-type: none"> ○ PLAY Sample 1 • END LOOP • Play Sample 2 ▪ END LOOP ▪ Play Sample 3

Activity	Learning Objective(s)	Details
25	Be able to create a program that features nested loops on multiple threads	<ul style="list-style-type: none"> • Play the students the example program and ask them to recreate it. <ul style="list-style-type: none"> ○ THREAD 1 Popcorn <ul style="list-style-type: none"> ▪ LOOP 2 times <ul style="list-style-type: none"> • Play Sample 1 • LOOP 2 times <ul style="list-style-type: none"> ○ PLAY Sample 2 • END LOOP • Play Sample 3 • Play the students the example program with backing and ask them to adapt their program to recreate it. <ul style="list-style-type: none"> ○ THREAD 1 Popcorn <ul style="list-style-type: none"> ▪ PAUSE 2 ▪ LOOP 2 times <ul style="list-style-type: none"> • LOOP 2 times <ul style="list-style-type: none"> ○ PLAY Sample 1 • END LOOP • PLAY Sample 2 • PLAY Sample 3 ▪ END LOOP • Play the students the example program with backing and ask them to adapt their program to recreate it. <ul style="list-style-type: none"> ○ THREAD 1 Backing <ul style="list-style-type: none"> ▪ LOOP 2 times <ul style="list-style-type: none"> • LOOP 6 times <ul style="list-style-type: none"> ○ PLAY Backing • END LOOP ▪ END LOOP
26	To develop an understanding of selection	<ul style="list-style-type: none"> • Introduction to selection pod and allow students to explore it • Ask the students what they think the selection pod does and then fill in the gaps in understanding • Get students to create two sequences and plug them into the selection pod • Get them to set the values on the dials randomly and predict which sequence they think will play.
27	To be able to develop an original program that employs selection and random	<ul style="list-style-type: none"> • Introduce the Story sound set to the students and ask them to develop an original story that could have two possible endings. • Introduce the random plug and get the students to use it to turn their story into a dynamic story. Discuss how we can make the odds of each ending playing fair.
28	To be able to use selection and repetition together in a program	<ul style="list-style-type: none"> • Challenge the students to adapt their dynamic story so that it plays 4 times with a 1 second delay between each repetition.

Activity	Learning Objective(s)	Details
29	To understand the purpose of and how to use variables	<ul style="list-style-type: none"> • Introduce the variable plugs and the value plugs that can be used to assign a value to a variable. • Ask the students to create a program on the piano thread that features two play pods inside a loop. • Get the students to choose a value and assign it to a variable in the first play pod. • Get the students to use the value stored in the variable for the sound of the second play pod.
30	To understand how variables and random values can be used together	<ul style="list-style-type: none"> • Ask the students to predict what they think will happen if we assign a random value to the variable on the first play pod. • Get them to test their predictions
31	To understand how counters can be used in programs	<ul style="list-style-type: none"> • Introduce the incrementor and decrementor plugs. • Get them to predict what they think they will do if they are used in conjunction with the variables and loops. • Get the students to create a program that will produce a piano scale.
32	To be able to develop programs that employ selection, repetition, variables and counters	<ul style="list-style-type: none"> • Play the example explosion countdown program and ask the students to discuss how they could recreate it. • Get the students to recreate and test the program based on their plans.

Appendix 12 Steven – Session 8 Timeline

[00:00:23.05] Teacher: Hands student mini version of loop pod

[00:00:37.10] Student: Identifies it as a pause pod because it has one circle

[00:01:23.01] Student: Realises it could also be a loop pod

[00:01:53.10] Teacher: Gives student a mini program with a loop pod with 3 play pods inside

[00:02:13.25] Student: "It's a difficult texture to feel I'm telling you"

[00:02:32.05] Student: Explores the play pods inside the loop in order of execution by following the program

[00:03:02.15] Teacher: Gives another program that features a sequence made up of play and pause pods

[00:03:10.17] Student: Follows the program in order of execution and correctly identifies each pod type

[00:03:42.00] Teacher: Gives another program with two plays and a pause inside a loop

[00:03:43.08] Student: Follows the program in order of execution, identifying the pods correctly. The exploring forms a looping motion

[00:04:21.05] Teacher: Plays the sound of a program for him to identify which mini program could create it

[00:05:11.17] Student: When describing the program he describes "it's got three play pods in there" as he explores them, "in there" seems to refer to the loop

[00:05:53.28] Student: Identifies there are six separate sounds in the program being played

[00:06:05.18] Student: Thinks all the sounds are different although there are in fact two sounds repeated three times

[00:06:40.07] Student: After listening to it a couple more times he identifies that two sounds are repeating "actually two sounds are repeating"

[00:07:02.00] Teacher: Asks how many times the two sounds repeat

[00:07:02.00] Student: Initially says two times then says 6 times (the total number of sounds in the program) (then taps his fingers on the table six times)

[00:07:07.04] Student: After listening to the program again he realises it repeats three times

[00:08:02.17] Student: Feels the pods as he is explaining that there are three play pods which means it couldn't create the sound he heard

[00:08:11.19] Student: Correctly identifies that the loop pod would need to have two plays inside it

[00:09:37.28] Student: Initially thinks the loop pod would need to be set to two times rather than 3

[00:10:47.06] Student: After listening to the program again "the two sounds are repeated 3 times"

[00:11:00.07] Teacher: Gives him the sequence mini program

[00:11:05.28] Student: Initially fails to follow the wires which results in him following the program in the wrong direction

[00:11:59.29] Student: Counts the pods as he is following the program to identify whether the right number of sounds are there, he also taps each pod as he says what pod type it is

[00:12:12.21] Student: "There's two play pods here.. pause pod... two play pods here... another pause pod... two play pods here... SO THAT SHOULD BE THE SOUND! (excitedly)"

[00:12:41.12] Teacher: Gives him the next mini program (two plays and a pause inside a loop)

[00:12:48.15] Student: Initially follows the loop in reverse

[00:13:29.16] Student: Says the names of the pods as he follows the program in order

[00:13:42.28] Student: Initially thinks the program couldn't make the sound he heard

[00:14:25.01] Student: Thinks the pause pod is missing

[00:15:24.24] Teacher: Puts the two remaining mini programs side by side so he can compare them

[00:16:00.28] Student: Taps fingers as the sounds play when the example program plays, almost to the rhythm of the repeats

[00:20:00.05] Student: Gets confused between the number of repeats and the number of unique sounds but remembers when he listens to the program again

[00:21:23.10] Student: Thinks there needs to be three play pods inside the loop as it repeats three times

