The Hong Kong Polytechnic University
Department of Computing

Exploring Children's Usage on Tangible Computational Construction Platforms

Hands-on Learning through Functionality, Crafts and Stories

LAU Wing Yiu

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Philosophy

December 2010

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____

LAU Wing Yiu

Abstract

Many tangible computational construction platforms are designed with new technologies and children's learning theories to assist children in learning computational concepts. This thesis explores how children learn computational concepts through these platforms, by focusing on three categories of tangible computational platforms: tangible programming systems, computational toolkits for crafts making, and computational interfaces for story creation. Our rationale for choosing these three categories is motivated by the hands-on learning theories argued by three influential educators: Montessori, Froebel and Vygotsky.

Given this motivation, we design three dimensions of case studies: abstract thinking through functional blocks construction, creativity through crafts making and expression through stories creation, for analysis of children's learning of computational concepts vertically through three main categories of computational construction kits respectively. Each dimension of empirical studies obtains qualitative and quantitative results, and the results indicate that children have positive learning experience on each category of computational construction kits in different aspects.

For the study in building functionality, we use a tangible programming system to study how children understand the abstract concepts by building functional blocks to simulate functions of smart clothing and flows of story, as well as computational concepts such as looping and branching.

To study creativity through crafts making, three kinds of computational craft platforms are adopted as case studies. These are the Lilypad Arduino with TeeBoard platform, i*CATch apparel platform and i*CATch robotic platform. The focus is on crafts making in robotic and apparel domains, integrating traditional materials such as paper and cloth with the electronic devices. Through these three different design approaches of the computational craft platforms, we gain a deeper understanding of how children use computational platforms for creating crafts.

To study expression via story creation, a storytelling expression media is proposed: storytelling augmented with computational apparel for children to express

their stories and learn computational concepts through story creation. The i*CATch wearable computing platform is used as the storytelling media. The results show that computational apparel can be a media for children to tell a story, and indicate that there is room for children to improve their stories through more practice on storytelling and programming electronic device skills.

To gain a broader insight into computational toolkits for children, we propose five evaluation criteria for computational construction toolkits. Five factors include coupling of computational concepts, construction interfaces, domains of tasks, learners' characteristics and learning environments. The evaluation results reveal fundamental differences between the computational construction kits pertaining to children's learning of computational concepts. Finally, our thesis concludes with four suggested guidelines: determine places of learning, support diversity of domains, support simple and challenging tasks and provide hybrid programming environments, and these guidelines should be useful for designers and researchers who wish to develop a computational construction toolkit for computational concept learning.

List of Publications

Journal Article

Grace Ngai, Stephen C.F. Chan, Joey C.Y. Cheung, and **Winnie W.Y. Lau**. 2010. Deploying a Wearable Computing Platform for Computing Education. *IEEE Trans. Learn. Technol.* 3, 1 (January 2010), 45-55.

Grace Ngai, Stephen C.F. Chan, Vincent T.Y. Ng, **Winnie W.Y. Lau**, and Jason T.P. Tse. i*CATch: A Novice Friendly Platform for Wearable Computing. *Trans. Sys. Man Cyber. Part A* (In submission).

Conference and Workshop Papers

Winnie W.Y. Lau, Grace Ngai, Stephen C.F. Chan, and Joey C.Y. Cheung. 2009. Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students. In *Proceedings of the 40th ACM technical symposium on Computer science education* (SIGCSE '09). ACM, New York, NY, USA, 504-508.

Joey C.Y. Cheung, Grace Ngai, Stephen C.F. Chan, and **Winnie W.Y. Lau**. 2009. Filling the gap in programming instruction: a text-enhanced graphical programming environment for junior high students. In *Proceedings of the 40th ACM technical symposium on Computer science education* (SIGCSE '09). ACM, New York, NY, USA, 276-280.

Grace Ngai, Stephen C.F. Chan, Joey C.Y. Cheung, and **Winnie W.Y. Lau**. 2009. The TeeBoard: an education-friendly construction platform for e-textiles and wearable computing. In *Proceedings of the 27th international conference on Human factors in computing systems* (CHI '09). ACM, New York, NY, USA, 249-258.

Grace Ngai, Stephen C.F. Chan, Joey C.Y. Cheung, and **Winnie W.Y. Lau**. 2009. An education-friendly construction platform for wearable computing. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems* (CHI '09). ACM, New York, NY, USA, 3235-3240.

Grace Ngai, Stephen C. F. Chan, **Winnie W. Y. Lau**, and Joey C. Y. Cheung. 2009. A framework for collaborative eTextiles design - An introduction to Co-eTex. In *Proceedings of the 2009 13th International Conference on Computer Supported Cooperative Work in Design* (CSCWD '09). IEEE Computer Society, Washington, DC, USA, 191-196.

Grace Ngai, Stephen C.F. Chan, Vincent T.Y. Ng, Joey C.Y. Cheung, Sam S.S. Choy, **Winnie W.Y. Lau**, and Jason T.P. Tse. 2010. i*CATch: a scalable plug-n-play wearable computing framework for novices and children. In *Proceedings of the 28th international conference on Human factors in computing systems* (CHI '10). ACM, New York, NY, USA, 443-452.

Acknowledgements

Thank God for leading me back to The Hong Kong Polytechnic University, Department of Computing, eToy Laboratory to start my amazing research study.

I would like to express my deepest gratitude to my supervisor Dr Grace Ngai for her invaluable support, advice, insight, and guidance throughout this interesting and challenging research work that I cannot find it in other universities in Hong Kong even in this moment. It is a great pleasure and honor for me to be her student and to work closely with her. I will never forget her kind encouragement, patient support and tolerance of my last-minute work habit, and even the days she took her valuable time working with me overnight. She enlightens me not only on academic research, but also values and goals in life. She gave me many opportunities to explore the world through novel technologies and community services. I would also like to express my sincere apology to her because I made her disappointed. Anyway, Dr Ngai is the best teacher I ever met.

I am also very thankful for the valuable feedback and support of my co-supervisor, Dr Stephen Chan, and the other faculties who generously shared their knowledge and experience with me: Dr Alvin Chan and Dr Hong-va Leong from the Department of Computing; Mr Rémi Leclerc and Mr Nury Vittachi from the School of Design; and Dr Cecilia Pang from the University of Colorado at Boulder at the Department of Theatre and Dance.

I wish to thank Dr Leah Buechley from the MIT Media Lab and Professor Francis Lau from the Hong Kong University at the Department of Computer Science for their willingness to be my external examiners and their comments and interests on my thesis.

I would like to thank all team members for their support and encouragement. Special thanks to Ms Joey Cheung and Mr Sam Choi for helping to develop my supervisors' ideas in the TeeBoard and i*CATch platforms; Mr Jason Tse for helping to develop the ideas in the simulation of the tangible programming system; and Ms Cat Lai for helping to organize a series of workshops for case studies. I also would like to thank Dr Simon Lui for helping us to develop the sound library for

performance workshops and the undergraduate students who helped in the workshops, special thanks to Ms Esther Leung and Mr Li Li.

Many thanks to my co-supervisor, Dr Chan again for acting as liaison to the Conservative Baptist Lui Ming Choi Primary School. Thanks to the principal, Mr. Enoch Yeung and the teachers, Ms Lau-tim Ho and Ms Lai-yee Ho for allowing us to work in their classrooms and to conduct user studies with their students.

Thanks to Dr Mable Chan for giving me comments on my English writing for my thesis.

I am grateful to all my teachers, especially to Professor Helen Meng and Dr Y.Y. Lo for mentoring me at the beginning of my research journey. Thanks to all my friends, especially to Kelan, Holly, Peggy and Iris for listening to my happiness and sadness.

Finally, I would like to sincerely thank my family for their understanding and tolerance. I know that I have not spent enough time on them over the years.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Current Computer Science Education

One of the core parts of computer science education is learning programming languages. Programming language is a language, which is written as a set of human understandable instructions, which can be translated by a compiler or interrupter into machine codes that can be understood by a computational device, such as a computer or a robot. The set of instructions can be a series of mathematical expressions or English-like statements. Even though the high-level programming languages, such as Pascal, Java and C, are designed to be more easily understood by humans, they are still too abstract for humans to grasp without any training.

Recently, computer science and engineering degrees worldwide are suffering from the same problem of declining student enrolment and interest [46, 87, 127]. As far as we know, one main reason is that computational concepts are too abstract for students to understand. As a result, students are relatively losing interests and confidence in pursuing computer science studies [1, 24]. To combat the decreasing enrolment of science and engineering majors in university, more science and technology outreach courses are held for middle school students. Basically, these courses aim to 1) motivate students' interests in science and technology, 2) encourage them more likely to take science and technology subjects in the future, and 3) in the long run, help increasing the enrolment rate in science and engineering degrees in university.

These outreach workshops have contributed many innovative methods to teach computer science to children. One of the biggest problems with teaching programming is that learning the correct syntax often interferes with the learning of programming concepts, making it difficult to learn. Hence, most outreach workshops use graphical programming languages, such as Alice [2] and Scratch [111], to teach programming concepts. The colorful graphics and animations attract children, and drag-and-drop icons address the syntax problem, but there is still the question of concretizing the abstract concepts.

1.2 Learning Computational Concepts from Tangible Manipulatives

According to the hands-on learning theories proposed by the educators such as Froebel, Montessori, Vygotsky and Piaget, children learn through their senses and physical or social activity, and that physical or social interaction is a critical learning factor in the child's cognitive development [48, 92, 93, 97, 129]. To simplify abstract computational concepts, some computer science researchers incorporated this hands-on learning theory into the computational manipulatives for children learning abstract computational concepts. In the 1970s, Seymour Papert was one of the first researchers who extended Piaget's constructivism into constructionism. Piaget's theory of constructivist learning explains the development of knowledge which is from an interaction between their experiences and their ideas. Papert's constructionist learning focuses on learning through actively constructing objects in the real world, this help us build mental models. Papert applied his theory to develop the LOGO programming language [83, 96] with using simple commands to control a physical turtle. In this system, the output was tangible, but the input programming framework is still abstract.

For almost 20 years, there was no significant breakthrough in computational manipulatives or tangible learning systems until the 1990s. First example was the AlgoBlocks system (developed in 1993) [120] which took the opposite approach: it consisted of a collection of physical computational building blocks that controlled a virtual submarine on the computer screen. Another important example was the best known LEGO Mindstorms platform (commercialized in 1998) [90] which consisted of an intelligent brick and a set of sensors and motors allowed children to construct and program their own robots. In 1997, one year before the appearance of the Mindstorms platform, Ishii proposed a "Tangible User Interface" (TUI) to represent digital information [67]. After these works, more researchers started to rethink the development of tangible computing systems for learning. With the rapid development of technology, better hardware and toolkits such as RFID and microcontrollers are available, which expand the possibilities of the forms of tangible systems. As a result, a variety of tangible construction kits for learning of

computational concepts have appeared, such as Tern [56], which is a tangible computer language developed by a set of wooden bricks with a webcam for controlling robots; LilyPad Arduino [80], which is a set of sewable electronic components for learners to build the soft and interactive fashion; StoryRoom Kit [91], which provides a space for children to create their interactive story by physical interactions with sensors and activation of actuators. Outreach workshops [19, 20, 56, 70, 77] also start to utilize these tangible construction kits to teach abstract computational concepts and engage children in learning computer science and technology.

1.3 Motivation

In the late 1960s, with limited technology, researchers could mainly develop some physical computational platforms for programming robotic movements, such as LOGO turtle [83]. Along with the rapid development of technology, learning manipulatives are not only made of wood, plastic or fabric, but also integrate computational elements inside. These advanced computational embedded systems and ubiquitous technologies provide more opportunities for researchers and designers to develop various tangible computational platforms. As a result, more and more novel tangible computational platforms are developed. This scenario motivates us to find a way to explore how children learn computational concepts through these tangible platforms. We find that there are three well-known educators' constructivist learning approaches which are similar to the design features of the current tangible computational construction platforms. Thus, these three educators' learning theories are adopted to divide the current tangible computational construction platforms into three categories, this help us to understand the characteristics of the current tangible platforms and to reveal the possibilities and opportunities of tangible computational platforms. The three influential educators' hands-on learning ideas are described as follows:

- **Maria Montessori** who created a set of learning materials called Montessori materials [92], which include various dimensions of blocks designed for developing children's sensory capabilities through exploration.

- **Friedrich Froebel** who designed a set of educational play materials called Froebel's gifts [47], which include geometric building blocks designed for children to explore the physical world by construction.
- **Lev Vygotsky** did not create any learning artifacts, but he emphasized that children learn through social interactions and believed that children develop their abstract concepts to understand the world through play [129].

And the three main categories of the current tangible computational construction platforms are:

- **Tangible programming systems** usually consist of a collection of bricks or modules, designed for hands-on exploration of abstract computational concepts. Children can learn through the construction of the bricks to program some kinds of simulations or robot movements. For example, Flow Blocks [134] is designed for exploring mathematical concepts such as counting, and Tern [56] is designed for exploring computational concepts such as looping by manipulating robot movements. The objective of these computational block building systems is similar to Montessori's idea of learning abstract concepts by building blocks.
- **Computational toolkits for construction** provide a set of components for building physical models or making crafts, enabled children to program their own computational artifacts. For example, LEGO Mindstorms [90] allows construction and programming robots, and Lilypad Arduino [80] enables design and programming crafts with sensors and actuators. The objective of these programming real world structures is similar to Froebel's view of exploring the physical world by construction.
- **Computational interfaces for story creation** provide an interface for storytelling through a set of sensors and actuators by physical interactions or making props or decorations. Children can express their emotions or stories through programming sensors and actuators. These activities encourage children to have social interaction and imaginative plays. For example, StoryRoom [91] provides a space for programming and integrating the technology into a story, and PicoBoard [100] creates interactive stories by

programming sensors. The achievement of these story creations is similar to Vygotsky's philosophy of understanding the world through play.

These three categories do not cover all aspects of the existing tangible computational interfaces, but the majority of them. On the other hand, some tangible computational construction platforms can be sorted into two categories, such as Tobopo [104] which combines a tangible programming system with a robotic construction kit. Again, the purpose of this kind of categories is to help us to analyze and understand the characteristics of the majority of tangible construction toolkits for children to learning computational concepts.

Most studies of tangible computational construction platforms received positive feedback that may be related to the novelty of the systems. We are interested in which category of these three categories of tangible computational platforms are more efficient for children to learning computational concepts or what kind of characteristics of tangible toolkits are more important to children's learning. In general, most researchers focus on the evaluation of their own platforms with a set of hypotheses vertically, such as tasks achievement, learner engagement, and interaction quality [20, 89, 134], in which few of them have tried to compare their platforms to other platforms horizontally. Even in a comparative study, they may have comparisons between tangible and graphical interfaces of similar platforms [58], but they may have few considerations on the learning benefit among different types of platforms. We believe that each category of construction kits has its own merits for children to use for learning, such as coupling of concepts which may be related to knowledge assimilation; user interfaces which may be related to collaborative learning; learning domains which may be related to engagement. If the kits are used by children with different backgrounds, the original design of the kits may be not fully utilized by children, or some unexpected usage of the kits may be discovered by children. Therefore, we aim to compare and analyze how children use different categories of computational construction platform, and decide whether to make any tradeoff for a design of a new computational platform for learning.

In this thesis, the three dimensions of case studies are developed corresponding to the three famous educators' views and the three categories of tangible

construction platforms mentioned in the above paragraph of this section, by observing children how to use and interact with each categories of the computational construction kits vertically with several various factors:

- **Abstract thinking through functional blocks construction** - observes how children understand the abstract concepts through using *tangible programming systems* by building function blocks to simulate functions of smart clothing and story flows, as well as computational concepts such as looping and branching.
- **Creativity through craft making** - examines how children construct their robots and their clothing by using different *computational interfaces for construction* for making crafts in robotic and apparel domains with integrating traditional materials such as paper and cloth.
- **Expression through story creation** - explores how children express their stories by programming on different *computational interfaces for story creation* and their performance methods.