[00:23:10.04] Student: Can remember that the dial can be used to play the sounds three times when holding a real loop pod

[00:25:56.04] Teacher: Gives him an algorithm for a nested loop program (activity 21) on the algorithm board to read

[00:27:42.03] Student: Identifies that a loop pod needs to be added first and adds it to the program successfully

[00:28:16.12] Student: Identifies that the loop needs to be set to three and sets it

[00:28:34.28] Student: When reading the algorithm "Loop three times again, so we need another loop pod"

[00:29:08.22] Student: Gets confused about which wire to use to add the new loop (long or short)

[00:29:50.18] Student: Is initially unsure of which port is the inside of the loop

[00:30:02.00] Student: Successfully sets the inner loop to three

[00:30:17.17] Student: Identifies that a play pod is needed for the next instruction

[00:30:35.29] Student: Correctly adds the play pod to the inside of the inner loop

[00:31:06.13] Student: Identifies the need to add a pause pod next

[00:31:56.17] Student: Correctly adds another play pod "so what we are going to need is a play pod again, I'm getting the hang of this" and sets it

[00:32:52.10] Student: Follows the program along to find the end to add the next pause pod

[00:34:27.16] Student: Gets a bit lost when trying to follow the program to find the inner loop

[00:37:15.21] Student: Listens to the complete program "hahaha, it's really funny! let's play it again"

[00:38:22.13] Student: Mainly follows the program correctly, occasionally skipping onto close wires of the outer loop, correctly identifies that the inner loop will play three times

[00:41:15.27] Teacher: Plays a new example program (activity 22) and gets him to design the algorithm on the algorithm board

[00:42:02.29] Student: Identifies that there are two claps and a stomp

[00:42:05.17] Student: Suggests starting with the loop pod (correctly), initially think the loop needs to be set to 3 times (should be 4)

[00:42:49.25] Student: Realises it repeats 4 times after listening again

[00:44:32.22] Student: Thinks he needs to put clap clap again when it is already inside a loop

[00:46:27.22] Student: Starts making the program, correctly adds a loop and sets it to 4

[00:46:48.29] Student: Next adds a play pod

[00:49:07.06] Student: Successfully builds the program using a single loop rather than nested but it is still correct

[00:49:27.05] Student: Realises the program is too fast and he needs to change the play pod speed

[00:50:44.13] Student: Doesn't realise that there is a pause pod needed in the program

Appendix 13 Steven – Coded Transcript for Session 8

Codes

- Sense of sequencing
- Sense of repetition
 - Sense of order of execution within a loop
 - Sense of relationship between number of instructions, number of repetitions and total number of sounds produced
 - Sense of how instructions are contained within a loop
- Sense associated with affordances of different tools
- Affect
- Debugging
- Gesture
- Spoken
- Play instruction as subroutine
- Using a hand as a placeholder

Activity 8.1 Narrative

In a previous session, Steven was introduced to mini 3D printed versions of the play and pause pods. These were designed to enable example programs to be created in advance for use in the learning activities. They also overcome the limitations imposed by only having eight play pods in each Code Jumper set. Steven was introduced to the mini version of the loop pod and initially identifies it as a pause pod because it has one circle (00:00:37.10). After some discussion and being asked what other pod it could possibly be, he realises that it could also be a loop pod (00:01:23.01).

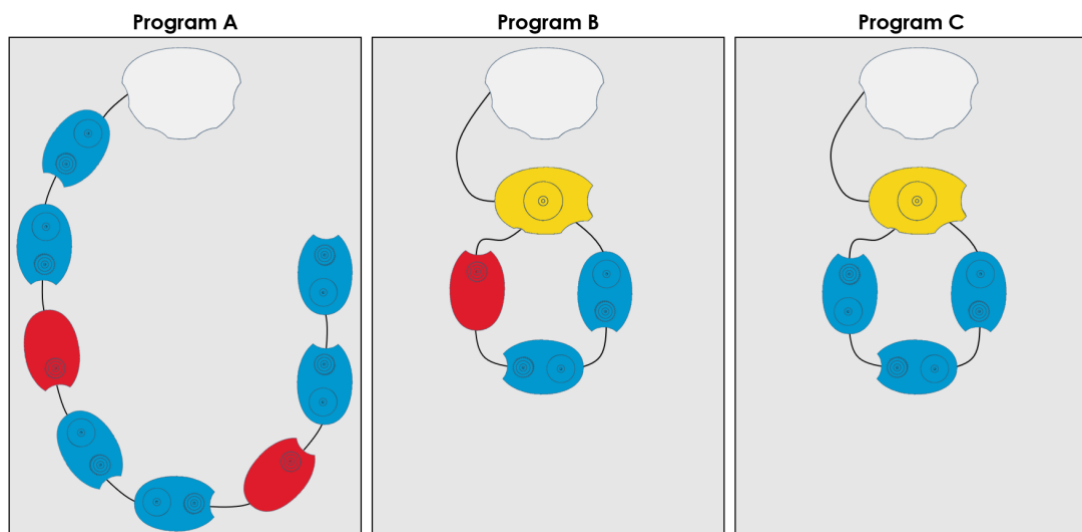


Figure 8a: Mini Code Jumper Programs

Steven is given program C to explore and describe (00:01:53.10). He is initially confused by the mini hub stating “It’s a difficult texture to feel I’m telling you” (00:02:13.25). Next Steven explores the program in order of execution, making a

looping gesture as he follows the loop (00:02:32.05). When he comes back to the loop pod he initially describes it as a pause pod but after being asked if he is sure he corrects himself and refers to it as a loop pod.

I then give Steven program A to explore (00:03:02.15) **which he follows in order of execution, saying the names of each pod as he touches it (00:03:10.17)**. Next Steven is given program B to explore (00:03:42.00) and **once again he follows the program in order of execution, naming the pods and making a looping gesture as he follows the loop (00:03:43.08)**. I then played the example program for Steven to listen to and asked him to identify which of the mini programs could have made that sound (00:04:21.05).

I gave Steven each of the mini programs again in turn to explore and decide whether they could make the sound he heard, starting with program C. **As he explores the program, he identifies each pod, exploring them in order of execution and coming back to the loop pod (00:05:11.17)**. He briefly calls the loop pod a pause pod but immediately corrects himself. On the second iteration of his exploration, as he starts with the loop and taps each of the play pods as he says “**play pod, play pod, play pod...** it’s got three play pods in there” he then returns to the loop pod. He seems to use “in there” to refer to the pods that are inside the loop. Figure 8b shows Steven exploring program C.