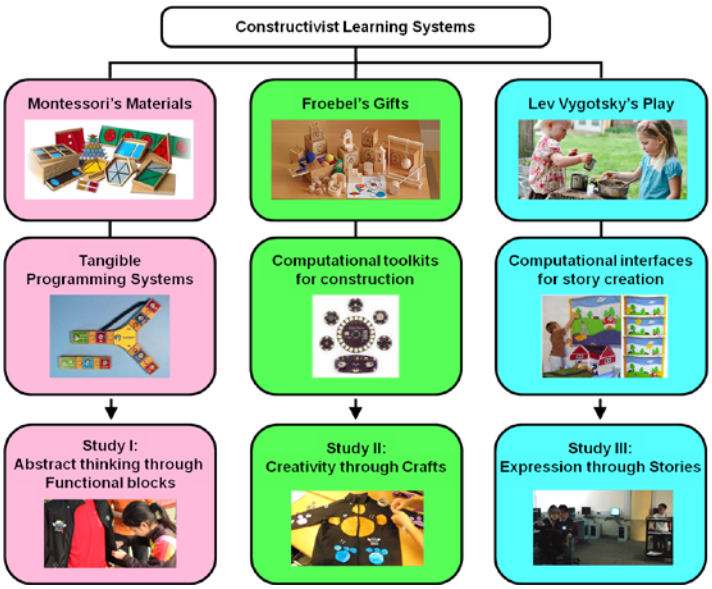


Figure 1.1. The categorization of the educators' learning systems and the current tangible computational systems, and the corresponding studies on the tangible computational construction platforms

After conducting the three dimensions of studies on tangible computational construction platforms, we do an overall evaluation of the learning of computational concepts. Based on the review of learning theories and conceptual frameworks of

tangible interfaces (more details introduced in chapter 2), and our experience from the three dimensions of studies, we consider five factors to further analyze how children learn through these three categories of tangible construction platforms horizontally. The five factors are:

- **Coupling of computational concepts** – focuses on how abstract computational concepts are coupled with physical or expressive representation. The representation may be physical blocks, physical actions or stories. Computational concepts shall be easier to learn if the concepts are coupled with intuitive representations.
- **Construction interfaces** – refers to the perceived affordance and the collaborative learning support. If the construction interface brings many difficulties to the learners, they will lose interest easily. Also, learners working together should aid effective learning.
- **Domains of tasks** – refers to the learning engagement in the domains of tasks (e.g. robots, e-fashion, and storytelling). The learning engagement may consider the children's responses before, during and after the activity. This factor may affect learner's learning interests and creativity.
- **Learner's characteristics –** highlights the characteristics of learners in terms of gender, age, knowledge and learning style. Different learners' characteristics are suited to use different kinds of interfaces and domains of tasks.
- **Learning environments** – focuses on the flexibility of the platforms for children to learn in different places, such as physical environment like classrooms, studios and exhibitions; virtual environment like online communities. Different learning places provide different learning experience for children. The factor may impact on learner's active learning and social interaction.

Finally, we base on our analysis to generate guidelines for design of a new tangible computational learning platform which should be useful for other researchers, educators and designers.

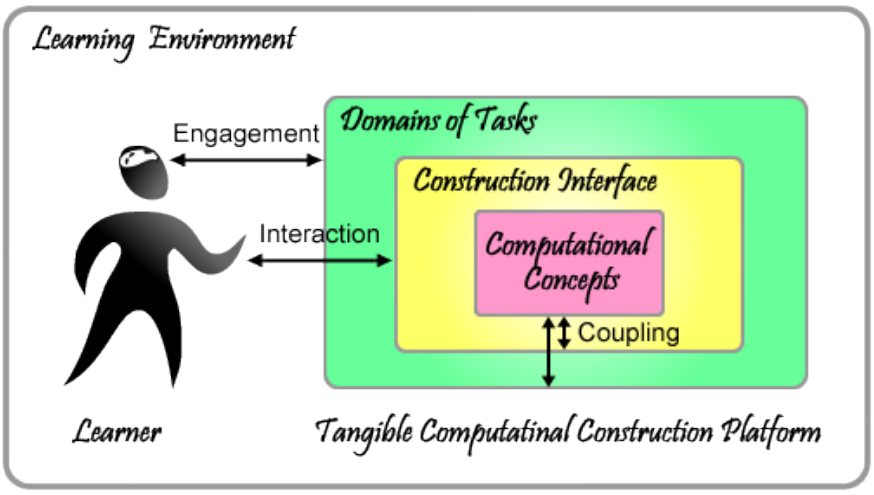


Figure 1.2. The relationship of the five evaluation factors: coupling of the computational concepts, construction interfaces, domains of tasks, learner characteristics and learning environments

1.4 Contributions

This thesis has the following contributions:

Empirical Evidence on the Learning of Different Categories of Tangible Computational Construction Kits by Children

During the process of this research, we performed a great number of case studies through workshops and activities to observe how children master computing concepts via tangible computational toolkits. The results of this study contain data and knowledge on how children learn to program and to solve problem solutions using a tangible kit.

Evaluation Factors for Tangible Computational Construction Kits

In this thesis, we suggest the use of five factors to evaluate three categories of computational construction kits. These are: coupling of computational concepts, construction interfaces, domains of tasks, learners' characteristics and learning environments. These factors may also be suitable to analyze other types of computational construction kits.

Guidelines for the Design of Tangible Computational Learning Platforms

This thesis also proposes four guidelines based on the empirical data of children's

usage on computational construction platforms. The guidelines include: determine places of learning, support diversity of domains, support simple and challenging tasks and provide hybrid programming environments. These guidelines should be helpful in giving directions to designers and researchers who wish to develop a tangible computational construction toolkit for computational concepts learning.

1.5 Thesis Overview

The remainder of this thesis is organized as follows:

Chapter 2 Background and Related Work

This chapter describes the learning theories, and reviews the frameworks specific for tangibles and learning, followed by a description of tangible computational construction kits.

Chapter 3 Abstract Thinking through Functional Blocks Construction

This chapter introduces tangible programming environments that are designed for children to learn abstract computational concepts. Case studies are used to analyze how children use the tangible programming toolkits and compare with the graphical approach. Finally, we also examine the effectiveness of tangible programming systems in helping children understand the concepts in different learning environments.

Chapter 4 Creativity through Craft Making

This chapter describes programming courses that focus on robotic with crafts making which is different from traditional tools with a well-designed construction platform, and wearable computing with apparel design. We explore how children use and learn through this kind of approach by the working process, program design and final products, and analyze the pros and cons of this approach.

Chapter 5 Expression through Story Creation

This chapter explores how children combine their low-tech individual storytelling or group performance methods with high-tech smart clothing components to express

their ideas and thoughts through creating stories and giving performances. We investigate how children create stories and write programs in this new approach.

Chapter 6 Analysis of the Design of Tangible Computational Construction Kits

This chapter suggests five factors to analyze the learning properties of each stream of the computational construction kits. It emphasizes the discussion of the strengths and limitations of each kind of the computational construction kits and the balance of some design factors on a new computational construction kit. Finally, we generate some design guidelines for a new learning kit.

Chapter 7 Conclusion

It summarizes the thesis and presents a framework to explore the computational construction kits learning through building functionalities, crafts making and stories expression. It also addresses future directions for design and development of the toolkits for learning computational concepts.

Chapter 2 Theories and Related Work

To develop a tangible computational construction platform, it often involves different scope of elements, including child cognitive development, constructivist learning, computational concepts and tangible user interfaces. To review constructivist learning theories and the current tangible computational construction platforms, it assists us in understanding how the features of tangible systems facilitate children's learning. Besides that, more tangible systems are developed, and more theoretical frameworks are proposed to conceptualize the features of tangible systems in terms of representations and spatial, which are often useful for developers to analyze the design features of tangible systems. The main purpose of the tangible computational construction platforms is to arouse children's interest in learning programming, so it is also necessary for us to study programming paradigms to aid the design of the toolkits and the syllabi in case studies. This chapter summarizes the relevant learning theories, modern tangible conceptual frameworks, evaluation approaches and program paradigms; and it also examines some computational toolkits based on the three main perspectives of the constructivist learning. This review should be helpful for us to understand the background of tangible computational construction toolkits and to design the research approaches in our study.

2.1 From Constructivist Learning to Tangible Computational Construction Toolkits

This section provides an overview of traditional constructivist learning systems to the current tangible systems, focusing on tangible computational construction toolkits. It first describes the approach of constructivist learning and the current tangible computational construction toolkits respectively, and then discusses the similarity of the features of the three traditional constructivist learning systems and the three categories of current tangible computational construction toolkits.

2.1.1 Constructivist Learning

Constructivist learning theory basically argues that the best approach of learning is hands-on learning that means learning from experience, from sensation and reflection. Formalization of constructivist learning theory is generally attributed to Jean Piaget in 1937. Before the work of Piaget, some educators also proposed the ideas of hands-on learning in different directions. For example, Froebel, Montessori and Vygotsky, who are some of the significant educators, proposed their own approaches: 1) Froebel's approach focuses on construction and design; 2) Montessori's method focuses on conceptual manipulation; and 3) Vygotsky's philosophy focuses on the group of role play. The extensions of constructivist learning theory influenced many researchers and designers for building learning tools, for example, Seymour Papert who is the founder of LEGO Mindstorms.

Jean Piaget

Jean Piaget (1896-1980) proposed mechanisms of learning – how learners develop cognitive abilities – called constructivist learning [97]. His constructivism proposes that learners cannot be given information which they immediately understand and use. Instead, learners must construct their new knowledge through experience. There are two processes of learning: assimilation and accommodation. Assimilation is a process of perceiving new objects or events in terms of existing schemas or operations. Accommodation is the process of changing internal mental structure of the external world to fit new experiences.

Constructivism is not a particular pedagogy rather a theory describing how learning happens. It is regardless of how learners use their experiences to understand a lecture or follow the instructions for building a physical model. However, constructivism is often associated with pedagogic approaches that promote active learning, or learning by doing.

Friedrich Froebel

Friedrich Froebel (1782-1852) developed a specific set of "gifts" [47] - physical objects, such as balls, blocks, and sticks, for children to use in the kindergarten.

Froebel designed carefully these gifts into three forms: nature, beauty, and knowledge, which help children to learn about colors, shapes and spaces through the design and construction of the models by the gifts. For example, gift 6 contains a set rectangular blocks which can be used to learn area and volume through constructing various forms of buildings; gift 7 is a set of paperboard pieces which can be used to learn various shapes and symmetry through creating freestyle decorations. Froebel's gifts were eventually distributed throughout the world, deeply influencing the design of the construction kits to children.

(a) (b)

Figure 2.1. Two examples of Froebel's Gifts: (a) Gift 6 is a set of blocks used to create buildings. (b) Gift 7 is a set of paperboard pieces used to create decorations. [47]

Maria Montessori

Maria Montessori (1870-1952) extended Froebel's ideas, developing a set of materials [92], such as the cylinder blocks, the pink tower and the board stair. The design of the materials considers a "control of error" feature. For example, the pink tower has different sizes of cubes, which helps children to know the tower cubes are in the right or wrong order. Montessori believed that the materials enable children to learn through personal investigation and exploration [93]. Her idea inspired many schools in which manipulative materials play a central role. Montessori sensorial materials also become the ancestors of many modern toys, such as puzzles and stacking toys.

(a) (b)

Figure 2.2. Two examples of Montessori's Materials: (a) The knobless cylinder contains a set of cylinders with varied height or width for learning the concepts of size. (b) The pink tower has different sizes of cubes. [92]

Lev Vygotsky

Lev Vygotsky (1896-1934) proposed that children's learning are not only through the physical world, but also through the interactions among people in relation to the world [129]. His view on learning through social interactions is extended to the learning through imaginative play [128]. He argued that play creates an imaginary situation which allows children in thinking about the abstract meaning from the objects in the world, and contains rules for behavior which stretches the logical skills. For example, children can imagine a banana as a phone or a stick as a horse; or they can act as a doctor in a role play.



(a) (b) (c)

Figure 2.3. Three pictures illustrate the Vygotsky's theory: (a) Imagining a banana as a phone (b) Imagining a stick as a horse (c) Acting as a doctor

2.1.2 Current Computational Construction Kits

Many researchers believe that children learn through sensory, physical objects and social interactions, so many programming environments for children are created specifically to interact or control tangible objects or environment, instead of a visual display only. In this section, we summarize some examples of the tangible computational construction platforms into three groups: *tangible programming systems*, *computational toolkits for construction* and *computational toolkits for story creation*, corresponding to the philosophy of hands-on learning from the three key educators, Froebel, Montessari and Vygotsky.

Tangible programming systems

FlowBlocks

FlowBlocks [134] are a set of physical blocks embedded with electronic devices. Blocks are connected to each other by magnetic connectors. There are four types of components to FlowBlocks: Paths, Generators, Rules and Probes. Children can learn

through arranging the blocks in different patterns to simulate different causal structures. For example, a light signal is sent through the blocks to create a visible chain reaction of “moving lights”. There is another example: a probability slider is set with 50% probability at the middle of the two loops to control the light which passes through right or left and displays each pass on the display probes to show a probability measurement.

Figure 2.4. A sample simulation: Two loops with 50% probability measurement [134].

Tern

Tern [56] is a tangible programming language for controlling robots. Its design emphasizes the use of inexpensive and durable parts with no embedded electronics or power supplies. Students create programs in offline settings - on their desks or on the floor - and use a portable scanning station to compile their code. Users connect wooden blocks shaped like jigsaw puzzle pieces to form program flow chains.

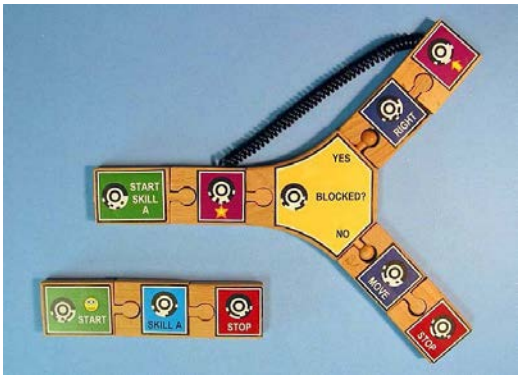


Figure 2.5. A sample program of Tern which contains condition, loop and subroutine constructs [56].

Computational toolkits for construction

LEGO Mindstorms NXT

The NXT robot [90] consists of an intelligent brick which can control other connected motors and sensors to make it come alive and to perform different

operations. Users can use the Lego parts to build a car or a robot and program it to perform different tasks. The program can be written in the NXT graphical programming environment to control NXT intelligent bricks and corresponding motors and sensors. A user constructs a program by dragging and dropping block icons. Each block represents a command and a pair of blocks represents a condition or a loop. It includes the data wires that show the data flow from block to block.

(a) (b) (c)

Figure 2.6. (a) The NXT brick which can be programmed to control the connected motors and sensors. (b) An NXT robot (c) The NXT programming environment [90]

Lilypad Arduino

The LilyPad Arduino [80] is a system for learners to build their own soft and interactive wearables by sewing a microcontroller, sensor and actuator modules together with conductive thread and writing a program to control the interactions among electronic modules. The program can be written in the text-based Arduino IDE [6], the icon-based programming environment Amici [7] or the hybrid text-graphical programming environment i*CATch IDE [61].

(a) (b) (c)

Figure 2.7. (a) A set of LilyPad Arduino electronic components: microcontroller, light sensor, accelerometer, buzzer, vibration motor, LED and battery case [80] (b) An interactive handbag with using LilyPad Arduino [81] (c) Arduino IDE [6]

Computational interfaces for story creation

StoryRoom Kits

StoryRoom Kits [91] are designed for children aged 4 to 6 to facilitate them to build interactive stories by providing sensors and actuators for them to create some rules as programs. These sensors and actuators can be used to augment everyday objects, such as chairs or teddy bears. For example, a child can combine a sensor, an actuator, and a prop into a magic programming wand.

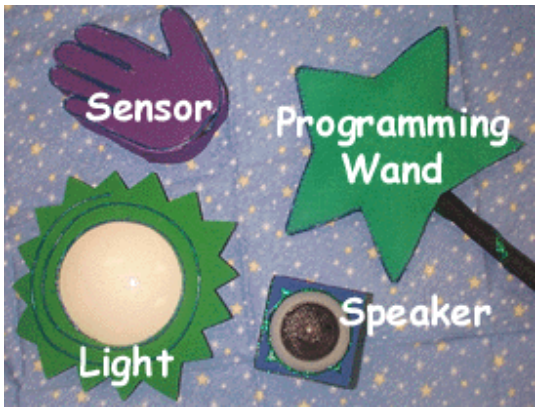
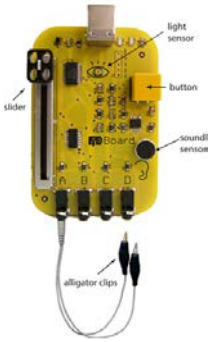


Figure 2.8. An example of physical programming devices like above: children can squeeze the purple hand to make the pink light turn on or to program the mouth shaped speaker to say whatever they want [91].

PicoBoard

The PicoBoard is a built-in sensors board, which enables the program to interact with things in the real world: Pencils, paper and water on the Scratch projects [100]. Scratch [111] is a kind of graphical programming environment for creating



(a) (b) (c)

Figure 2.9. (a) PicoBoard can be programmed to interact with Scratch Project [100]. (b) Scratch programming environment [111] (c) An example of a custom sensor: the clips are attached to a pair of home-made bracelets used for the wrists touch detection [100].

animations and interactive stories. For example, children can write a Scratch program to control an action of sprite by the sound sensor on the PicoBoard: when there is a loud sound, a sprite changes its color. The PicoBoard also contains a USB cable and four sets of alligator clips which are used to measure the electrical resistance in a circuit. Thus, children can use the alligator clips to build different kinds of custom sensors. For example, the clips are attached to a pair of home-made bracelets for the detection of the wrists touch.

2.1.3 Summary of the categories from traditional learning systems to current tangible computational construction platforms

Previous two subsections describe the features of traditional learning systems and some examples of current tangible computational construction toolkits. This subsection highlights the similarity of the features of the traditional and the current tangible learning systems into three categories respectively (see Table 2.1, Table 2.2 and Table 2.3). The highlighted features basically refer to three perspectives: physical materials, construction process and learning objectives.

Montessori's Materials	Tangible Programming Systems
A set of materials, e.g. cylinder blocks, cubic blocks	A collection of functional bricks or modules (maybe with embedded electronic devices)
With a "control of error" feature	Usually no syntax error
Designed for developing children's sensory capabilities through personal investigation and exploration	Designed for hands-on exploration of abstract computational concepts
E.g. The board stair, the pink tower	E.g. Flow blocks, Tern

Table 2.1. A summary of the similar features of Montessori's materials and tangible programming systems

Froebel's Gifts	Computational Toolkits for Construction
The physical materials include balls, blocks, paperboard pieces and sticks	A set of components consists include a microcontroller, actuators, sensors and some materials like blocks or cloth
Used to create freestyle designs with beauty and structure in child's creations (e.g. houses, decorations)	Used to build physical models (e.g. robots) or make crafts (e.g. light-flashing bags)
Learning arithmetic concepts through introducing different sizes of blocks or shapes	Learning computational concepts through programming their own computational artifacts
E.g. Gift 6, gift 7	E.g. LEGO Mindstorms, LilyPad Arduino

Table 2.2. A summary of the similar features of Froebel's gifts and computational toolkits for construction

Vygotsky's Theory of Play	Computational Interfaces for Story Creation
Any objects, social rules in the world	An interface with a set of sensors and actuators for storytelling
Through giving meaning to objects, e.g. a banana as a phone, a stick as a horse	Through programming artifacts to create physical interactions or interactive props
To reach beyond a child's average age, level of skills or knowledge through play	Learning computational concepts through programming their stories
E.g. Use an banana as a phone, take a stick as a horse, act themselves as a doctor	E.g. StoryRoom Kits, PicoBoard

Table 2.3. A summary of the similar features of Vygotsky's theory of play and computational interfaces for story creation

2.2 Other Concepts related to Learning

Besides the idea of constructivist learning has a deep impact on children's learning, there are other learning concepts which also have a significant influence on children's education. The concepts include child cognitive development, zone of proximal development, collaborative learning and learning styles. The details are described as follows.

2.2.1 Cognitive Development

Jean Piaget (1896-1980) proposed four stages of cognitive development: sensorimotor, preoperational, concrete operational and formal operational (see Table 2.4) [98]. He theorized that children's knowledge is constructed through "concrete operations" before moving on to "formal operations" that is characterized by purely abstract thinking.

Age	Period	Characteristics
0-2	Sensori-motor	The infant learns to differentiate between itself and other objects within its environment, learning the difference between "me" and "not me".
2-4*	Pre-operational thought	The child is still very egocentric, but now classifies objects in simple ways - particularly by individual important features.
4-7*	Intuitive	The child can classify things more generally, but is not aware of the classes that he or she uses.
7-11	Concrete operations	The child can use logical operations, such as reversal, deliberate classification and serialization.
11-15	Formal operations	Now things become more conceptual as the child is able to think in terms of abstract ideas.

** Age 2-4 and age 4-7 can be grouped into one stage.*

Table 2.4. The stages of cognitive development [98]

Piaget extended his cognitive development into play theory and classified it into four categories [99]:

- *Sensory-motor play* - Child repeats a physical activity, e.g. running and climbing.
- *Fantasy play* - Child mentally represents realities that are not present, e.g. role play.
- *Construction Play* – Involve accidental learning emerging from symbolic play, e.g. Brick building and clay modelling.
- *Game Play* – Game with rules, e.g. Board games and card games.

Play provides a relaxed atmosphere in which learning can easily occur. However, play is not the same as learning: cognitive development requires both assimilation and accommodation, while play is assimilation with or without accommodation.

Piaget's stage of cognitive development indicates that children aged 7 to 15 who are able to learn logical operations through concrete materials, and therefore our target subjects should be able to learn abstract computational concepts if we make it into concrete materials such as bricks. Piaget's play theory points out the play which is not a must to require accommodation, so the tangible computational construction platforms as a kind of learning tools are important to support both assimilation and accommodation.

2.2.2 Zone of Proximal Development

Lev Vygotsky advocated a concept called "Zone of Proximal Development" (ZPD) that is the distance between the actually ability of a child to finish a task independently and the potential ability of a child to solve a problem under adult guidance or in collaboration with peers [129]. The idea is like "scaffolding" which builds up a child's new knowledge by the assistance of an adult [131]. This concept pushes more children's activities played with peers and adults, not played individually. This concept has influenced many researchers and teachers to provide children with experiences by encouraging and advancing their individual learning. The benefit of this concept also influences us to organize children to work with their peers and be assisted by our student helpers during the workshops.

2.2.3 Collaborative Learning

Collaborative learning is a situation in which two or more people learn something together [32]. This concept extends from the Vygotsky's theory of zone of proximal development [129] that addresses the social nature of learning of a child interact with adults or peers. Collaborative learning is more likely to occur among people with similar level of knowledge than among people with different levels such as a teacher and a student. This concept inspires us to have a criterion which is to determine the interfaces of tangible computational construction platforms whether support children to work collaboratively to obtain the benefit of learning.

2.2.4 Learning Styles

Learning styles are various preferred approaches of learners to perceive knowledge. Proponents of learning styles encourage teachers to adapt various teaching methods to best fit each student's learning style that help students learn most effectively. There are some well-known models and theories related to learning styles which are helpful to identify learner's preferred ways to learn and develop. For example, Jung's theory of psychological types [63], Kolb's experimental learning model [74], Fleming's VARK model [44], Gregorc's mind styles model [53] and Gardner's multiple intelligences [50]. This concept drives us to analyze the design of tangible computational construction platforms whether support different styles of learners to achieve a diverse population in science and technology disciplines.

Jung's Psychological Types

Jung's theory of psychological types [63] is to classify people into four kinds of functions of consciousness: two perceiving functions – sensation and intuition; two judging functions – thinking and feeling. These four functions are further modified by two main attitude types: extraversion and introversion. As Jung believed that the dominant function characterized consciousness, its opposite will tend to be repressed and to characterize the functioning of the unconsciousness. To give a complete description of a psychological type, the function and attitude type should be combined, and therefore the eight psychological types are: extraverted sensation;

introverted sensation; extraverted intuition; introverted intuition; extraverted thinking; introverted thinking; extraverted feeling; introverted feeling. This theory is not exactly used to describe the styles of learners, while the concept of learning styles is rooted in the classification of psychological types.

Kolb's Experimental Learning Model

Kolb's experimental learning model [74] is a cyclical model of learning with four stages: concrete experience, reflective observation, abstract conceptualization, active experimentation. Learners can enter the cycle at any point depending on their particular preferred learning style, but must follow each stage in sequence for successful learning to take place. Each learning style was based on two of the learning cycle stages. The learning styles are as follows: divergers (concrete experience and reflective observation) are good at thinking deeply and coming up with multiple possibilities of ideas; convergers (abstract conceptualization and active experimentation) are good at making practical applications of ideas and using deductive reasoning to solve problems; accommodators (concrete experience and active experimentation) are good at actively engaging with the world and actually doing things rather than thinking only; assimilators (abstract conceptualization and reflective observation) are good at creating theoretical models by means of inductive reasoning rather than taking practical actions.

Fleming's VARK Model

Fleming's VARK model [44] is an extension of concept of VAK (visual, auditory and kinesthetic) which comes from neuro-linguistic programming developed by Bandler, R. and Grinder, J. [11] and the Dunn and Dunn's VAK model [36]. Fleming improved the VAK model and divided the visual learning component into two parts: a symbolic aspect (represented as visual) and a text aspect (represented as read-write). Fleming VARK learning model states that each learner is inclined to one of its four styles to learn best. Visual learners learn best by seeing things in charts and diagrams. Auditory learners learn best through listening lectures. Read-

write learners learn best by reading and writing in text. Kinesthetic learners learn best by doing and by using their sense of touch.

Gregorc's Mind Styles Model

Gregorc's mind styles model [53] provides an organized way to describe how the mind works. In this model, there are two perceptual qualities: concrete and abstract; and two ordering abilities: sequential and random. Concrete perceptions enable learners to register information directly through their five senses: sight, smell, touch, taste, and hearing. Abstract perceptions allow learners to visualize, imagine, to conceive ideas, to understand some concepts that they can't really see. Sequential ability allows learners' mind to organize information in a linear, step-by-step manner. Random ability enables learners' mind to organize information by chunks, and in no specific order. All the perceptual qualities and the ordering abilities are present in each learner, but learners usually tend to use one of the combinations of the strongest perceptual and ordering ability more easily. The four combinations are: concrete sequential, abstract random, abstract sequential and concrete random.

Gardner's Multiple Intelligences

Gardner's theory of multiple intelligences [50] is a cognitive model to understand learner's intellectual ability. This model includes some of the VARK modalities as "intelligences" and extends that list to at least five other dimensions. The idea of multiple intelligences has changed the view of traditional schools which mainly focus on verbal-linguistic and logical-mathematical skills, so now most schools focus on developing other talents and capacities of their students. The eight intelligences include: linguistic intelligence (strong in use of language), logical-mathematical intelligence (strong in scientific thinking and problem solving), spatial intelligence (strong in visual thinking), bodily-kinesthetic intelligence (strong in body control and movement), musical intelligence (strong in music and rhythm), interpersonal intelligence (strong in communication with people), intrapersonal intelligence (strong in self-awareness), naturalist intelligence (strong in finding relationships to nature).

Felder-Silverman's Learning Styles Model

Richard Felder and Linda K. Silverman co-developed a learning style model to classify the ways of engineering learners to perceiving information [39]. This model consists of four dimensions including sensing-intuitive; visual-verbal; active-reflective, and sequential-global. The proposed dimensions are neither original nor comprehensive, such as sensing/intuitive dimension is based on Jung's theory of psychological types [63]; visual/verbal dimension is a component of VARK model [44]; active/reflective dimension is a component of a Kolb's model [74]; sequential/global dimension is according to Gregorc's mind styles model [53]. The contents of four scales are summarized as follows [40]:

- *Sensing* (concrete thinker, practical, oriented toward facts and procedures) or *intuitive* (abstract thinker, innovative, oriented toward theories and underlying meanings);
- *Visual* (prefer visual representations of presented material, such as pictures, diagrams, and flow charts) or *verbal* (prefer written and spoken explanations);
- *Active* (learn by trying things out, enjoy working in groups) or *reflective* (learn by thinking things through, prefer working alone or with one or two familiar partners);
- *Sequential* (linear thinking process, learn in incremental steps) or *global* (holistic thinking process, learn in large leaps).

These four dimensions help instructors to provide effective teaching materials to their students.

According to the above models, few learning style models have been developed and specified to the engineering students except Felder-Silverman's model. Felder-Silverman's model has been studied in a vast number of engineering students [40]. In addition, learning of computational concepts is similar to learning of engineering subjects, which also emphasizes concepts and logic. The aim of this thesis is to explore how children learn computational concepts through tangible computational construction platforms, and therefore we adopted Felder-Silverman's model in our study, as described in Chapter 6, Section 6.4.

2.3 Conceptual Frameworks for Tangible Interfaces

Tangible computational construction platforms contain tangible construction interfaces, thus it is important for us to understand the features of tangible interfaces and how tangible interfaces support learning. These may be helpful for us to define factors for exploring our proposed three kinds of tangible platforms. Researchers believe that besides tangible interfaces take the advantage of interacting with physical objects, and also the results of observations and responses on physical activity [4, 34, 86, 103]. To aid developers in designing new tangible systems, researchers have proposed some conceptual frameworks which provide explanatory power to understand the design factors of tangible interfaces and analyze the results of the tangible systems. We try to address three conceptual frameworks which focus on the current directions of the exploration of tangible interfaces for children's usage and learning. Before that, we introduce the fundamental concept of the tangible user interface.

2.3.1 Tangible User Interface (TUI)

Ishii and Ullmer proposed a new user interface called "Tangible User Interface" (TUI) that uses tangible objects to represent the digital world [67]. TUI aims to bridge the gaps between the virtual and the physical environments by manipulating the digital information directly with our hands and perceiving its physical embodiment through our peripheral senses. They also defined the term "tangible bits" to represent the tangible digital information. The metaDESK design approach is an example of physical instantiation of GUI elements such as windows, icons, and handles in TUI (see Figure 2.10 and 2.11). This framework provides a basic model of representational relationships between the digital information and the physical artefacts, but not includes the relationship between tangible interfaces and learning.

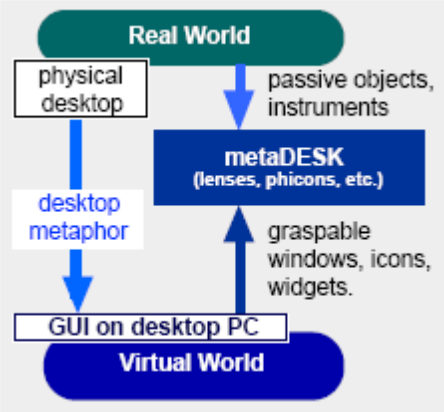


Figure 2.10. metaDESK design approach [67]

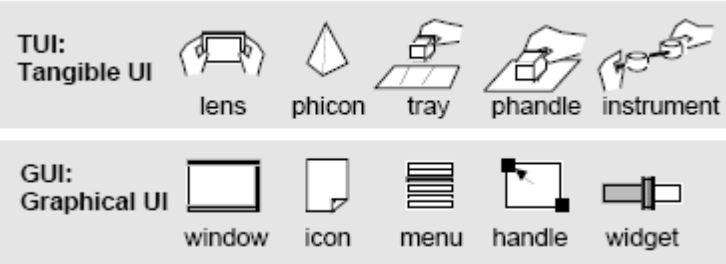


Figure 2.11. Physical instantiation of GUI elements in TUI [61]

2.3.2 Child Tangible Interaction framework (CTI)

The Child Tangible Interaction (CTI) framework [4] is a conceptual design framework that derived the design features for tangible and spatial interactive systems from the literature on child cognitive development for children under the age of twelve. The CTI framework consists of five aspects of tangible systems: 1) *spaces for action* are related to how the actions affect computation in the traditional learning environment for children, such as theme parks and museums, 2) *perceptual mappings* refer to the perceptual coupling between the physical and digital aspects of the system, 3) *behavioral mappings* focus on the relationship of the input behavior and output effects of the physical and digital aspects of the system, 4) *semantic mappings* refer to the carried information linkage of the physical and digital aspects of the system, and 5) *space for friends* means for support collaboration and imitation. This framework provides the design concepts related to spatial aspects of the systems and the various kinds of mappings between the physical and digital aspects of the systems. In tangible computational construction platforms, spatial aspects may be related to the support of flexibility of learning environment and the collaborative

construction interface which may influence children's computational learning; mappings between the physical and digital aspects may be related to the coupling of computational concepts which should be one of the core factors for children to learn computational concepts. These two aspects may be helpful in analyzing the tangible computational systems.

2.3.3 A framework for conceptualizing tangible environments

A framework for conceptualizing tangible environments [103] is proposed by Price. This conceptual framework highlights the central role of the external representations in tangible environments. It addresses four features of the representation relationships of the digital information and physical artifacts in environment impact on learning: 1) *locations* refer to the different location coupling between the digital information and the physical artifacts, 2) *dynamics* refer to the different created information associations between the digital information and the physical artifacts, 3) *correspondence* that refers to the closeness of the mapping between the physical artifacts and the learning concepts, and 4) *modality* focuses on the understanding the value of the conjunction of the interactions of visual, audio and tactile. This framework helps us to understand the design features of tangibles and representations on interaction for learning, and the underlying mechanisms of tangible environments that impact on learning. The highlighted features especially dynamics and correspondence are also important for the tangible computational construction platforms to support learning, which may help us to identify the advantages and disadvantages of the representations in tangible learning environments.

2.3.4 A framework on tangibles for learning

An analytic framework on tangibles for learning is proposed by Marshall [86]. This framework identifies the six perspectives of the latest trends and assumptions that might facilitate the design of tangible interfaces for learning. The six perspectives are: 1) *learning benefits* refer to how tangibles support more effective or more natural learning related to child cognitive development, 2) *learning domain* refers to

how tangible interface designs highlight the interesting commonality such as molecular biology and chemistry, 3) *types of learning activity* focus on the discussion on the learning possible under two types of learning activity: exploratory and expressive, 4) *integration of representations* refer to the spatial and temporal relationship between representations to support learning, 5) *concreteness and sensory directness* focus on the discussion on the impact of the potential learning benefit with interacting the concreteness of physical representation , and 6) *effects of physicality* bring out the discussions on the effectiveness of using physical materials for learning. Marshall also addresses the needs on the measurable demonstrations of the learning benefit of using physical materials. These perspectives may inspire us to consider the factors of tangible computational construction platforms which may focus on improving learning, and the measurement of learning benefits for tangible computational construction platforms which may be related to some factors such as user interfaces, domains and representations.