Figure 8b: Exploring Mini Program C

When asked whether program C could make the sound of the example program Steven answers yes. I played the example program again and as it is playing Steven follows program C, almost touching one of the three play pods for each repetition. I asked him how many sounds he heard and he immediately answered six, which is correct (00:05:53.28). When asked if any of the sounds are repeated or whether they are all different he says they are all different (00:06:05.18). After listening to the program again a couple more times Steven identifies that two sounds are repeating “actually two sounds are repeating” (00:06:40.07).

Next I asked Steven how many times the two sounds repeat and initially he said two times (the number of unique sounds) and then said six times (the total number of sounds that are produced), following this he tapped his fingers on the table six times (00:07:02.00). After listening to the program again Steven realised that the sounds

repeated three times (00:07:07.04). When asked again if program C could create that sound he heard, Steven says no and after I ask him why he touched the pods as he said “because there’s three play pods and a loop pod in the middle... I think that might be the reason why” and I asked him whether it was the loop pod or the three plays that meant it could not create the sound and he said “the three plays” heard (00:08:02.17). He then correctly identifies that the loop would need to have two play pods inside it (00:08:11.19). When asked how many repetitions the loop pod would need to be set to, Steven initially says two (the number of instructions inside the loop) (00:09:37.28). After asking him to think about how many sounds are produced in total and listening to the program again Steven says “the two sounds are repeated three times” (00:10:47.06).

Having discounted program C as a possible solution, I next give Steven program A again to explore (00:11:00.07). **He initially does not check which end of the sequence is attached to the hub and as a result follows the program in the wrong direction, he corrects this after I suggest that he checks where the program connects to the hub (00:11:05.28). Steven taps the pods as he is exploring the program saying “there’s two play pods here... pause pod... two play pods here... another pause pod... two play pods here... SO THAT SHOULD BE THE SOUND! (excitedly)” (00:11:59.29).**

Having identified that program A could make the sound of the example program, I then give Steven program B to explore again (00:12:41.12). **He initially explores the loop in reverse order of execution but when I ask him which way round it goes he corrects himself. Steven then follows the program again in the correct direction making a looping gesture as he does and saying the name of each pod (00:13:29.16).** When asked if program B could make the sound he heard, Steven says no (00:13:42.28) and when asked why he initially struggles but eventually says that he thinks the pause pod is missing (even though he had already identified a pause pod when exploring the program) (00:14:25.01).

I placed programs A and B side by side in order allow Steven to compare them (00:15:24.24) and Steven explores program A again. **I play the example program again and Steven taps his fingers as the sounds play, almost to the rhythm of the repeats (00:16:00.28).** I ask him to explain how program B differs to program A and he says as he explores the program in order of execution “it’s got two plays on it, one pause and a loop pod” (00:16:45:06). He then goes on **to explore program A again in order of execution, describing the pods as he goes.** He feels the loop on program B and then moves to the hub on program A and says “this one doesn’t have a loop” (00:17:27:04).

I played the example program to Steven again and I ask him if he thinks both programs could make the sound he heard or only one. Initially he says “I’m not sure” so I ask him to think back to what we already know about the program starting with how many sounds there are in total and he correctly answers six. When asked how many unique sounds there are in the program he says three (the number of repetitions) (00:20:00.05). After listening to the example program again he says two and he correctly identifies that it repeats three times.

I ask him to explore program B again and tell me whether or not it could make the sound of the example program. Steven says it could not as there needs to be three

play pods as it repeats three times (00:21:23.10). We discuss how many instructions are needed in the program and I ask him what we need to do to program B to make it repeat three times, he identifies the loop pod and I ask what we need to do to the loop pod and is unable to answer. I get him to feel a real loop pod and he turns the dial on it while he explains that we use it to set the number of times we want it to repeat (00:23:10.04) (shown in Figure 8c).

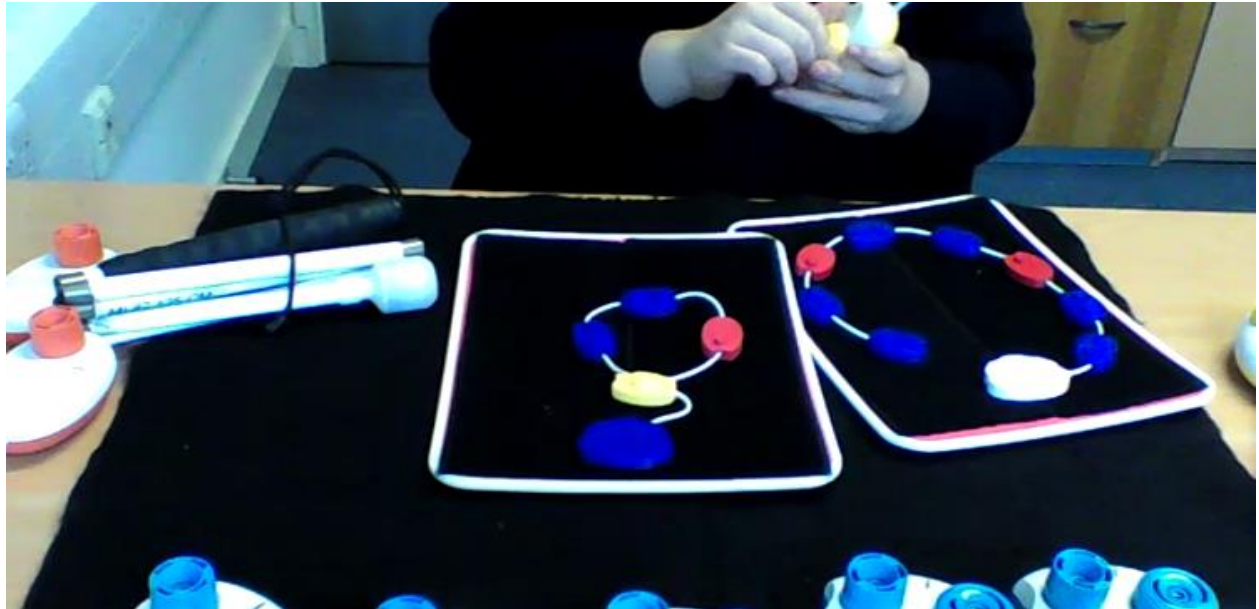


Figure 8c: Turning the Dial on a Loop Pod

Activity 8.1 Analysis

In this activity Steven has demonstrated his sense of sequencing in a number of ways, for example he uses a gesture to explore programs that involves tapping or touching each pod, while at the same time often saying the name of each pod. The fact that Steven almost always follows programs in order of execution seems to suggest that his sense of sequencing is well developed at this stage. On the other hand, his sense of repetition appears to be still forming. For example, the way in which he explores loops gives the impression that at times he is viewing them in terms of sequences, starting exploration at the loop pod, tapping each instruction in turn before returning to the loop pod again and saying its name almost like it is a new instruction.