Overall, the above conceptual frameworks focus on the general tangible learning systems, not specific to the tangible computational platforms. Some aspects of the above conceptual frameworks may not have to be considered in tangible computational construction platforms. For example, CTI framework focuses on various kinds of mappings between the physical and digital aspects of the system, in which *behavioral mappings* may not be directly related to tangible computational platforms because learners learn computational concepts that should be explored through programming the input behaviors and the output effects of physical and digital aspects of the system. Price's framework considers the *modality* aspect which refers to understand the value of the conjunction of the interactions of visual, audio and tactile, but tangible computational construction platforms should provide a programming environment for learners to program the conjunction of the interactions of visual (e.g. LEDs), audio (e.g. buzzers) and tactile (e.g. switches). Marshall's framework considers the *integration of representations* aspect which refers to the spatial and temporal relationship between representations, but tangible construction interfaces that contain a set of physical materials should not include

temporal feature for representations. In addition, as far as we know, most conceptual frameworks focus on the relationship between the physical and digital aspects of the system rather than the relationship between the styles of learners and tangible interfaces aspects. Therefore, we indentify five factors which are drawn from the frameworks described above and traditional learning theories are specified for further analysis on the three kinds of tangible computational construction platforms to support children's learning (applied in chapter 6).

2.4 General Evaluation Approaches for Tangible Interfaces

To our best knowledge, there are no standard evaluation methods for tangible user interfaces, as it is relatively difficult to define the benchmark to compare in this novel area. To evaluate the tangible systems, there are three general evaluation methods for measuring the performance: proof-of-concept prototypes, ethnography and comparative studies. Most of these approaches are also applied in our case studies.

2.4.1 Proof-of-concept Prototypes

A proof-of-concept prototype is an initial evaluation method in tangible user interface or other novel research fields to verify the concept or theory by the demonstration of its feasibility [112]. According to the Bruce Carsten's definition [25], proof-of-concept prototype is a term that is similar to an engineering prototype, but one in which the purpose was only to demonstrate the feasibility of a new circuit and/or a fabrication technique, and was not planned to be an early version of a production design. For example, Mcnerney developed Tangible Programming Bricks to prove the concepts of tangible programming [89]. He redesigned the LEGO bricks embedded with microprocessor and adding a card slot to the side of each brick that could be used for building simple programs with parameters passing, such as counting the number of the wheel revolutions and displaying the speed and distance of the bicycle travelled. He arranged a 30-minute informal user testing with four children to complete some tasks such as using the bricks and bicycle to measure

distance. The successes of children in accomplishing tasks show that tangible programming systems are simple for children to work on.

2.4.2 Ethnography

Ethnography is a research strategy often used in the HCI area [112]. It is often employed for gathering empirical data on user experience. Data collection is often done through qualitative observations, interviews, questionnaires, and video analysis. Video analysis is well suited to investigating verbal and nonverbal behavior, and focusing on the interaction between the user and the physical interfaces of the system. The video data can be observed iteratively to remain the possibility to open new aspects and develop new analysis criteria, especially only with a loosely phrased hypothesis. For example, AlgoBlock programming language [112] was taken a video to analyze the collaboration of children body movement and positioning in playing the blocks. Qualitative observations and interviews tend to be more suitable for small groups of participants. For example, Zuckerman et al [134] interviewed the children during FlowBlocks session that involved doing a set of tasks to examine their understanding of the tasks and the systems. Questionnaires are also suitable in case studies; especially standardized questionnaires are often useful to analyze the user's feedback in a series of experimental workshops or in the wild. For example, Katterfeldt et al [70] used a set of the pre and post survey to assist in analyzing the qualitative results obtained from workshops.

2.4.3 Comparative Studies

Comparative study is a method to quantify the performance of user interfaces by comparing to other similar platforms with different variants of the interfaces, such as graphical user interface compared to tangible user interface or comparing different interaction styles [112]. The comparative studies would often measure the objective quantitative factors, such as task completion time, error rate, and memorization time. For example, Jacob's study on the performance of the Senseboard [62], he measured the performance of the interfaces under four different conditions by comparing the correctness rate and the time completion of the tasks. In the recent years, the

comparative studies focus on the high-level interaction qualities more, such as enjoyment, engagement and legibility of actions. For example, Horn's study on the comparison of the effectiveness of a tangible and a graphical programming interface [58]. He defined six qualities to measure the effectiveness: inviting, apprehendable, active collaborative, engaging, programs and child-focused. There are still rather rare examples of comparative studies among the tangible user interface systems, particular in the domain of learning computational concepts.

2.5 Programming Paradigms

Programming paradigm is a pattern of problem solving styles that underlies a particular type of programs and languages. To understand the characteristics of each kind of paradigms, it is important for us to design a workshop syllabus which is suitable for children to learn, and to determine how the programming environments to be designed or used in the case studies. There are five fundamental programming paradigms introduced below.

2.5.1 Imperative Programming

Imperative programming describes computation in terms of statements or commands, and each execution of each statement changes a program state [102]. In this model, both the program and its variables are stored together, and the program contains a series of commands that perform calculations, assign values to variables, retrieve input and produce output, etc. Procedural abstraction is an essential building block for imperative programming as sequences, conditions and loops [123]. The examples of imperative programming languages include C, Basic and Pascal, etc.

2.5.2 Object-Oriented Programming

Object-Oriented programming uses objects interact with each other by passing messages that transform their states [123]. Message passing means that the data objects are allowed to become active rather than passive. There are some fundamental techniques included, classification, inheritance and message passing. Examples of object-oriented programming languages include C++, Java and C#, etc.

2.5.3 Functional Programming

Functional programming models a computational problem as a collection of mathematical functions, each with an input (domain) and a result (range) space [123]. Functions interact and combine with each other using functional composition, conditional, and recursion. Examples of functional programming languages include Lisp, Scheme and Haskell, etc.

2.5.4 Logic Programming

Logic programming, also named declarative programming, allows a program to model a problem by declaring the outcome instead of the algorithm [123]. It also provides a natural vehicle for expressing non-determinism, which is appropriate for problems whose specifications are incomplete. An example of a logic programming language is Prolog.

2.5.5 End-user programming

End-user programming cannot be said a programming paradigm. Instead, it is a method or technique to present programming such that most users will not require as much time to learn the tools and skills of a professional programmer [77]. There is a variety of techniques to make it easier for the user to write a program, such as programming by demonstration [30] and graphical programming [68]. As children are mostly programming novices, many child-oriented programming frameworks are designed as end-user programming applications.

Chapter 3 Abstract Thinking through Functional Blocks Construction

This chapter examines the impact of tangible programming system on children's learning of computational concepts. This study investigates the design of tangible languages and construction interfaces. Two paper prototypes were designed with different expression methods to model the computational concepts associated with story elements or robot actions. Based on the observation of the children's interactions with these two paper prototypes, the i*CATch wearable toolkit was modified to simulate a tangible programming system and was called "i*CATchBadges". This system was used to investigate how children interact with the badges in two different learning environments: school fun fair game booth and the classroom. During the task period, children were encouraged to associate the construction badges with some meanings such as the functionality of intelligent clothing and the representations of the functions of the badges. The following presents a detailed discussion on the effectiveness of tangible programming systems in helping children learn computational concepts through three perspectives:

- **Computational Concepts -** analyze how children learn computational concepts through tangible programming systems
- **Task Outcomes -** discuss the possible aspects of tasks solved by tangible programming systems
- **Learning Environments -** explore the potential usage of tangible programming systems in different learning environments

3.1 Tangible Programming System – a kind of Conceptual Manipulation

Tangible programming system supports a tangible programming language, which is programmed by grasping a set of tangible blocks instead of writing texts or manipulating virtual objects displayed on a computer screen. It consists of a set of tangible blocks, and each of which represents a function or an expression, which

focuses on modeling conceptual structures instead of constructing physical objects. This means that the learning approach that is different from some well-known computational construction platforms like LEGO Mindstorms [90]. This tangible learning approach supports hands-on learning that is a natural way for children to learn, and also encourages learning of abstract concepts such as computational concepts.

There are two kinds of tangible programming systems. One kind of tangible programming systems contains two parts: the input is through a tangible user interface (TUI) such as a set of physical blocks, and the output is displayed on a separated graphical user interface (GUI) such as a computer screen or a physical object such as a robot. For example, AlgoBlocks [120] has a collection of physical computational building blocks that controls a virtual submarine on the computer screen; Tern [56] has a set of wooden blocks shaped like jigsaw puzzle pieces that created a program by a chain of wooden blocks to control the actions of a robot. Another kind of tangible programming system contains only one part, which serves as both input and output, with no standard computers, but embedded microcontrollers involved. For example, Tangible Programming Bricks [89] is made of computational LEGO bricks to explore measurement and computation such as bicycle's velocity computation; SystemBlocks [135] and FlowBlocks [134] are similar and both consist of a set of computational blocks which simulates computational concepts through dynamic behaviours or processes.

3.2 Design Process

This section presents our two early paper prototypes: LivePic Study and COATline Study. The preliminary design dimensions of these two prototypes are listed, based on which the final prototypes: i*CATchBadges is designed.

3.2.1 Two Early Prototypes

We developed two early paper prototypes LivePic and COATline to review the user interface of the tangible programming environment. A tangible user interface (TUI) was simulated as an input device to control an output device such as a monitor or a

physical robot. The objective of such simulation is to evaluate the pros and cons of these two initial designs. A story structure was used as the theme to simulate the computational concepts and behavior, as we believe that storytelling is a natural way for children to learn and express ideas. The design dimensions of two prototypes were basically the same, except the expression method of programming statements. In these two preliminary studies, we learned from several flaws on the user interface designs, which helped us to improve the final technical and physical design of a tangible programming system. The details of initial design dimensions and the studies of two paper prototypes are described as follows.

Initial Design Dimensions

There are three core initial design dimensions considered in our design process.

(1) Knowledge of Computational Concepts

There are some children's computational toolkits used the event-based or object-based paradigms [72] that are convenient for programming, but they are not necessarily appropriate in a toolkit that is meant to support the teaching of programming. This toolkit should contain basic computation concepts, including sequences, conditions and repetitions.

(2) Gender-Neutral Programming Domain

Many child-centred programming environments focus on robot control and mathematical concepts. However, these kinds of learning domains tend to be appealing to boys. Therefore, storytelling was chosen to be the programming domain, as it is familiar to children and is a fairly gender-neutral activity at least for younger children. In addition, some story structures actually are similar to a program structure. For example, stories are usually written in chronological order, echoing the sequentiality of programs; interactive stories consist of choices, similar to the concept of conditions in programming; in fairy tales, it is common to repeat events three times, each time with slightly different elements, which is similar to the concept of repetition. Storytelling should be able to broaden the population of children who may be interested in learning programming, and the characteristics of

story structure should help children to understand the abstract programming concepts more easily.

(3) Expressions of Programming Statements

In LivePic prototype, the program specified textual statements by moving to a scene-based expression, in the style of a pictorial story book. Each programming statement acts as a scene in the story book. Our rationale for using this is to facilitate learners to understand the program flow, because of the similarity of reading a pictorial storybook. In COATline prototype, a flowchart-based notation was used to make the design of the system more intuitive and simple. It was simplified in terms of the rules of the scene-based expression and formed a flowchart-like sequence flow. Our reason for using a flowchart-based notation is to assist learners in focusing on each step without being overwhelmed by the whole process.

LivePic Study

The initial paper prototype was called LivePic, with a meaning of live pictograph, in which animated pictures demonstrated their actions or meanings within the context of a story or a message. Our approach was similar to a graphic-based programming system, but the tangible user interface was applied to make it more intuitive and reduce the abstractions. The LivePic system was therefore a combination of these two elements -- a kind of “tangible pictograph” – which was used to represent scenes (or programming statements in terms of computer language). Originally, a set of rules were designed to express story (or program) statements in a way that would be free of ambiguity. Two examples of the rules: 1) four types of badges were defined to be used in the particular positions of a patch to represent a programming statement: *characters*, *actions*, *objects and descriptions*, which represent subjects, verbs, objects, and adjectives or adverbs in English grammar (Figure 3.1); 2) three categories of patches were designed to echo the three basic constructs of programming: sequences, conditions and iterations. However, these rules were not so successful, because the rules were too complex for children to digest and apply into a story.

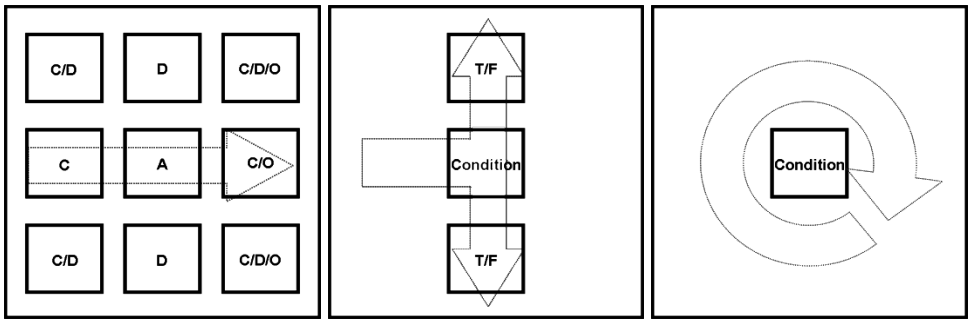
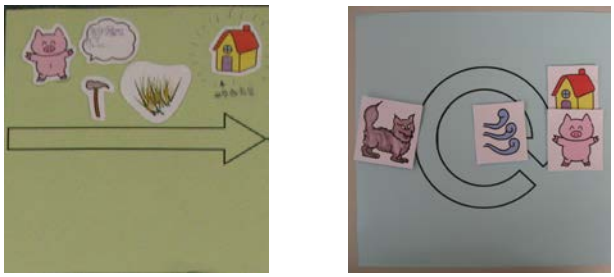


Figure 3.1. The grid distribution for the three types of patches: (a) sequence, (b) condition and (c) iteration, where C means Characters, D mean Descriptions, A means Actions, O means Objects, T means True and F mean False.

Task: Recreate the story of the Three Little Pigs

Ten children (six boys and four girls) from 8 to 12 years old studying in local public primary schools participated in the study. They were separated by age and gender into three groups of three to four students each: one was all-male, one was all-female and one was mixed. The children were provided a set of LivePic materials (badges and patches) to recreate the *Three Little Pigs* story in a playground. We gave a brief introduction of the rules of LivePic, but we did not expect the children to remember all the rules and gave them freedom to place the badges on the patches as they felt appropriate. The objectives were 1) to investigate whether LivePic can assist them to analyze or organize a story, and 2) to observe how easily children could use LivePic to express their ideas.



(a) (b)

Figure 3.2. (a) A sequence patch represents a statement “A pig cuts grass to build a house” with badges [pig], [hummer,] [grass] and [house], and adding a [dialog cloud] to add more interest to the scene. (b) A simplified version of the *Three Little Pigs* using an iteration patch

Observations

In general, the children encountered two main difficulties when doing the task: 1) expressing a concise, complete statement using the given badges, and 2) developing the story flow with respect to program constructs. For example, the children often gave too much detail in their expressions. In one scene of the story created by a mixed group, they illustrated a statement with "the first pig finds a salesman from whom he buys the straw to build his house" (Figure 3.2 (a)), rather than using a simpler manner (involves fewer characters) such as "the first pig buys straw". This phenomenon is similar to novices who write a simple program with redundancies, making it unwieldy and clumsy. This observed behaviour informs us about the need to reconsider the user interface design and its affordances, so as to enable the user to more easily follow the programming/storytelling rules. On the other hand, the children tended to focus on the sequence of story, and did not try to use conditions or iterations to summarize the story flow. The exception was one all-girl group, which tried to use the iteration patch to make a simplified version of the *Three Little Pigs* (Figure 3.2 (b)). In addition, some objects were designed to have more than one part of speech in LivePic, such as the "fire" badge, which can be used to represent its actual meaning: a fire (which would be a noun), and being angry (which would be an adjective). However, it seems that children tend to focus more on actions, rather than descriptions.

Children's Feedback

Interviews were used to obtain feedback directly from the children rather than filling in a survey form, due to the limited time and the age of the children. The questions mainly focused on the medium for storytelling - drawing versus LivePic. Most of them preferred using LivePic to drawing, as they did not think they can draw well. On the other hand, they enjoyed the idea of using tangible objects to build something. This gives more evidence to the hypothesis that tangible interfaces are more appealing to children.

Discussions

The observation of the study revealed two weaknesses of LivePic: 1) the rules are not intuitive to children, and 2) the ambiguity of the meaning of patches may occur.

Most of them put badges on the wrong position. For example, the children always placed all badges on the top half of a patch (see Figure 3.2 (a)), but the action badge should be placed in the centre. Each badge basically is an object, but it has different parts of speech when it is placed on different positions of a patch. Some positions share three types of parts of speech, and this rule causes the ambiguity of the meaning of a patch. For example, one all-girl group placed the [house] on the top of the [pig] (see Figure 3.2 (b)), in which the position of the [house] can be a character, a description or an object. This patch can be interpreted as "the wolf blows the house and the pig" or "the wolf blows the pig as a house". As we know that the children retold the *Three Little Pigs* story, we do not choose the latter interpretation. However, it is difficult to determine the real meaning in other new stories. The original rationale of this design is to be more flexible to express stories, but this flexibility brings ambiguity to understand.