At times there seems to be some confusion between the number instructions within a loop, the number of repetitions and the total number of sounds the program produces. It is possible that this partly stems from how Steven is conceptualising the play instruction. He seemed sure that program C could create the sound, while program B could not because program C had three play instructions. From following through his reasoning throughout this activity he seems to be viewing each play instruction almost as a call to a subroutine that plays both sounds that are repeated. From this perspective he is correct, if each play instruction played both sounds, it could recreate the program he heard as there are three play pods in program C which would result in the two sounds being played three times.

When I think about it further, each play pod does act like a call to a subroutine as each sound sample that it plays can contain one or multiple sounds. For example, it could play one note or a set of spoken words.

Activity 8.2 Narrative

In the next activity Steven is given a written representation of a nested loop program, it was presented on a mini whiteboard with braille magnetic strips that could be arranged to form different algorithms. The algorithm is given below:

```
LOOP 3 times
  LOOP 3 times
    PLAY Finger Snap
    PAUSE ½
    PLAY Tongue Click
    PAUSE ½
  END LOOP
  PLAY Stomp
END LOOP
```

Steven starts by reading the algorithm and when asked what he needs to do first he identifies that a loop pod needs to be added and does this successfully (00:27:42.03). Next, he identifies that the loop pod needs to be set to three and sets it using the dial (00:28:16.12). When reading the next part of the algorithm he says “[loop three times again, so we need another loop pod](#)” (00:28:34.28). When adding the inner loop pod he briefly gets confused about which wire he needs to plug into the outer loop but remembers when I ask him if it is the long or the short wire (00:29:08.22). He was also initially unsure which port on the outer loop he should plug the new loop into but corrects himself when I ask him which port is inside the loop (00:29:50.18). Steven then sets the inner loop to three repetitions (00:30:02.00) and goes back to the algorithm to check the next instruction. In Figure 8d Steven is reading the algorithm on the board next to the program he is creating.



Figure 8d: Reading the Algorithm

Having realised a play instruction is next, Steven gets a play pod and adds it to the inside of the inner loop (00:30:35.29), he then checks the algorithm again and adds a pause pod (00:31:06.13). The next instruction in the program is another play pod and Steven says “so what we are going to need is a play pod again, I’m getting the hang of this!” (00:31:56.17). Steven then **follows the sequence of instructions** within the loop to find the end to add the next pause pod (shown in Figure 8e) (00:32:52.10).



Figure 8e: Finding the End of a Sequence

In the next step Steven is working out **how to close the loops and initially gets a bit confused between the inner and outer loops and which wires should go where.** Figure 8f shows Steven holding the wire from the end of the inner loop with one hand while following the **sequence to find the end with the other.**

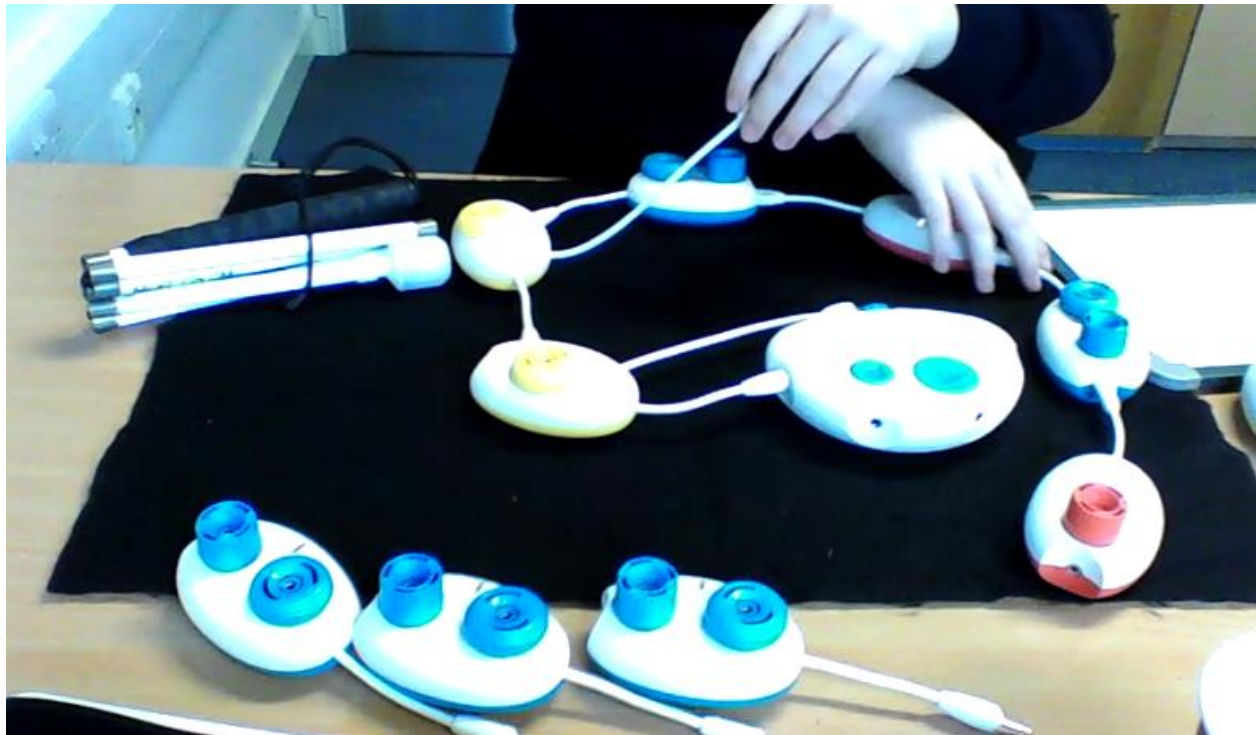


Figure 8f: Closing the Inner Loop

Steven then returns to the algorithm and realises he needs to add another play pod (00:34:10.00). **Initially he finds it challenging to find the exit of the inner loop to add the next play pod into.** **As he explores the program he follows the sequence of instructions within the inner loop before returning to the loop pod.** Once he has successfully added and set the next play pod I give him an extension that will help him close the outer loop around the inner loop (00:35:20.00). He finds the outer loop quickly, but initially struggles to find the long wire that he needs to use to close the loop, he then finds it with further exploration.