Despite some problems with LivePic, there are still two strengths: 1) it allows children to have a figure classification practice, and 2) it encourages collaboration among children. One interesting discovery was that most of the children tried to classify the badges before creating the story. That way, they could know which badges were related, and which were not, hence indirectly practicing a form of classification exercise. Also, LivePic supported children to discuss and construct the story in a collaborative fashion. Each student had his/her chance to put badges onto patches to make different parts of the story. Furthermore, none of them wanted to stop for a break when given the opportunity, preferring to finish the story.

COATline Study

In order to address the weaknesses of LivePic as evidenced by the evaluation, modifications to the design was made, which resulted in: COATline. It follows the idea of LivePic in supporting basic programming concepts, including sequences, conditionals and iterations. It was simplified from four to three terms of badges: characters, objects and actions, and supported a simple sentence structure: *subject (character) verb (action) object (object).* It also focused on the interface design on the relationship between character, object and action, using the programming

flowchart as an inspiration to visualize the timeline flowing from one event to another (Figure 3.3). The characteristics of character and object are similar to variables or parameters in terms of programming constructs, whereas the characteristics of actions are similar to sequences, conditions and iterations, and therefore there are three types of actions (Figure 3.4).

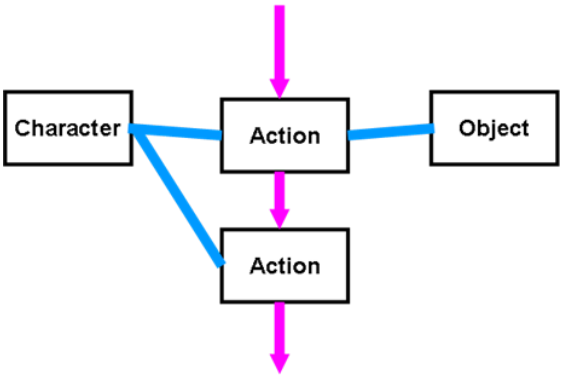


Figure 3.3. Overview of COATline design: link up a character or an object with an action by a blue ribbon without arrow; link up actions by a pink ribbon with arrow

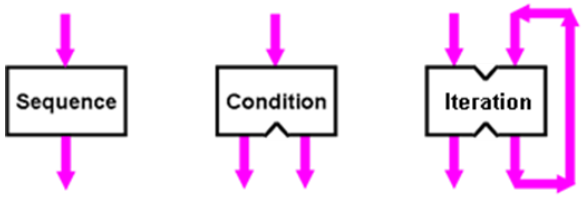


Figure 3.4. Three types of actions: sequence, condition and iteration

Researchers found that a common novice programming problem is variable usage [23]. It is easy to forget which variables have been declared, and which have not. This mistake causes frustration and confusion, and is something inherent in all programming languages. Even graphical programming languages may not completely relieve the variable usage problem. Another common problem is that it is often difficult to trace a program step by step, especially when it comes to repetitions [13]. The design of COATline specifically addressed these two problems. To deal with the problem of variable usage, each character or object was designed to connect to one or more actions (see Figure 3.3). In this way, it is easy to figure out which action(s) are related to which character(s) and object(s). To help tracing a program flow, we use connectors with arrows to link up the actions one by one. This clue helps to avoid step-skipping by mistake. This design emphasized the

relationship between character, object and action to assist students in training up their visual and analytic abilities.

On the visual screen, this design of the programming expression resembles a graph, with a number of edges everywhere. The program seems more messy than intuitive. However, on a tangible interface, the user can physically touch the connecting ribbons and follow them from badge to badge. This integrates the sense of touch into the language, which helps to visualize the program flow (see Figure 3.5).

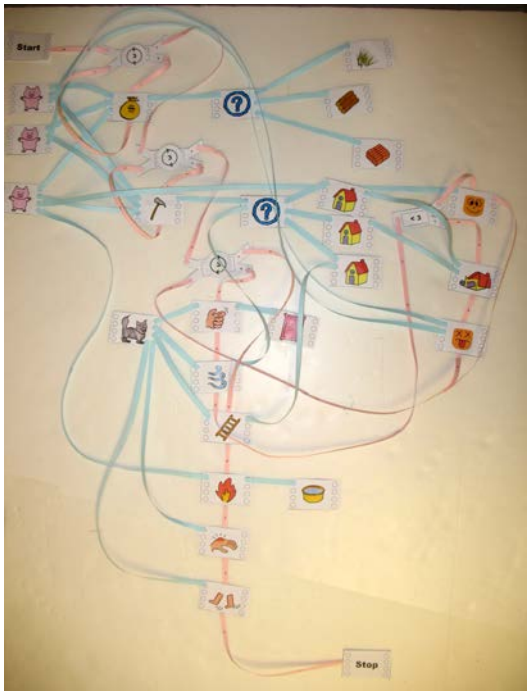


Figure 3.5. The Three Little Pigs story in COATline version

Methods

COATline was evaluated on two dimensions: 1) the readability of the notation, and 2) the usability of the system with respect to writing a story or constructing a program. Five subjects were invited to test the readability, to investigate whether it better assists the learner in reading a story, and to determine how difficult it is to use COATline to express a story. Two people were invited to test the usability, to observe how difficult it is for the learners to construct a COATline program to control a robot. Learning from the experience of the LivePic study, the evaluation

was carried out with one subject at a time such that we could better direct and observe the participants.

Readability Task: Reading the Three Little Pigs

Five subjects were invited: Two female subjects were university staff with computer science background; one male and one female subject were first-year computer science students with weak programming knowledge; the last male subject was a writer of children's stories. Although none of these individuals are the representatives of our target users, the purpose of this evaluation is to obtain users' reactions towards the non-traditional and non-conventional kind of representation of COATline. For this evaluation, the participants first had a brief introduction to the COATline version of the *Three Little Pigs story* (Figure 3.5), and then were requested to explain the meaning of the representations. The objective was to gauge the understandability and intuitiveness of the new COATline design. The motivation for this evaluation is that a tool needs to be easy to understand and read if it is to be effective at learning.

Observations

When the subjects were presented with COATline, the first impression of the one female staff and one female student was that it was very confusing, as it had many criss-crossing lines, while the rest of the subjects had no particular feeling one way or another. Basically, all of them could follow the event flow of the story without any problem; the only difficulty was that the participants sometimes mixed up the pink lines (which are used for connecting actions to each other in the form of a timeline/action sequence) with the blue ones (which connect characters and objects to actions that they perform). The only exception was the loop structure, where all they needed to be shown first how to read the loop. That may be because a loop is less natural when compared to sequential action.

Programming Tasks: Controlling electronic components

For this evaluation, two subjects were asked to do several tasks using the COATline version to construct programs for electronic device control. One subject was a boy, aged 16, who was studying in arts and commercial subjects and had never learned any programming before. Another subject was a boy, aged 5 ½, who was studying in

kindergarten and had a little experience with Lego robotics. The objective was to gauge the difficulty of using COATline, which is a story-flowchart-based representation of a program problem. We believe if the user finds it easy to pick up the rules of the tool, he/she can spend more time focusing on learning logic concepts and problem solving.

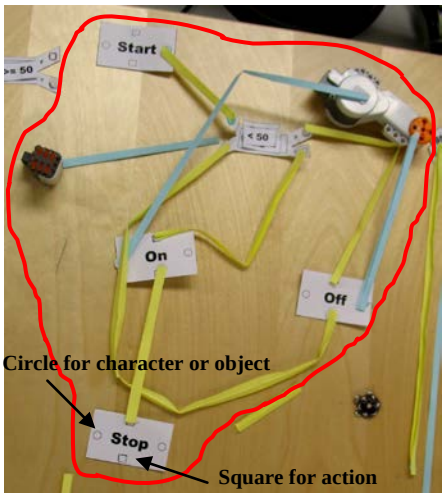
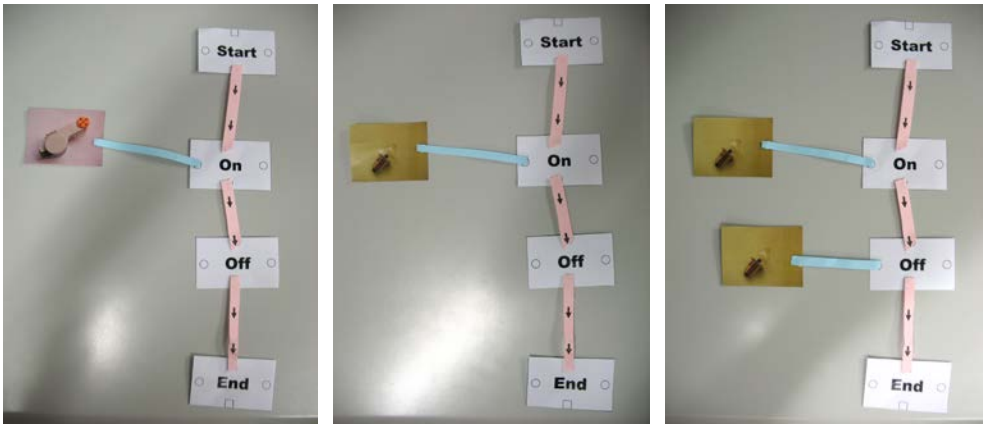


Figure 3.6. The results of the third task done by the 16-year-old boy: after additional clarifications and the addition of an affordance onto the badges, the boy performed the task correctly. Yellow strip is used for the connection between two actions and blue strip is used for the connection between character, action and object.

Results

The 16-year-old boy was asked to do three similar tasks with different types of actuators and sensors: turn on or off actuators (light or motor) by a sensor (light or sound sensor). During the first task, he could not distinguish between the action and the object (e.g. [light sensor] is an object, not an action, but it was connected to [start] action and [check sensor value condition] action with blue strips), and between the data flow and the action flow (e.g. in the [condition] action badge with the true and false connection points, one blue strip was connected to true and one yellow strip was connected to false). After being given the answer of the first task, we let him try another similar task again. He still had a similar problem with the connections of badges. He also was not able to generalize and identify the common areas between the first and the second task (both involve controlling an actuator using a light sensor in which one is a light and another one is a motor). Before he started to do the third task, we suggested him thinking about the flow first and then the relationship

between the object and action. We also drew two squares and two circles onto the action badges, and told him that the action strips should be attached to the squares, and the data strips to the circles. With this additional clarification and help, he was able to finish the third task (see Figure 3.6). This shows that the affordance of the physical interface is an important factor towards reducing errors and enhancing the efficiency of learning.



(a) (b) (c)

Figure 3.7. (a) The example task for teaching the 5 1/2 year-old boy: Control a motor on and off. However, there is a mistake on no connection between [motor] and [off] badges. (b) The task done by the boy: Control a light on and off (c) The corrected version of the task done by the boy: add one [light] and connect it to [off]

The 5½-year-old boy was asked to do two tasks: one is to control a light bulb on or off; another one is to explain the movements of the car. Before he started on the first task, he was taught how to control a motor on and off by demonstrating how to connect the badges (Figure 3.7 (a)). Then, he was asked to use COATline to turn a light bulb on and off. He had no problem finishing the task (Figure 3.7 (b)), but was very easily distracted and needed to be reminded and encouraged to finish. As a result of the distraction, one interesting observation was made: a mistake was made on our example of controlling a motor on and off by forgetting to make a connection between the [motor] and [off] badges. This, however, turned to be an opportunity to ask the boy if anything was missing on the action [off]. The response was to pick up another [light bulb] icon and connect it to the [off] badge, as he thought any picture of a light would refer to the same light bulb. The result was unexpected, as we had not previously explained that one object could have two actions, but the boy was

able to extrapolate from similar data to correct our mistake (Figure 3.7 (c)). In the second task, we asked him to trace and explain a program that would cause a car to move when the environment is bright, and to stop when it was dark. At the very beginning, we tried to connect a [wheel] to action [on], and so on. However, this caused some difficulties in understanding, as the boy was not able to associate a wheel turning on with a car moving. We then changed the wordings into more general terms, such as using [move] instead of [on], [stop] instead of [off], and connecting to a [car] instead of a [wheel]. With these changes, he could follow the flow and explain how the car moves (Figure 3.8).

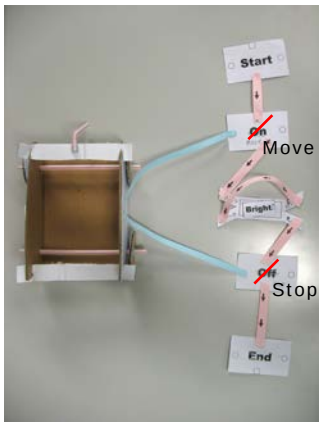


Figure 3.8. Using COATline expression method to control the car moves while it senses the environment is bright or not. To make it easier for the boy to understand, the action ON was changed into MOVE, and OFF was changed into STOP.

Lessons Learned from Two Paper Prototypes

In these two studies, we used two different story expression approaches to represent a programming problem. We found that some design elements or improvements would be useful for the next iteration prototype:

- **Simple Output** - From our observations, the children had no problem imagining the output of the tangible programs as a story animation displayed on the screen or a real robot movement either in LivePic or COATline. This inspired us to design the interactive prototype with a tangible user interface that serve as both inputs and outputs, thus the simple and concrete functional output provides more space for children to practice their imaginations, which facilitates learning of abstract computational concepts.

- **Affordance** – The LivePic interface does not provide any visual clues for children to put the badges onto the patches, while in the COATline interface, the additional squares and circles helped the 16-year-old boy to finish the task. Therefore, effective use of affordance on the physical interface is important in reducing errors and enhancing the efficiency of learning.
- **Syntax Errors Prevention** – From our observations, the story-flowchart-based structure (character, action and object) used in COATline is more intuitive for children to express their ideas and construct their programs, but the children sometimes introduced syntax errors especially in condition and iteration action badges. Since these two kinds of action badges contain branches that connect to more than one action badge, it is difficult to direct a connection strip to attach to the correct place. To prevent this problem, the construction interface has to be redesigned; otherwise a complier is needed to check the syntax errors.
- **Plot/Action Driven** – According to our observation, the children preferred the action description to the character/object description. In addition, the writer, one of our subjects, stated that action events were appealing to children, so a child's story should be plot-driven instead of character-driven. His professional comment supported our decision to base our design on an event timeline.

3.2.2 The Evolution of the design into i*CATchBadges

To transform the COATline paper prototype into the i*CATchBadges interactive prototype, we revised the design dimensions based on the two early paper prototypes and modified a suitable current platform – i*CATch [95] – a plug and play wearable construction kit (see Figure 3.9) into "i*CATchBadges" for our study.

Figure 3.9. The i*CATch wearable construction kit

The revised design dimensions are shown below:

(1) Knowledge of Computational Concepts

The computation concepts sequences, conditions and repetitions were basically kept in the interactive prototype. However, due to providing a “syntax” error free construction interface, the design of the construction interface was simplified to eliminate the connection strips between the badges. To accommodate the simplified construction interface, the conditional and repetitive constructs would be more constrained than the paper prototype version. In addition, as children prefer to work with the action events, the character and object elements were discarded, keeping only the action element. Therefore, the programming paradigm of the interactive prototype is functional rather than imperative.

(2) Increment of Abstraction

In the paper prototype, each badge had one concrete meaning (e.g. run) and was designed with certain outputs. It is not scalable to create numerous physical badges each corresponding to a word in the dictionary for storytelling. To solve this problem, the meaning of each badge should be more abstract, thus allowing it to be mapped to more than one concrete meaning to increase the flexibility. For example, a LED device flashes (more abstract) could be used to represent “pig runs” or “wolf escapes” (more concrete). Thus, the abstract tangible programming language would support a variety of domains such as storytelling, robot movement controls, and even intelligent clothing functions.

(3) Space of Imagination

According to our studies, the children are able to imagine the output results of their tangible programs (demonstrated by paper badges) as a scene or an action. Therefore, the external output does not have to be a screen display, but can be some simple effects of actuators or sensors such as flashing light patterns and melodies, and each badge would be a kind of program functions. These concrete physical functions (but abstract to children's story or life) in a general programming flow provide space for children to imagine concrete things and map them to familiar meanings for themselves.

(4) Function-based Expression

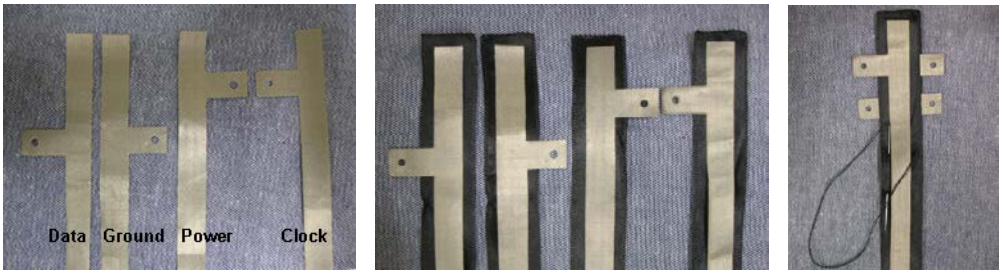
To simplify the tangible construction interface, the LivePic scene-based expression and COATline flowchart-based notation were discarded, and moved to a function-based notation. The interface would support both input (program construction) and output (program result). Our rationale was to provide a syntax free interface for programming and maintain a certain level of abstraction for children's imagination, as we believe that this direct input and output function-based expression is intuitive for children and their imaginations on the program output would assist them to assimilate the computational concepts.

3.3 Implementation

3.3.1 Technical Setup

i*CATchBadges were constructed on the i*CATch wearable construction toolkit. The i*CATch main board and peripheral modules (or badges in this setting) were designed on the Arduino platform, and used an ATMega 168 chip as the microcontroller, with pins 27 and 28 for inter-integrated circuit (I2C) communications. To support the plug-and-play functionality of the i*CATchBadges, I2C bus technology was applied as the communications channel between the electronic devices. The I2C protocol requires two lines (signal and clock) for data communication and most modules also need to be connected to the power supply and the ground wires. On the I2C bus, the main board acts as the master device and

is responsible for sending instructions with the peripheral input and output badges. All custom instructions of each i*CATchBadges were written into a program developed in the Arduino integrated development environment (IDE).



(a) (b) (c)

Figure 3.10. Constructing the communications bus (a) The individual bus lines, with tabs for the snap buttons (b) Bus lines adhered to insulating nylon (c) Combining the individual lines to make the communication bus

To create the communication bus, the four strips of conductive fabric were precut to the appropriate length, with tabs for affixing snap buttons at the correct locations (Figure 3.10 (a)). Each strip is then adhered onto a length of close-weave insulating nylon (Figure 3.10 (b)). The strips are then laid over each other with the data line at the bottom, then the ground line, and then the power, and finally the clock line on top. This configuration allows necessary separation between the data clock lines to minimize potential capacitance and crosstalk issues. [64]. The edges of the insulating material are then tacked together to create the bus (Figure 3.10 (c)). The communication channel is created with 3 ohms of resistance per meter, which is well within the range of tolerance required by I2C for signal transmission.

To create the final construction platform, the communications bus is first stitched onto a wearable garment or accessory (Figure 3.11 (a)). The snap button connectors are then affixed to the extension tabs using a button gun (Figure 3.11 (b)). To make it less likely that the user would misuse the interface, two lines use male snap buttons, while the other lines use female snaps (Figure 3.11 (c)). This prevents the user from plugging in a device backwards and causing damage. Finally, a layer of lining material is affixed over the communications bus and the exposed extension tabs, both to prevent short-circuiting from the inside of the garment as well as for comfort (Figure 3.11 (d)).

(a) (b) (c) (d)

Figure 3.11. Making the construction platform (a) Affixing the bus to the garment substrate (b) Fixing the interface snap buttons (c) The standardized i*CATch interface socket (d) Insulating the inside of the garment

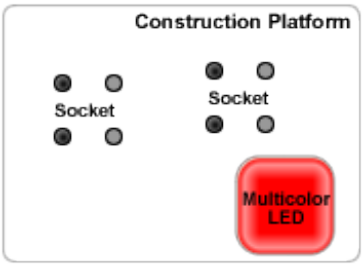
3.3.2 Programming Language

The badges are plugged on the garment construction platform one by one, and the output of the badges follows the sequence of when they are plugged onto the garment. This plugging order illustrates the concept of sequentiality (see Figure 3.12). For example, a multicolored LED and buzzer badges are plugged onto the construction platform in order, which may refer to the predefined program statements LED_ON(RED, 0.8); LED_OFF(RED, 0.3); SOUND(C, 0.5), that means that multicolored LED turns on red color for 0.8 seconds, and then turns off for 0.3 seconds, finally buzzer plays a C tone for 0.5 second. The repeated patterns are likely to reveal an iteration concept; on the other hand, the output sequence of the badges keeps going until runs out of battery, this is another way to reveal an iteration concept by an infinite loop structure. For example of a repeated pattern of a badge, a buzzer badge plays a D tone for 0.5 seconds with five times, which may refer to the predefined program statements REPEAT 5 { SOUND(D, 0.5) }. Sensor badges cannot function independently and require working with the same color labeled actuators; this combination is likely to represent a conditional construct. For example, an ultrasonic sensor and a vibration motor badges are plugged onto the garment construction platform, which may refer to a predefined program statement IF (ULTRASONIC-SENSOR-VALUE < 20) THEN VIBRATION_ON(1) ELSE VIBRATION_OFF(1), that means if the ultrasonic sensor detects an obstacle closer than 20cm, vibration motor turns on for a second; if not, it turns off. This design focuses on the flow of a program and aims to give a very simple idea of the basic computational concepts for learners within a very short period of time.

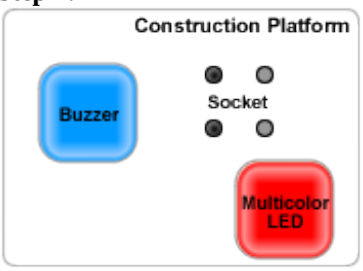
Sequences

In i*CATchBadges:

Step 1:



Step 2:



In the text program:

```
LED_ON(RED, 0.8)
LED_OFF(RED, 0.3)
SOUND(C, 0.5)
```

Interpretation:

Assume 1) the (red) multicolored LED badge is predefined to turn on red light for 0.8 second and then turn off red light for 0.3 second; 2) the (light blue) buzzer badge is predefined to play a C tone for 0.5 second.

The (red) multicolored LED badge is first plugged onto the construction platform, which is equal to the first two lines of the text program, and then the (light blue) buzzer badge is plugged on it, which is equal to the third line of the text program. The position of the badge does not affect the sequence of output.

Iterations

In i*CATchBadges:

In the text program:

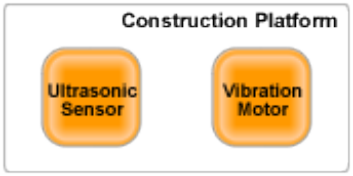
```
REPEAT 5 { SOUND(D, 0.5) }
```

Interpretation:

Assume the (pink) buzzer badge is predefined to repeat to play a D tone for 0.5 second for 5 times. The (pink) buzzer badge plugged onto the platform that is equal to the program statement in the text program.

Conditions

In i*CATchBadges:



In the text program:

```
IF (ULTRASONIC-SENSOR-VALUE < 100) THEN
      VIBRATION_ON(1)
ELSE
      VIBRATION_OFF(1)
```

Interpretation:

Assume the (orange) ultrasonic sensor is predefined to detect an obstacle less than 20cm, and then the (orange) vibration motor is triggered to turn on for 1 second; otherwise it turns off for 1 second.

To express a conditional statement, the sensor and actuator badges with the same color are plugged onto the same construction platform, regardless of the plugging order of them. The (orange) ultrasonic sensor and (orange) vibration motor badges are plugged onto the platform, which is equal to the set of program statements in the text program.

Figure 3.12. Illustration of three basic programming constructs (sequences, iterations and conditions) in i*CATchBadges programming language

The design of the programming language follows the design dimensions mentioned in section 3.3.1 to be simple with basic computational constructs: sequences, and simplified conditions and repetitions, supporting the level of abstraction and space of imagination, and function-based expression. The language syntax is expressed by a set of the i*CATch badges and each badge represents a particular function, thus it does not let users produce a syntax error. Table 3.1 summarizes the details of the language syntax of i*CATchBadges and the corresponding functions and the sample patterns. The functions and patterns could be changed in the custom program to fit the requirements of the study, and the custom program was set in the main board. Overall, there were eight types of badges: four types of which were actuators included multicolored LEDs, white LEDs, buzzers and vibration motors; and four types of which were sensors included light sensors, ultrasonic sensors, infrared sensors and switches. Each type of badges could be more than one with unique identifier address (each badge was labeled with different color sticker for children to recognize them easily.) Theoretically, up to 128 different badges can be connected onto the platform. In addition, this language is also designed to use felt icons to add meaning to the sound, light and vibration effects to personalize the construct (see Figure 3.13 and Figure 3.14).

Figure 3.13. Some samples of felt icons were provided for children to design their smart clothes or stories.

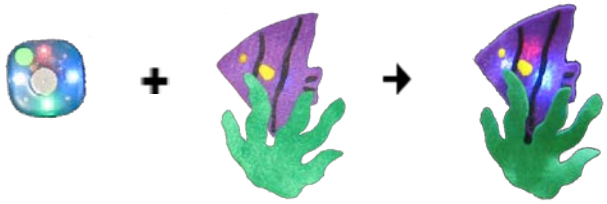


Figure 3.14. An example of using felt icons to add meaning to a multicolored LED badge: a purple fish wakes up when multicolored LEDs light up.

Badge Types	Physical Types	Functions	Sample Patterns
Multicolored LED	Actuator	Flash in a multicolored light pattern	Turn on a red LED for 0.8 seconds and turn it off for 0.3 seconds
White LED	Actuator	Flash in a white light pattern	Repeat 2 times to turn on for 0.5 seconds and turn off for 0.5 seconds
Buzzer	Actuator	Play a melody	Play a C note 4 times in 2 seconds
Vibration Motor	Actuator	Produce a vibration pattern	Repeat three times to turn on a vibration motor for 0.3 seconds and turn it off for 0.3 seconds
Light Sensor	Sensor	Turn on or off LEDs in different level of brightness	Turn off a white LED in the dark
Ultrasonic Sensor	Sensor	Change the color of lights and the speed of vibration in different level of distance	Turn on four colors of the lights and the vibration motor when in the closest distance detection
Infrared (IR) Sensor	Sensor	Produce a tone on a buzzer when receive a IR signal	Make a buzzer to produce a E note for a second when receive a IR signal from a remote control
Switch	Sensor	Change a light pattern	Turn on four colors of the lights

Table 3.1. A list of the i*CATch modules used for the i*CATchBadges programming language. There are two physical types: actuator and sensor. Each badge is mapped into a specified function.

3.4 Evaluation

Our evaluation of the i*CATchBadges took place in two different settings with primary school students aged from 6 to 12. Through these two studies of two different settings, we examined whether children could obtain some ideas of the computational concepts and what they did with the i*CATchBadges system. Furthermore, we investigated how children performed their works on the system in two different learning environments – game booth and classroom. A video camera and audio equipment were used to record children's interactions with the badges. To support the studies, the i*CATchBadges programming language used at least 20 to 30 badges (the ratio of actuators to sensors is around 2 to 1), to produce at least 30 to 40 patterns, where around 15 patterns were produced by the combination of two to three badges.

Figure 3.15. A girl decorates the jacket with i*CATchBadges in the game booth.

3.4.1 i*CATchBadges Study in Game Booth

This study was carried out as a game booth activity held in a primary school fun fair. The theme of the booth was *plug and play e-fashion*. The objective of the task was to use the tangible programming interface (i*CATchBadges and some felt icons) to decorate a jacket (see Figure 3.13). There was an animation display to demonstrate the effect of each badge or combination. While students were plugging the badges or sticking the felt icons on the jacket, they were encouraged to explore the effects of the badges and associate the function properties of the badges to the felt icons or functionality of the smart clothing. There were 16 males and 12 females, resulting in 28 persons in total. Among the subjects these, four pairs of boys, a trio of girls, and three mixed groups worked together while five male and six female students worked alone, resulting in 11 individuals and 8 groups in total. One mixed group consisted of a female teacher with her male student; another mixed group was a mother with her son; and the last mixed group was two students. All students were aged around 12 years old, except one who was 6 years old. All of them had no previous experience with programming systems.

In the decoration task, each student or group put around 3 badges on a jacket. The boys used around 2.9 badges; the girls used around 3 badges; and the mixed group used 3.7 badges (see Table 3.2). Basically, there is no significant difference between the genders while the number of badges used by the mixed group is above average. This is because a 6-year-old boy liked to plug all the badges onto the jacket, no matter what the effects of the badges were. The number of patterns produced by two badges or above and the number of felt icons on T-shirt in boys group are more than those numbers of girls group (1.0 > 0.3 patterns and 3.4 > 1.4 felt icons) while

the number of felt icons associated with badges in boys group is less than that of the girls group (1.1 < 2.4 felt icons). These numbers indicate that most of the boys spent more time combining the badges to explore different effects of the sensors and actuators, and thus they put the felt icons on the T-shirt directly instead of putting the felt icons and the badges together. Two boys working together mentioned that the function of the ultrasonic sensor could be built as a kind of anti-stalker or anti-rape devices. In contrast, the girls mainly liked to observe the effects of the lights and sounds and think how to make it pretty, and thus they spent more time on coupling the effects with the felt icons such as a tropical fish felt icon with a light and a pig felt icon with a sound (see Figure 3.14 and Appendix C). The statistical results on duration also indicate that the girls spent more time to decorate the jacket than the boys (5:10 > 4:44 minutes). Each student or group spent at least 1 minute, at most 7.25 minutes, and around 4.5 minutes in average.

	Boys	**Girls**	**Mixed**	**Average**
Number of individuals or groups	9	7	3	N/A
Number of used badges	2.9	3.0	3.7	3.1
Number of patterns (produced by two badges or above)	1.0	0.3	0.3	0.6
Number of associations (felt icons associated with the badges)	1.1	2.4	1.3	1.6
Number of felt icons on T-shirt only	3.4	1.4	6.0	3.1
Duration (minutes)	4:44	5:10	2:30	4:32

Table 3.2. Summary of the results of the students interaction with the i*CATchBadges to decorate a jacket

Figure 3.16. A representative example of a boy's jacket (left) and a girl's jacket (right)

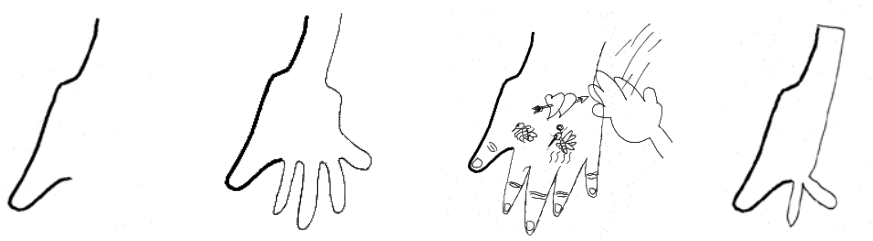
3.4.2 i*CATchBadges Study in Technology Workshop

Our next evaluation of i*CATchBadges took place in a primary school, in an after-class workshop for students to learn computing and technology. This workshop ran

for 4 lessons and 1.5 hour each. There were 12 males and 6 females, making 18 students in total, with an age range from 8 to 12. All students had no previous programming experience. They were arranged into five groups of three to four by their class levels: three groups consisted of males only (all-male), one group with females only (all-female) and one mixed. This workshop had three parts: 1) Creativity Test – a simple creativity test for children that served as a reference for students' ability of association on their tasks; 2) Graphical Programming Session – teaching the basic functions of Scratch [111], a kind of graphical programming environments with syntax-error-free design; and 3) Tangible Programming Session – teaching the features of the i*CATchBadges system. The last two sessions were designed to compare how students learned computational concepts through these two different kinds of user interfaces, so both sessions also requested students to use the tools to do the same task – presenting a story.

The creativity test was based upon the Torrance Tests of Creative Thinking-Figural (TTCT-Figural) [124] to develop and consult the scoring method to measure the divergent thinking of students in terms of figure. This test was adopted because using electronic functions and felt icons to create different meanings requires students to have divergent thinking and the associations of electronic functions with felt icons as a kind of visual meanings are similar to a picture. The scoring method consists of three categories: 1) fluency – the number of relevant ideas; 2) originality – the number of statistically unusual responses among the test subjects; and 3) elaboration – the number of detail in the responses. Each student was asked to sketch three pictures from given three simple line figures (one sample is shown in Figure 3.15 (a)) within 10 minutes. Given the same figure, the students may have similar idea to complete the figure, but the results of the pictures can be quite different. For example, Figure 3.15 (b) and (c) shows two students' sketches under the same figure: two boys both had an idea to sketch a hand, but one boy drew a simple hand only, and another boy drew a hand with tattoo. Both boys obtained the score of fluency (since the hand is relevant to the given figure) and lost the score of originality (since they have same response on the same figure to draw a hand), but the boy who drew a hand with nails and tattoos obtained the score of elaboration

(since his drawing with nails and tattoos is more details than another boy). Some students may have difficulties to construct a picture with meaning. For example, one girl drew an object but even she could not explain what it is (see Figure 3.15 (d)). The average of score range of each group is between 5.3 and 7.7, which shows that the students' divergent thinking is quite various (see Table 3.3).



(a) (b) (c) (d)

Figure 3.17. (a) The original figure has a simple curve. An example of three students' sketches: (b) A boy's sketch: a hand (c) Another boy's sketch: a hand with nails and tattoos (d) A girl's sketch: no meaning

In the graphical programming session using Scratch, the students were introduced to the basic controls of the sprite such as motion, message display, and conditional and iterative control (around an hour), and carried out a story task (around 40 minutes). The Scratch programming environment was adopted in this session, since it provides a relatively simple graphical user interface for children to create animations without considering syntax errors. Since most of the students did not have coordinate geometry background, we spent much time on the motion control and less time on the other three control functions. However, they still had difficulties to create the expected motions for the sprites in their stories. They mainly used the message display function to present their stories. The all-female groups attempted to use some conditional controls, which they applied to control the message display only, but not in other functions such as motion. Almost no one showed that they could fully understand the conditional concept. When we assigned the story task to the students, we planned to allow them 20 minutes to complete their stories. However, after 20 minutes, no group could finish their tasks. Some of them were struggling with the controls, but more of them were busy typing words for message display. On the other hand, laptop is designed for individual use, so it is common for us to observe that one group mate often took control of the computer

most of the time. As a result, they could not complete the task within the time set. Finally, we extended 20 more minutes for them to complete it. This situation reflects that a computer with one keyboard and one mouse only is difficult to support a group of children to learn collaboratively and effectively.