When Steven plays the complete program he says **“hahaha, it’s really funny! Let’s play it again!”** (00:37:15.21). Once he had listened to the program again I asked him to explore the program, he started with the **outer loop and then went to the inner loop and went back and forth between these a couple of times.** **He then followed the sequence within the inner loop in order of execution (00:38:22.13),** at one stage he did accidentally skip onto the return wire for the outer loop but corrected himself after further exploration. Next, **he started to follow the sequence in the inner loop again** and I asked him **how many times it will go round and he replied “three times”**. Following this I ask him where the program will go once the inner loop has finished and he initially starts to explore the inner loop again, after I pointed out that he was on the inner loop he corrected himself and moved onto the outer loop. Figure 8g shows Steven exploring the outer loop.



Figure 8g: Exploring the Complete Nested Loop Program

Activity 8.2 Analysis

- At times Steven seemed to use one hand as a placeholder within the program while manipulating the program with the other. For example, when adding a new pod to the program.
- A number of times Steven explored the sequence within the inner loop in order of execution. He did this when asked to explore the program and exploring it himself to find where to add the next pod. This could be viewed as demonstrating his sense of sequencing.
- When given a completed algorithm Steven is able to successfully construct a program featuring a nested loop. Despite some hurdles with syntax he does seem to have a sense of the purpose of nested loops, although it is still developing.
- His confidence in working with loops is developing, supported my comments like “I’m getting the hang of this!”.

Activity 8.3 Narrative

In the final activity of this session I played Steven the sound of a completed program that he needed to recreate (00:41:15.27). The program features two claps followed by a stomp repeated four times and could be solved with a single or a nested loop. The original algorithm from the session plan is included below:

```
LOOP 4 times
  LOOP 2 times
    PLAY Clap 1.5
    PAUSE ¼
```

END LOOP
PLAY Stomp 0.5
END LOOP

After listening to the program I give Steven a blank whiteboard to design his algorithm on and I ask him what the first thing he thinks he will need and he says “so that part sounds like a clap, two claps, one stomp” (00:42:02.29). I then ask him if he thinks anything repeats and he said “shall we start with the loop pod?”. Following this I asked Steven how many times it repeats and he initially says three times after listening to it again he says “twelve... it sounds like twelve” (the total number of sounds produced) (00:42:15.17). Next, I asked him how many times the pattern repeats in the program and as the program is played he said “one... four claps... two stomps... three... four!”. When I asked Steven what the loop pod needs to be set to he said “eight... no four times” (00:42:49.25).

I gave Steven the magnetic pieces he needed to put LOOP 4 times on the board. After this I asked him what we are going to put inside the loop and he said “the clap, clap and stomp” (tapping his finger on the board as he named each sound) (00:43:50:25). I asked him how many clap pieces he wants for the algorithm and he said “clap, clap, two claps”. I gave him the pieces and he added them to the algorithm. Following this I asked Steven what needed to go at the end of the algorithm and he replied “clap, clap again” (00:44:32.22), I asked him why we needed to loop and he said “because the sounds are repeated four times” and he still thought the sounds needed to be duplicated within the loop but was not able to explain why. As I did not have additional copies of the clap magnetic strips for him to add to his algorithm I had to explain that the loop will repeat the sounds for us so there is no need to put the sounds in again. Finally, I give him an end loop strip to add to the end of his algorithm.

Steven started creating his program by adding a loop pod and setting it to four. Following this he added a play pod to the inside of the loop correctly and set the sound (00:46:27.22). When I asked him what was next he read from the algorithm and said “clap, clap, STOMP... no actually two claps”. He correctly adds the next play pod, initially he starts to change the first play pod but corrects himself and sets the second one (00:46:48.29). Following this he said “clap, clap and then we’ve got a stomp” and I ask what we need for that and he replied “another play pod, oh my goodness so many play pods” and then added and set the third play pod. Next Steven went back to the algorithm and read the end loop command, I asked him what we needed to do for this and he replied “we close it” (00:48:45:15). He initially struggles to find the correct port to connect the end of the loop into but quickly realises and successfully closes the loop.

Steven played back the completed program and realised that it was too fast saying “ah... the speed!” and I ask him “the speed of what?” and he replied “the speed of the play pod” (00:49:07.06). He asks me to play back the example program again so he can compare it, following this he goes through a process of trial and error, changing the speed of each sound and playing the program back. In Figure 8h Steven can be seen setting the duration on one of the play pods. As we were running out of time I decided to give Steven a clue and said that I think there may be a gap

between some of the sounds and asked him how we put a gap between sounds, Steven replied “a pause”. At this point I ended the session as we had run out of time.



Figure 8h: Setting the Duration of Play Pods

Activity 8.3 Analysis

- Steven seems to be still developing his understanding of the relationship between the number of repetitions, the number of instructions within a loop and the total number of sounds produced. For example, he initially says that the sounds repeat three times (the number of instructions within the loop when using a single loop). After listening to it again he says twelve (the total number of sounds produced), after asking him how many times the pattern repeats and listening to it again he correctly identified four times.
- Although Steven identified the need for a loop, he thought that the instructions themselves needed to be placed within the loop four times.
- When describing the sequence of instructions that needed to be placed inside the loop he described it as “the clap, clap stomp”.
- In this session Steven is placing the sounds one after the other rather than next to each other as he did in the previous session. This is likely to be due to me correcting him in the previous session, which is a shame as if I had identified this I could have explored how this tied into his sense of the play instruction further.

Appendix 14 Steven – Narrative for Session 8

In preparation for this session I prepared three mini programs, these are shown in Figure 10. I also created a program recording that could have been created by one or more of these example programs. The recording featured a clap and a finger snap followed by a short pause repeated three times. Both programs A and B could create the sound, program A uses a sequence, whereas program B uses a loop. Program C could not create the sound as there are three sounds repeated rather than two.