In the tangible programming session using i*CATchBadges, the students firstly learned the functions of each badge or combination, at the same time, encouraged to couple the functions of the badges with real-world examples (around an hour), and finally they were assigned to create a story (around 20 minutes). The students were eager to speak aloud their imaginations. According to the students' association of the badges, there were three types of the mappings: 1) daily electronic applications, such as LED lights associated with festival lights or traffic lights; the frequency of vibration motor associated with the vibration mode of mobile phones or moving vehicle's engine; the combination of an IR remote control sending signal to the IR sensor with a "du" sound - car alarm or laser gun; 2) natural environment, such as LED lights associated with stars blinking; the frequency of vibration motor associated with natural disasters and earthquake 3) emotions, such as ultrasonic sensor with light output associated with oppressed feeling. Each group's story was used two to three badges and they coupled the badges with one to three representations. Only one all-male group used the combination of the badges - IR sensor with a buzzer that was coupled to a laser gun - in their *Superman Fights with Monster* story, the rest of the groups used the effect of the badges one by one into their stories. For example, one all-male group's *School Accident* story -- a boy sings a song (used a buzzer with a melody) in the singing contest and then earthquake happens (represented by the feature of a vibration motor), and finally the boy is sent to the hospital (used another buzzer with ambulance siren); one mixed group's *fire accident* story – used different kinds of badges to represent a fire-engine (More details refer to Appendix D). During the task period, the students had more discussion with their peers, and most of them worked together in plugging the badges onto the jacket to explore the effects and design their stories. All groups could complete their stories within our stated time period.

Based on the results shown in Table 3.3, there is no significant relationship between the ability of the students' divergent thinking in the TCTT-Figural test and their association ability in their stories. The result indicates that the abstract tangible user interface of i*CATchBadges supports general students to exercise their creativity and explore the computational concepts. On the other hand, based on the results shown in Table 3.4, the students spent less preparation time to create stories of the similar length of their stories in the tangible user interface than that of the graphical user interface. This result consists with our observation during the task period that the tangible user interface supports collaborative working and is more efficient for students to learn this way.

Group	Gender	Number of Students	TCTT – Figural in Average	Number of Associations	Number of Badges
1	Female	4	5.3 (SD=2.6)	2	2
2	Male	3	5.5 (SD=5.3)	2	2
3	Mixed	3	5.7 (SD=3.8)	3	3
4	Male	4	7.0 (SD=2.9)	3	3
5	Male	4	7.7 (SD=1.5)	1	2

Table 3.3. Summary of the five groups' results of TCTT-Figural test, the number of associations and devices applied in their stories

	Graphical Programming Interface (Scratch)	Tangible Programming Interface (i*CATchBadges)
Teaching time	60 min	60 min
Preparation time	40 min	20 min
Storytelling time (average of 5 groups)	1:20 min	1:24 min

Table 3.4. The time for teaching, preparation and storytelling in graphical programming interface (Scratch) and tangible programming interface (i*CATchBadges) sessions

3.5 Discussion

In two case studies in different scenarios, through observations, conservations with the students, and their task outcomes, it was found that our proposed tangible language design supports learning of basic level of computational concepts and provides space for children to exercise their imaginations. This section discusses the directions of the design of the tangible programming systems that facilitate students in learning the computational concepts.

3.5.1 Computational Concepts

This study focuses on the investigation of language design on the tangible user interface. The plug-and-play programming language without syntax error decreases the complexity of the program. It also lowers the barrier and makes it easier for students to construct programs and learn computational concepts. Our hypothesis is that students learn computational concepts by creating a program to present their ideas and following the program flow. Therefore, even the language seems to be quite limited, students still can learn the basic computational concepts through exploration. In the game booth and technology workshop studies, not every student uses the combination of badges to express their ideas. This means that few of them explore the conditional concepts, but at least they did use the badges one after another to express their ideas, which means that most of them understand the concepts of sequences and iterations. Furthermore, in contrast to Scratch, which is a kind of syntax-free graphical programming platforms supporting more complex program design, it was also found that the students had difficulties applying the conditional concept on their stories (see Appendix D). This observation indicates that it is not easy for students to assimilate the conditional concept in a very short period of time (within five minutes to one hour) compared to a 5-day workshop. However, the advantages of tangible interface are clearly shown in the study, which facilitate a physical and sensory engagement, provide room for students to practice imagination, and support collaborative learning. Therefore, the tangible programming language assists students in learning the abstract concepts better.

3.5.2 Task Outcomes

Due to the scalability and the versatility of the tangible user interface, most of the tangible programming languages support simple tasks in monotonous aspects. For example, each puzzle of the Tern language is mapped to one command or variable, which is only used for controlling the motion of a robot. FlowBlocks is another example, which contains a set of abstract construction blocks to represent real-world system such as probability measurement; even though the representations of the blocks are relatively versatile, it is used in causality relationship only.

Our tangible language design could not escape from similar physical limitations of the tangible user interface [56, 134, 135]. It was difficult to support a sophisticated task compared to the graphical or text-based programming environments. However, the functions of the actuator and sensor badges are easier for students to associate with different kinds of representations in terms of functionality and aesthetic of smart clothing, the daily applications and feelings. This design shows that the abstraction interface with concrete physical function of the construction blocks provide a wider scope for discovery, exploration and association, which supports different topics of learning such as science, design and storytelling. In other words, the design better caters for different learning styles and preferences.

3.5.3 Learning Environment

Tangible learning tools support hands-on learning that let children learn actively by manipulating physical objects. If a tangible learning tool is used in different places or activities, students may gain other learning benefits. In our study, the students in two different learning environments - game booth and classroom, could gain some knowledge of computational concepts and wearable computing (see Figure 3.15). In the classroom workshop, the tangible programming system supported the students to work collaboratively on their tasks; in other words, the students had social interaction between peers while they were doing their tasks. In the game booth, even though the students stayed in the booth for only a short time, it does not mean that they learned nothing from it. They had more social interaction between peers and adults in the game booth than in the classroom workshop. Most of the students liked to discuss together how to build their intelligent jackets; some liked to invite their peers to come to our booth and introduce them to the tool; a student gain more ideas from his teacher to design a turtle and rabbit theme jacket with lights. These interactions indicate that the tangible programming system is suitable for running workshops as well as for public area interactive exhibitions.

Currently, more physical or tangible systems are available, not only in research labs or classrooms, but also in public areas such as science museums [57, 114]. The

systems set in public areas attract a diversity of children to interact with technology, and thus children have more opportunity to learn technology outside classrooms. However, those systems usually occupy a large space augmented with sensors, cameras, monitors, and other physical settings. They are not flexible and seem to be better established in the indoor environment. However, the embedded electronic devices are getting smaller and thinner. These advanced materials provide potential to design smaller and even hand-carried tangible programming systems which serve both input and output, and thus they can be flexibly used in outdoor and indoor environment without installation.

Figure 3.18. Two different learning environments: game booth (left) and classroom (right)

3.6 Summary

The tangible language design process was presented through two paper prototype studies, and finalized some important design dimensions of the tangible programming systems. The final design dimensions inspired the development of the i*CATchBadges tangible programming system, and the system was evaluated by two case studies in two different learning environments - game booth and classroom. The students were able to successfully use the system for learning basic computational concepts, and creating simple and meaningful tasks by building simple function badges. The study indicates that the abstract construction interface with concrete physical functions of the tangible programming system provides a potential way for students learning computational concepts through interacting, exploring and association. Furthermore, the tangible programming system serves as both input and output on the same interface, which supports simple installation, and thus it can be flexibly used in different learning environments, such as indoor museums and outdoor exhibition. Therefore, it has potential to draw more attention of a diversity of children engaged in computing and technology.

Chapter 4 Creativity through Crafts Making

As the proportion of female students in computer science has always been small [119], those who wish to see this trend reversed have developed some different programming domains to cater for both male and female learners, such as the multimedia-based programming environments ALICE [2] and Scratch [111], which are related to games and animation design. Some researchers believed these multimedia platforms would not be very successful due to the gender expectation that computers were primarily associated with males and hence female did not play with computers [119]. Some other researchers are looking for new strategies to reduce the gender imbalance in computer science education, such as developing computational construction platforms and motivating children's interests in science and technology [18, 37].

In traditional, computational construction kits are constructed by standard unit blocks, and are only used for building robots and mechanism such as LEGO construction kit Mindstorms [90]. Recently, the rapid development of embedded electronics, wearable computing and e-textiles technology raises the possibility of using computational construction kits for children, or we call it as computational crafts kits. Craft is about making objects by hand where functional and aesthetic considerations are equally important [3, 33], often with the use of different materials such as wood, clay, textile, metal, paper, plastic, and beads. The computational crafts kits are a set of electronic modules, which are constructed with traditional materials together to create artifacts with computational functions and aesthetic design. There are some examples of computational crafts kits, such as Lilypad Arduino [80] which is a wearable construction kit for making soft computational crafts, and paper computing [21] which is used for creating functional computational artifacts on painted paper substrates.

Crafts activity is a kind of traditional hands-on learning in children education. The nature of this activity involves design and creativity, and thus it often attracts both boys and girls. These integrated-technological crafts expand the range of children's constructions with high and low tech materials not only to construct

robots, but also to create artistic sculptures and clothing. Therefore, the computational crafts have the potential to broaden the diversity of the children who are interested in exploring science and computational world through customizing their personalized artifacts.

Even though there were much positive feedback from children using those computational tools for crafts making [19, 70], most research studies mainly focus on the ability of such tools to engage the children's interests; few research studies examine the learning process of children in using different types of the computational platforms for crafts making. This chapter investigates the learning process of children in creating computational crafts. Three kinds of computational crafts toolkits were used in different learning and design objectives in the case studies. The first computational crafts toolkit was the *Lilypad Arduino with TeeBoard platform* [94], which supported learning electronic circuitry concepts and circuitry design on a T-shirt; the second computational toolkit was the *i*CATch platform for apparel* [95], which focused on learning computational logic and programming design on clothing; the last toolkit was the *i*CATch platform for robot*, which enabled robotic construction and robotic car movement control. The study also examines which of the three platforms would be the most conducive to creating innovative applications of the technology. There are four research questions to investigate:

- How does the design of the construction platforms affect the space of creativity, the curriculum design and the learning process of children?
- What kinds of crafts are created by children under different objectives of the computational platforms?
- How do children design the program on their crafts?
- Are there any factors will reduce the learning engagement between children and computational crafts activities? (E.g. students' characteristics, domains of tasks and types of programming environments, etc.)

The remainder is arranged as follows: Section 4.1 introduces the technology of three types of computational construction platforms used for the case study. Section 4.2 describes the course information of the outreach programs and presents the

evaluation results of the children's work and feedback. In section 4.3, discussions are presented about the learning process and the challenges of the children using computational construction platforms to learn computational concepts. A summary follows in section 4.4.

4.1 Computational Platforms for Construction and Materials

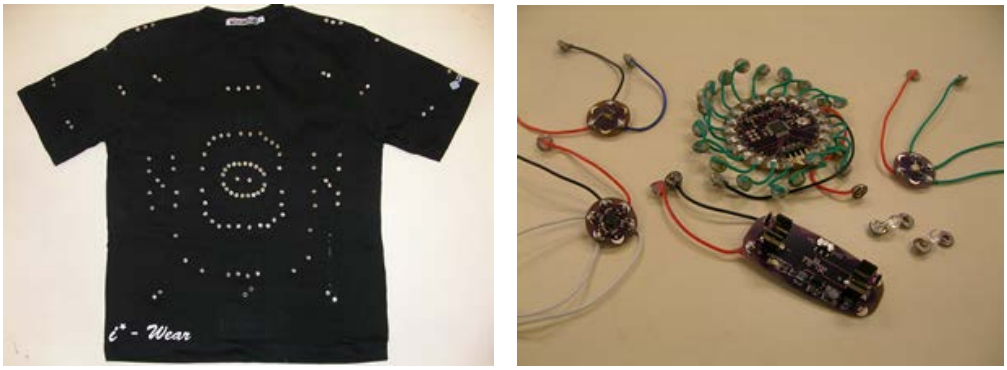
In order to investigate how children use computational toolkits for construction to learn computational concepts, three kinds of computational toolkits for construction were used: *TeeBoard and LilyPad Arduino*, *i*CATch for apparel* and *i*CATch for robots*, all of which are based upon the Arduino platform and are integrated with three science topics respectively: electronic circuitry, computational logic and robotics. To examine the degree of engagement related to the nature of computational craft activities rather than other factors such as the types of programming environment, three kinds of programming environments were designed to support writing Arduino programs: *Arduino Integrated Development Environment (IDE)* which is a kind of text-based programming environment; *BrickLayer* which is a kind of text-enhanced graphical programming environment; and *i*CATch IDE* which is a kind of hybrid text-icon-based programming environment. Craft materials should provide more space for children to design and create their artifacts, so we provided some traditional craft materials like paper and cloth rather than some well-designed standard-sized blocks for children.

4.1.1 Computational Toolkits for Construction

TeeBoard and LilyPad Arduino

TeeBoard is a reconfigurable t-shirt integrated with a breadboard (Figure 4.1 (a)). TeeBoard is designed for young learners to construct circuits easily on a garment similar to a conventional breadboard, which supports the teaching of electronics by making it easy to construct electronic circuits. Students can snap the microcontroller, actuators and sensors in place onto the TeeBoard, and program the microcontroller to control actuators and sensors. The LilyPad Arduino [80] is a set of sewable

electronic components including microcontroller, actuators and sensors, which is designed for wearable computing and e-textiles. As the LilyPad does not have plug-and-play capability, we modified the LilyPad components with snap buttons to connect to the TeeBoard substrate and the ribbon-wrapped pieces of conductive fabric served as wires (Figure 4.1 (b)).



(a) (b)

Figure 4.1. (a) The TeeBoard construction interface (b) The modified LilyPad components with snaps buttons

i*CATch Apparel Platform

i*CATch [95] is a scalable, extensible wearable computing toolkit, which consists of a set of plug-and-play electronic components includes LED lights, buzzers, a light sensor, an ultrasonic sensor and switches, and a pluggable jacket construction platform. It is similar to the TeeBoard/LilyPad wearable computing platform specially designed for use by novices and support wearable computing teaching in the classroom. The main difference between the i*CATch and the TeeBoard/LilyPad platforms is the communication architecture. The i*CATch apparel platform uses inter-integrated circuit (I2C) bus technology [65] as the communications channel between the electronic devices, in which there is no need to match input and output ports between the main controller and the peripheral device. The TeeBoard/LilyPad platform uses point-to-point connections, in which individual input and output pins on the microcontroller connect directly to those on the peripheral input devices such as sensors and output devices such as actuators. Hence, in the i*CATch platform, there is no need to trace connection lines or connect extra ribbon wires to control the devices.

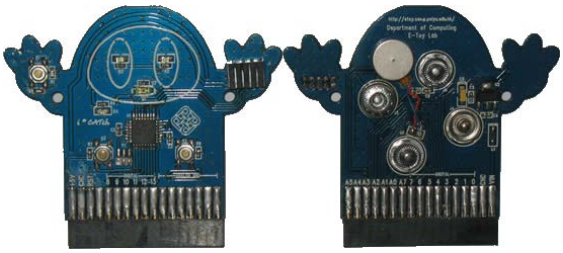


Figure 4.2. The i*CATch main board

i*CATch Robotic Platform

The i*CATch platform is designed for wearable computing, and it is also supported robotic computing. In contrast to the wearable construction kit, there are two motors and one interface box (Figure 4.3) extra for the robot construction kit. The interface box contains six ports, and each port is used to connect to either input or output device. In the robot version, the platform uses point-to-point architecture. In order to connect the wearable electronic devices to the interface box, a robot adapter block (Figure 4.4) is designed for converting the connection interface of the wearable electronic device into the connection interface of the robot construction.

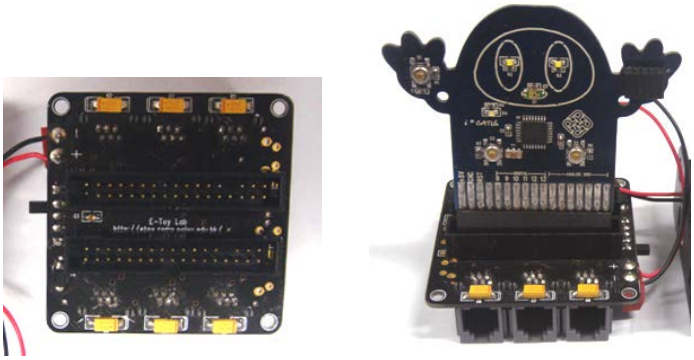


Figure 4.3. An interface box for i*CATch robot version (left) and the i*CATch main board inserted in the socket of the interface box (right)

Figure 4.4. A robot adapter block: the side of the connection interface for the robot interface box (left) and the side of the connection interface for the wearable electronic devices (right)

4.1.2 Programming Environments

Arduino IDE: Text-based programming environment

The Arduino IDE [6] is an open-source text-based programming environment. It allows users to write programs in C language to control an Arduino main board or ATMEGA168 and ATMEGA328 microcontroller. It also provides a standard library to support *Two Wire Interface (TWI)* or *Inter-Integrated Circuit (I2C)* for sending and receiving data over a net of actuators or sensors. However, C language has a number of syntactic rules, which is only suitable for the experienced programmers. Therefore, we developed a library called *i*CATch sensor* [60] to modularize codes into functions to control the I2C devices. Programmers only need to know the function parameters like I2C address and output value of the actuator. This library makes it easier for programmer to write a program to control the i*CATch devices, even children who only have limited programming experience can still handle it. Figure 4.5 shows the Arduino IDE and a sample i*CATch program.