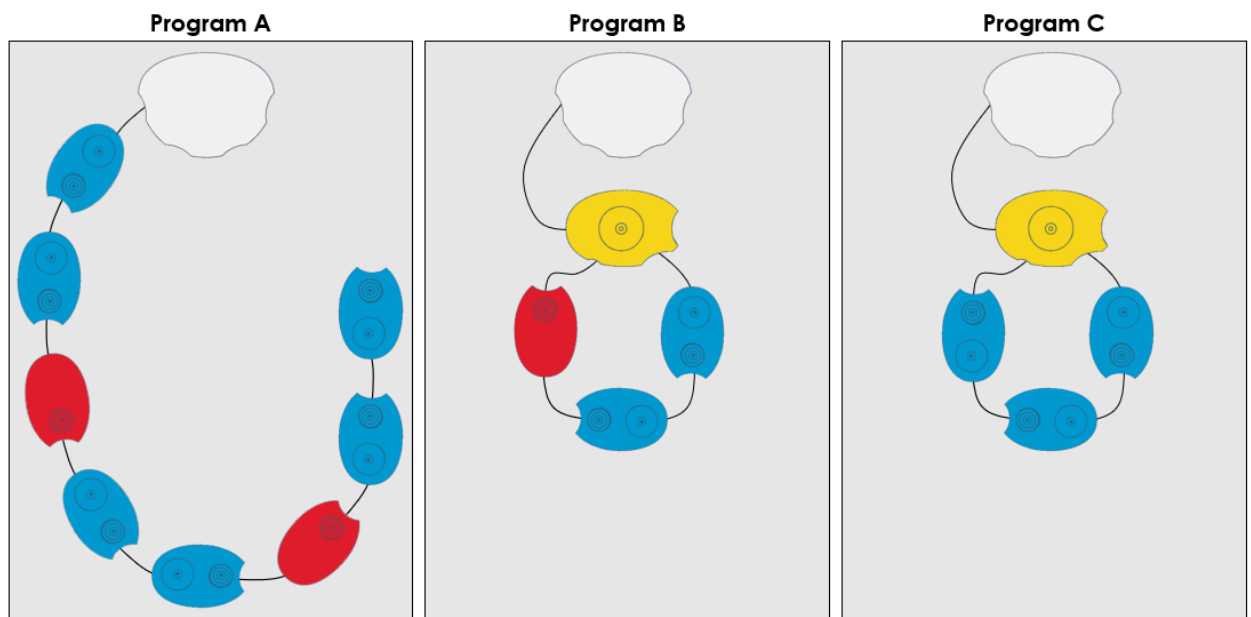


Figure 1: Mini Programs – Set Two

Steven started by exploring program C, following the interior of the loop in order of execution. He then explored program A, following the sequence in order of execution. Finally, Steven explored program B, once again following it in order of execution. After listening to the recording, Steven explored program C again, tapping each pod and naming it in sequence “play pod, play pod, play pod... it’s got three play pods in there”. I asked Steven if program C could create the sound of the recording and as I played it again, he tapped each play pod as each repetition played and said yes it could create the sound.

When I asked Steven how many times the two sounds repeated he initially said “two times” (the number of unique sounds) and then he said “six times” (the total number of sounds produced), following this he tapped his fingers on the table six times. When I asked Steven why he thought that program C could create the sound he heard he said, “because there’s three play pods and a loop pod in the middle”. Steven then explored program A again, naming the pods as he went “there’s two play pods here... pause pod... two play pods here... another pause pod... two play pods here... SO THAT SHOULD BE THE SOUND! (excitedly)”. He then looked at program B again but did not think it could create the sound he heard because it needed to have three play pods as it repeats three times. When I asked Steven how we set the number of repetitions with a loop pod he was unable to answer, however, when I gave him a real loop pod he turned the dial and explained that we use it to set

the number of times we want to repeat it. Steven can be seen turning the dial on the loop pod in Figure 2.

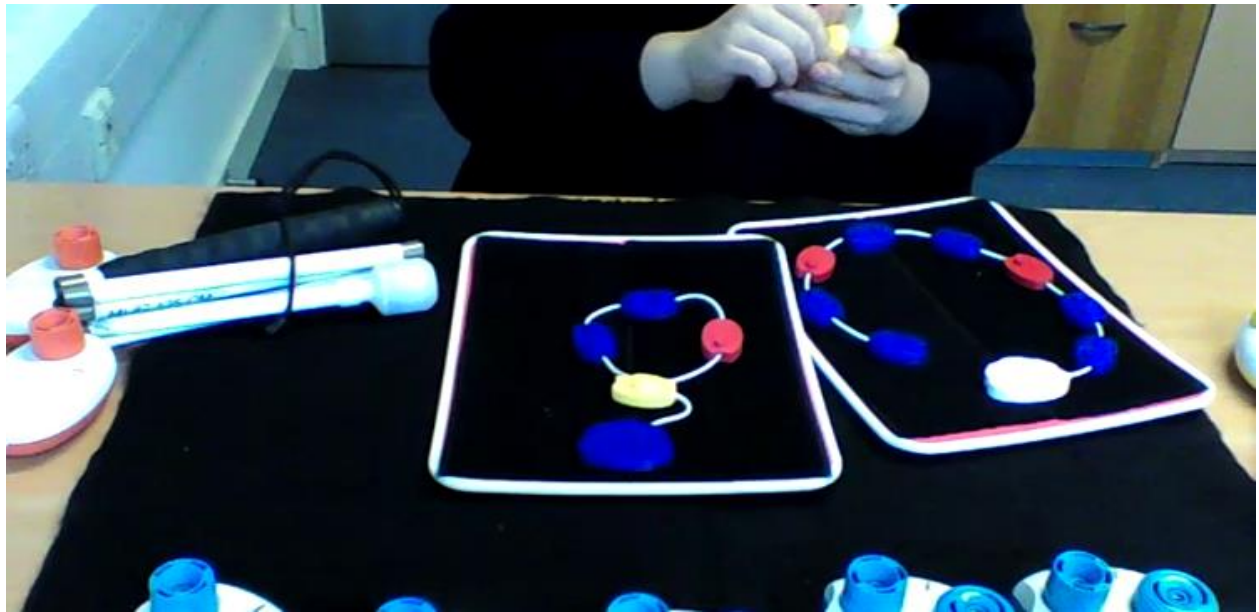


Figure 2: Turning the Dial on a Loop Pod

In the next activity Steven was given an algorithm for a nested loop program, the inner loop featured two sounds, with pauses after each sound and after the inner loop was a single sound. Steven read the algorithm and correctly identified that a loop pod was needed first and added one to his program. When reading the next part of the algorithm he said, “loop three times again, so we need another loop pod”. Steven started adding the play and pause pods to the inner loop and followed the sequence of pods to find the end to add the next pod. As he was adding pods to the inside of the loop, Steven said “I’m getting the hang of this!”. When closing the inner loop, Steven used one hand to hold the long wire and the other hand to follow the sequence to locate the end he needed to plug the wire into. When Steven listened to the complete program he said “hahaha, it’s really funny! Let’s play it again!”. After this he followed the completed program, starting with the outer loop and then following the inner loop in order of execution. Steven can be seen exploring the program in Figure 3.



Figure 3: Exploring the Complete Nested Loop Program

In the final activity of this session I played Steven the sound of a completed program that he needed to recreate. The program featured two claps followed by a stomp repeated four times and could be solved with a single or a nested loop. After listening to the program, he said “so that part sounds like a clap, two claps, one stomp”. I then asked him if he thought anything repeats and he said, “shall we start with the loop pod?”. When discussing how many times the sounds repeated Steven initially said three times, but after listening to it again he said “twelve... it sounds like twelve” (the total number of sounds produced). I then asked him how many times the pattern repeated, and as the recording played, he said “one... four claps... two stomps... three... four!”.

Next Steven started designing the algorithm and I asked him what should go inside the loop. Steven replied, “the clap, clap and stomp” (tapping his finger on the board as he named each sound). After adding these sounds I asked Steven what needed to go at the end of the algorithm and he replied “clap, clap again”, I asked him why we needed to loop and he said “because the sounds are repeated four times” and he still thought the sounds needed to be duplicated within the loop but was not able to explain why.

When Steven had started creating the actual program and added the loop, I asked him what he needed to add next and he replied “clap, clap, STOMP... no actually two claps”. Once Steven had completed building the program, he realised that it did not sound quite right and he went through a process of trial and error, changing the speed of each sound and playing the program back. Steven realised that a pause was needed in the program, however we had run out of time and had to end the session there.

Appendix 15 Analysis of Steven's Narrative

Sequence

In the first couple of his *perezhivaniya*, the exploratory procedures that Steven employed to examine programs and locate specific parts appears to be rather haphazard in approach. This resulted in him often adjusting a different pod to the one he intended. The absence of a clear exploratory procedure for working with sequences could be seen as a reflection of Steven's sense of sequence being in the early stages of development at that point in time. During this period, Steven expresses a mixture of frustration, motivation, and amusement. These observations would tie in with the literature that suggests that novice programmers often have difficulties in understanding the order in which statements are executed (Swidan et al., 2018). The suggestion of feeling along the pods from the start of the program seemed to trigger the beginning of the development of a more systematic exploratory procedure. Steven started to locate specific parts of programs by following the contours whilst counting the pods, which is similar to the contour following exploratory procedure identified in the literature (Lederman & Klatzky, 1987). The counting either took the form of a gesture such as tapping each pod or through private speech involving the audible counting of the pods. This exploratory procedure seems to indicate the further development of Steven's sense of sequence. Initially he uses the exploratory procedure inconsistently, however it becomes more consistent by the third *perezhivanie*. At which time he also seemed to be becoming more confident in working with sequences, making a number of statements which imply a feeling of motivation and confidence.

In his third *perezhivanie*, Steven continued using the exploratory procedure, however the only external manifestation was through contour following. He stopped displaying external signs of counting, however it not being externally perceivable does not imply that it is not still a part of this exploratory procedure for Steven. On a number of occasions, when introduced to a new programming concept, Steven resumed external manifestations of counting through gestures and/or private speech. Additionally, when Steven appeared to be particularly outside his comfort zone, he would seemingly forget to employ the exploratory procedure, resulting in him encountering similar challenges working with sequences that he encountered during the first two *perezhivaniya*. However, when reminded about his exploratory procedure, Steven would usually overcome these challenges quickly.

In later *perezhivaniya*, Steven often employed his exploratory procedure in the examination of programs which featured other constructs. For example, he would explore a sequence containing a loop without going inside the loop itself. This seems to be an indication of his sense of sequence within the main body of the program.

Repetition

In his fourth *perezhivanie*, Steven was introduced to the loop pod. When initially exploring it, he drew comparisons with the pause pod as both pods have just one dial. The exploratory procedure which Steven employed to examine loops involved him contour following in either a clockwise or counter clockwise direction. Although in Code Jumper the instructions are executed in a clockwise direction,

Steven's use of counter clockwise examination does not seem to be indicative of his understanding of the order of execution. For example, he sometimes followed a loop in reverse but then set the sounds in order or named the pods in order. Other times he explored clockwise but then set the pods in reverse order. As the loop in Code Jumper starts and finishes at the same place, perhaps conceptually for Steven the direction he follows them in is not important to his sense of repetition. Additionally, at times Steven wanted to place the loop pod after the instructions he wished to repeat, which could be related to loops in Code Jumper starting and ending in the same place. This could also be viewed as a sign that Steven's sense of repetition features a transitional theory that sees the parameters of the loop as occurring after the instructions, as is the case in DO UNTIL style loop structures.

Although Steven initially confused the loop pod with the pause pod, he quickly seemed to develop a close association between repetition and the loop pod. For example, on some occasions he seemed to need to physically hold or touch a loop pod in order to answer a question about repetition. This appears to suggest that the loop pod forms an important part of Steven's sense of repetition. Additionally, on a number of occasions from perezhivanie four up until the last episode, Steven produced a gesture which involved the closing of an empty loop on itself. This could potentially be interpreted as Steven's sense of repetition being linked to a loop pod that is closed.

In his fourth perezhivanie, Steven's sense of repetition seems to incorporate a transitional theory in which instructions that are outside the loop are also repeated. This ties in with the literature regarding novice programmers believing that adjacent code executes within the loop (Swidan et al., 2018). However, for Steven, this transitional theory seems to be revised after his fourth perezhivanie. There is evidence to suggest Steven developed other transitional theories in relation to repetition. For example, Steven made a strong connection between the number of sounds a program produces, the number of instructions within a loop and the number of repetitions. In certain situations, some of these numbers could be the same and in these cases this transitional theory would apply. Over time, this transitional theory seemed to evolve through Steven's experiences to see these numbers as linked but separate. On occasions, however, Steven would still use these numbers interchangeably even towards the end of his perezhivaniya.

Throughout his perezhivaniya, Steven drew upon previous perezhivaniya with Code Jumper in the development of his sense of programming concepts. Additionally, he also drew upon perezhivaniya from his wider experience outside the domain of computing. A good example of this is in his linking repetition to his prior experience with music. On numerous occasions from his sixth perezhivanie, Steven described a loop as a beat and beats as the number of repetitions; for instance he stated "it sounds like a music beat like you dance to normally... with eight beats". It does seem that the musical nature of many of the Code Jumper activities may have influenced this form of expression, however it backs up the idea that the tool employed in the learning process shapes the development of an individual's sense of a concept. Through the use of this particular tool, Steven was able to connect his prior concrete experiences with music to the programming concepts he was learning. Thus, enabling him to navigate between the abstract and the concrete in both directions.

Steven also made connections between repetition and his prior experiences with mathematics. When exploring nested loops, he was able to draw upon addition and multiplication in order to develop his sense of how nested loops work. He discovered that nested loops in effect multiply the number of sounds produced whereas sequential loops add. When I asked him how many times a particular sound in a program would play, he expressed his answer in terms of multiplication, stating “3 times 3 is 9”.

Another way in which Steven expressed his sense of repetition was through the use of gestures to indicate the number of repetitions. When listening to example programs that featured repetition, he would often make a repeated gesture that matched the number of repetitions. For example, sometimes he rocked to a beat or tapped his finger for the number of repetitions. For Steven, this appears to be a tool, but it also serves as a window into the continuing development of his sense of repetition as it is in evidence to a greater extent in later perezhivaniya. Steven also occasionally used private speech in a similar manner, saying or making the sounds that he wanted the loop to produce.

Selection

Steven is introduced to the selection pod in his ninth perezhivanie and I explained how it works in terms of a question. His sense of selection quickly began to manifest itself through his external speech when he described the conditions he planned to use to achieve his desired result; for example, he suggested that the condition “5 is bigger than 1” was needed. My framing of conditions in terms of questions also seemed to shape his sense of selection with him expressing the results of conditions as either “yes” or “no”. When asked to set the dials to achieve a certain outcome, Steven would often express his planned condition in terms of a question before setting the dials to match; this demonstrates a connection between his sense of selection and the dials on the selection pod.

At the start of his tenth perezhivanie, I asked Steven if he remembered what we used the selection pod for and he seemed to struggle initially, so I asked him what question the selection pod asked and he replied “is that number greater than the other number”. This external speech provides a window into his sense of selection and its link to questions for him. He continued to be able to suggest appropriate conditions during this perezhivanie, however he did struggle with the syntax of implementing selection with Code Jumper. With the limited time available, Steven was not able to gain much experience of creating original programs featuring selection and given more time it is likely he would have become more familiar with the syntax required.

Subroutines

On a number of occasions throughout his perezhivaniya, Steven appeared to interact with the play pod in such a way that could suggest he was viewing it in terms of a subroutine. Early on, when designing programs, Steven would sometimes place instructions next to each other rather than underneath, seemingly making them part of the same instruction. This would often be when two instructions completed a sentence and it seemed logical to place them together. However, on a couple of occasions when I asked Steven what he needed to say the next play pod to he would say both commands rather than just one. Later, when working with loops, Steven

was sure that a program with a loop containing three play pods would definitely make two sounds repeated three times. From looking at the video it seems clear that Steven was thinking of each of those three play pods as somehow triggering the two sounds to play, like a subroutine call for example. It appears that Steven's sense of the play pod enables him to view it as being both a single instruction and the encapsulation of a group of instructions.

Discussion

From analysing Steven's narrative, it is clear that his sense of the different programming constructs is shaped from a variety of influences. These include: the microworld, teacher talk and non-computing perezhivaniya. The microworld includes the learning tools, such as Code Jumper, and the pedagogical approach which is incorporated within them. On occasions, Steven needed to physically pickup and hold the loop pod in order to answer a question regarding repetition, demonstrating how the microworld shaped his sense of repetition. Teacher talk refers to the language used to introduce and explain concepts which fall beyond the confines of the microworld. The influence of teacher talk can be seen by looking at the way in which Steven developed his contour following exploratory procedure, which was seemingly triggered by my suggestion of following the wires. Finally, non-computing perezhivaniya indicates perezhivaniya which fall outside those relating to computing education. The influence of these is in evidence in the link Steven creates between repetition and music, with him referring to a loop as a beat the number of repetitions as beats. It is also in evidence in the way in which he links repetition to the mathematical concepts of addition and multiplication.

It is important to note that the link between repetition and mathematics was instigated through teacher talk as I used the concepts of addition and multiplication as a way of explaining the concepts of loops and nested loops. Additionally, the relationship between music and repetition may have been influenced by the design of the microworld, which is geared towards the creation of programs that produce music. The points raised above demonstrate that there are many factors involved in the development of an individual's sense of a programming construct. This makes it highly unlikely that any two people's sense of a construct would be identical.

Another consideration is the affordances of a physical programming languages in relation to traditional text-based programming. For example, the analysis highlighted the differences between the physical loop in Code Jumper and a loop as implemented in a text-based language. In Code Jumper, the loop is a physical loop, whereas in a text-based language a loop is represented as a sequential list. As a result, a loop in a text-based program starts and finishes in different places, whereas in Code Jumper they start and finish in the same place. It seemed important to Steven for a loop to be closed, it was almost as if it was not a loop if it was not closed. This leads me to question whether this may be important for some learners' development of sense of repetition and whether text-based languages facilitate this feeling of a loop being closed.

When teaching programming to novices, a single line of code is usually described as a single instruction, however, in fact in many cases, a single line of code can represent many instructions. For example, in the case of an inbuilt subroutine such as PRINT, the learner will type their print statement without seeing the instructions

inside the subroutine. The play pod in Code Jumper can represent both a single instruction or a group of instructions as it can play one or multiple sounds. Steven appeared to view the play pod as both a single instruction and a group of instructions at different times, perhaps facilitating the two-way movement between the abstract and the concrete. On some occasions, Steven seemed to want to solve a problem by using a play pod as a subroutine to encapsulate a group of instructions he wanted to repeat in place of employing a loop. This leads me to question, whether for Steven, subroutines as a concept is less challenging to master than repetition and whether he would have benefited from being introduced to subroutines first.

Summary

Throughout his *perezhivaniya*, Steven provided a window into the development of his sense of each programming construct through his use of external and private speech, gestures and exploratory procedures. The development of his sense of sequence is clearly demonstrated through his initial unstructured gestures, through to his employment of the contour following exploratory procedure. Additionally, he demonstrated the use of counting through both gestures and private speech. He used counting as a tool, both to locate the correct pod in a sequence and to keep track of the number of repetitions in a program he was listening to. When working with selection, Steven's sense of the construct was exhibited using external speech as he described conditions in terms of yes/no questions.

Steven's sense of repetition is closely linked with the physical loop pod. The physical loops, as seen in Code Jumper, take the form of real loops as opposed to loops in a text-based language which appear as a linear sequence of commands. This difference seems to have had an impact on Steven's sense of repetition, as it did not seem to matter to him which direction he explored a loop in as they start and finish in the same place. Additionally, there is a suggestion that for him the loop must be closed in order for it to actually be a loop. This makes me question whether the ability to close an actual physical loop may benefit some learners in the development of their sense of repetition.

For Steven, the play pod could both represent a single instruction or a group of instructions, seemingly facilitating the movement between the abstract and the concrete. In some tasks, it appeared preferable for Steven to use the play pod as a subroutine that could represent a group of instructions and on occasions he seemed to prefer to use this approach to the use of repetition. It is possible that, for Steven, subroutines would be less challenging to master than repetition.