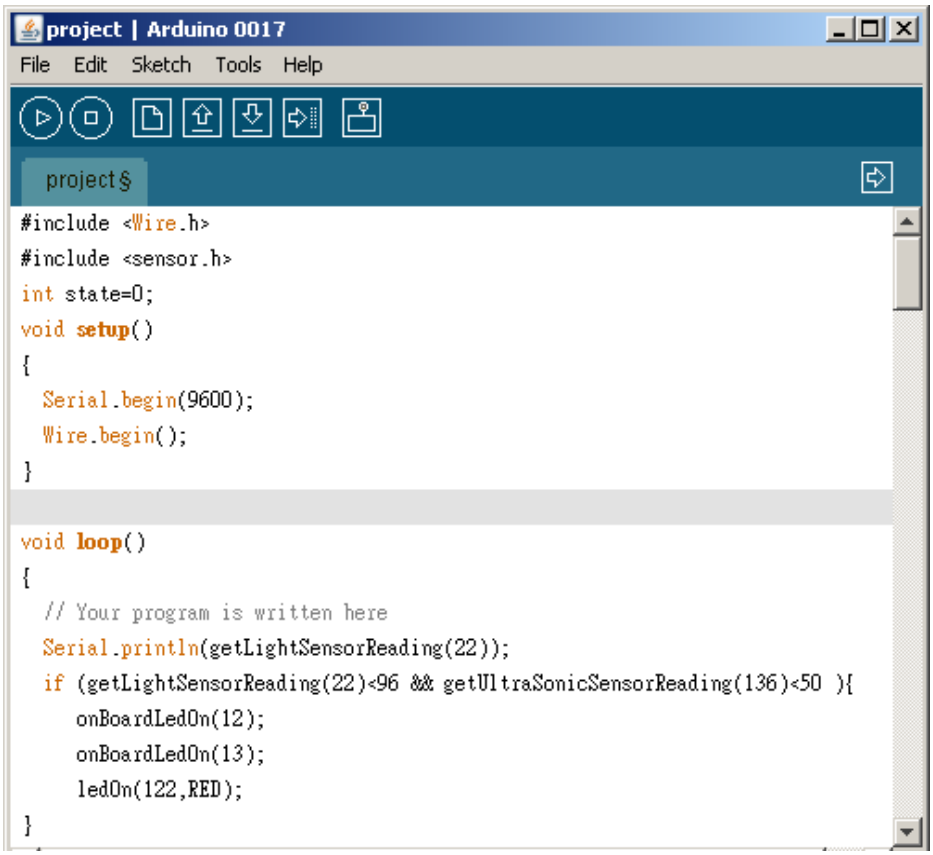


Figure 4.5. The Arduino IDE and the sample i*CATch program code

BrickLayer: Text-enhanced graphical programming environment

BrickLayer [14, 26] is a text-enhanced graphical-based programming environment for programming the *Arduino* microcontroller. BrickLayer consists of three areas: *brick area*, *construction area* and *source code area* (Figure 4.6). Each graphic brick on the *brick area* represents one programming construct. Students build programs by dragging the brick from *brick area* and dropping it onto the *construction area*, and the corresponding program code will be auto-generated on the *source code area*. As we do not expect that our young learners are good at typing and have any programming experience, this text-enhanced graphical programming environment is designed to free students from having to worry about the syntax; but at the same time, they are also presented with the syntactic statements so that they learn to make a connection between the program structure and programming statements.

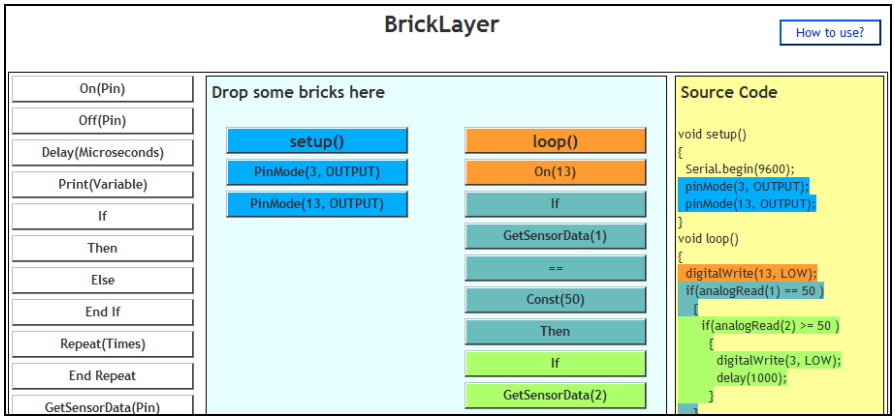


Figure 4.6. The BrickLayer's interface: A brick area (left), construction area (middle) and source code (right) [26]

i*CATch IDE: Hybrid text-icon-based programming environment

i*CATch IDE is a hybrid text-icon-based programming environment [60], which is also designed for programming the Arduino platform. The idea of this programming environment is inspired by Bricklayer [26] and Robolab [101], which allows programming by drag-and-drop icons that represent programming constructs and joining them together to denote a program flow, thus giving the user to have a graphical view on control structures such as conditionals and repetitions; at the same time, the actual source code is generated immediately, thus allowing the user to actually see what is being generated and sent to the i*CATch main board, thus

facilitating a later switch to pure text-based programming (Figure 4.7). In addition, this IDE supports both point-to-point architecture and bus-based communications between the main board and the peripheral electronic modules, and therefore we replace the BrickLayer platform with i*CATch IDE in the workshops with using i*CATch devices.

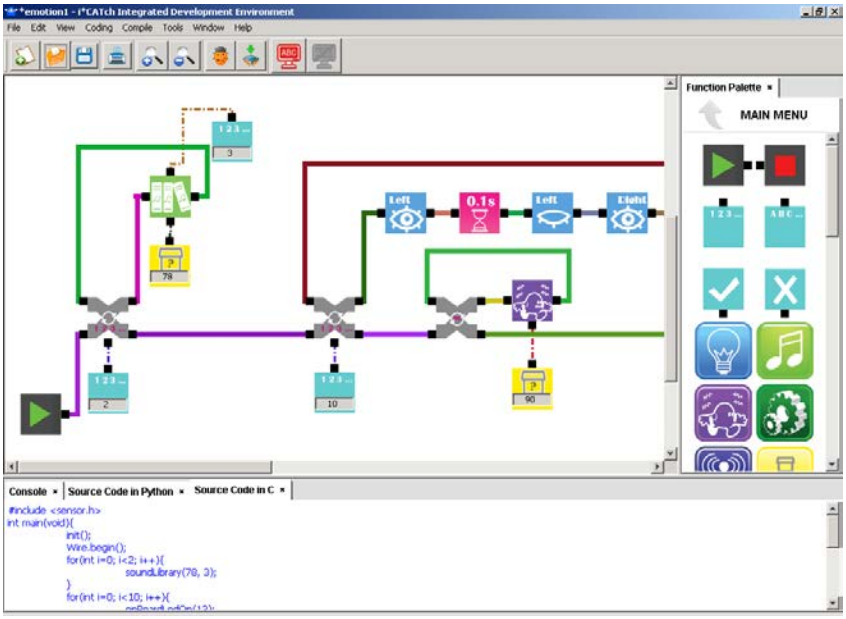


Figure 4.7. The i*CATch IDE's interface.

4.1.3 Craft Materials

In the workshops, we provided craft materials including "low-tech" materials such as fabric, color pens, and some recycle materials, as well as "high-tech" materials such as electronic components designed for computational clothing or robots. In order to provide more space for children to design and create their computational artifacts, the traditional craft materials were used like fabric and paper rather than the standard unit wooden or plastic blocks. Because the length and the course content of the workshops were slightly different, the set of the craft materials that was provided was also slightly different, but all the provided set of materials was sufficient to achieve the course objectives. Table 4.1 summarizes the materials provided in our workshops.

Construction Toolkits		TeeBoard and Lilypad Arduino	i*CATch for Apparel	i*CATch for Robots
High-Tech Materials	Construction Platforms	TeeBoard with conductive ribbon-wires	Jacket with integrated communication bus	Interface Box with LAN Cables
	Input Sensors	Accelerometer Light sensor	Switch Joystick Infar-red sensor Light sensor Ultrasonic sensor	Switch Ultrasonic sensor Light sensor
	Output Actuators	LEDs	LEDs Buzzer Vibration motor	LEDs Buzzer Motor
Low-Tech Materials		Felt scraps	Cloth scraps Paper Ribbons Fabric paint pens	Carton paper Magazine Plastic bottles Balloons Color marker pens

Table 4.1. Materials provided in three kinds of workshops

4.2 Research Methodology

We explored how children used the computational toolkits for crafts making to achieve the learning benefits along four angles: the flexibility of the construction interfaces for children to build their crafts iteratively; the creativity space of the platform for children to exercise; the design of the programs; engagement factors such as workshop theme, construction interface, nature of tasks and learner characteristics. We used three sets of computational toolkits for making different kinds of crafts: TeeBoard with Lilypad Arduino, i*CATch for Apparel and i*CATch for Robots respectively in our workshops to collect data over the past two years. Each set of the computational toolkits combined with different programming environments was used in at least two workshops.

All workshops were taught or assisted by our undergraduate computer science students and the project team members. In average, the instructor student ratio was about 1:6. We mostly conducted four kinds of evaluation in our workshops: observations, interviews, surveys and project outcomes, and documented by videos and photographs. We also tried to record the design progress of students to quantify the creativity space of the construction interfaces in one of the workshops using i*CATch for apparel. This section presents three kinds of course background and the

results of three case studies. Further analysis of three case studies is discussed in section 5.

4.2.1 Courses Background

The workshops were organized for primary school students (aged 8-12) and secondary school students (aged 12-15). The participants were recruited via open advertisements or some via particular schools. The class size of the workshops was related to the results of the recruitments each time without any rejection, usually around 10 to 30 students in each class. All workshops were held in a school environment. Table 4.2 summarizes three syllabi of our workshops (see the details in Appendix A). In order to cater for the younger children, the length of the workshops for primary school was usually shorter than the secondary school one, but the contents of the syllabus were roughly the same.

<table>
<tr><th colspan="2">Computational Toolkits for Crafts Making</th><th>TeeBoard with LilyPad</th><th>i*CATch for Apparel</th><th>i*CATch for Robots</th></tr>
<tr><td colspan="2">Learning Contents</td><td colspan="3">Example Tasks</td></tr>
<tr><td colspan="2">Robotics</td><td>N/A</td><td>N/A</td><td>Motor controls
Building a robotic chassis</td></tr>
<tr><td colspan="2">Electricity and Circuitry</td><td>Complete circuits
Serial circuits
Parallel circuits
Short circuits</td><td>N/A</td><td>N/A</td></tr>
<tr><td colspan="2">Electronic devices</td><td colspan="3">Output devices (e.g. LEDs, buzzers, etc.)
Input devices (e.g. Light sensors, switches, etc.)</td></tr>
<tr><td rowspan="3">Computational Concepts</td><td>Sequential logic</td><td colspan="3">Blink a multicolored LED in rainbow color sequence; play a melody on buzzer, etc.</td></tr>
<tr><td>Conditionals</td><td colspan="3">Turn on a LED in the dark, give a sound when too close to a wall, etc.</td></tr>
<tr><td>Repetitions</td><td colspan="3">Set LEDs to flash repeatedly in reciprocal pattern on the clothing; or set the car to move a square pattern, etc.</td></tr>
<tr><td colspan="2">Project</td><td colspan="2">Design an e-fashion</td><td>Solve challenging tasks</td></tr>
</table>

Table 4.2. Three syllabi of our workshops (see the details in Appendix A)

4.2.2 Case Study 1: TeeBoard with LilyPad

We organized two workshops which used the TeeBoard with LilyPad platform. The main differences between the workshops were the age group of students, academic

background of students, the length of the course and the programming environment. One workshop was designed for middle school students. There were 19 boys and 6 girls, making 25 students in total, with an age range from 11 to 16, and academic background from public and private schools. It ran for 5 full days (around 27 hours in total) and used BrickLayer, a text-enhanced graphical programming environment, for students to write their programs on the LilyPad main boards. Another workshop was designed for primary school students. There were 7 boys and 13 girls, making 20 students in total, with an age range from 8 to 12, and they were studied in the same school. It took place over 9 weeks with 1.5 hour each week (around 13.5 hours in total) and used i*CATch IDE, a hybrid text-icon-based programming environment, for students to write their programs. In the workshop for the middle school students, we divided the students into eight groups of three to four according to their age and gender: six all-male groups and two all-female groups. In the workshop for the primary school students, the school teacher helped us to separate the students mainly by age into five groups of four students each: one all-male group, two all-female groups and two mixed.

The themes of the two workshops were similar: the one for middle school students was about sports; another one for primary school students was about smart t-shirt. The objective of both workshops was to introduce students to programming and electronic circuitry, and exercise their creativity as well as the newly learned programming and electronics knowledge through the final project: make their own interactive garment. More descriptions of the syllabus of the workshops are shown in Table 4.2 and Appendix A (A).

Instructors' Feedback

The class activities were usually observed by our instructors and project team members. Our project team members also interviewed students while they were carrying out the tasks or the presentations to understand more what the students' ideas. According to our instructors' reports, the responses of the primary and middle school students were similar: while the students did not show much interest in learning about electrical theory (voltage and resistances, etc.), they became excited

and animated when they were presented with the TeeBoard and immediately started trying to make their own circuits. They experimented the TeeBoard on their own, connecting the LEDs and wires in all sorts of configurations, trying color mixing, and even testing the surrounding light level. This indicates that e-textiles and wearable computing seems to be an appropriate teaching medium as the students find it interesting and exciting.

In performing the assigned tasks, there were some differences between the primary and middle school students. In the middle school workshop, our instructors observed that most of the all-boy groups focused on finishing the tasks as quickly as possible, without paying much attention to the aesthetics of the final product. In contrast, the all-female groups worked slowly and methodically, considering factors such as design, colors and patterns. They also inserted their own interpretations into their assignments: for instance, they designed a circuit in which the color of the multicolored LED changed according to the colors of the rainbow. When asked to explain their idea, they stated that they chose the rainbow pattern as it represented hope. In the primary school workshop, both the boys and girls groups focused on finishing the tasks as quickly as possible. While the all-female group knew that other groups did not complete their task, they started to consider the colors and patterns to revise their design. The all-female group's action motivated the all-male group to design their work. This working style also shows that girls appeals to design and make art works.

According to the instructors, the students experienced the most difficulty when first presented with the TeeBoard, as they did not really understand the whole structure of the conductive strip pattern at first. Most of them constructed their circuits using direct connecting ribbon wires rather than exploiting the strips on the TeeBoard. However, once the concept was explained to them, they picked it up quickly and had no trouble using the TeeBoard in the way that it was designed for. Even though they understood the structure of the TeeBoard, one all-female group from the primary school workshop still preferred to use the extra color ribbon wires to create the forms of beauty. The instructors did not find that the primary or middle school students had difficulty to use the programming environments, either

BrickLayer or i*CATch IDE. Some of the middle school students tried to modify the text code, because they had prior programming experience.

Survey Results

An informal survey was done before the lesson started. More than half of the middle school students had some programming experience with LEGO NXT robots and/or Scratch, but most of them did not feel that they knew much about programming. All of the primary school students did not have any programming experience. We also conducted a survey to collect the students' feedback at the end of the workshops. The result of the level of difficulty of the course reveals that the primary school students had more difficulty than the middle school students in the course (see Figure 4.12). This result is to be expected, as electronic circuitry and wearable computing would certainly be new and unfamiliar to the primary school students. Most of the primary and middle school students also felt that the programming element was difficult. This is due to the nature of programming the Arduino microcontroller, which involves physical constraints as well as logical issues. For example, when asked to program a toggle switch to control the LED, most of the students were able to come up with the idea of creating a variable to store the stage of the LED (on or off) by themselves. However, when asked to write a program to flash the LED on and off, most of the students did not realize that a timed delay was necessary to slow down the program to make the LED's light visible to the human eye. Figure 4.13 shows both primary and middle school students gave very positive feelings to their experiences. According to our survey results, almost all of the students felt that our course was interesting in the design, circuitry and programming sections. One 12 year-old primary school girl said she was not interested in traditional science subjects, but she felt that the tasks involving science and crafts were very interesting.

Students' Projects

We were delighted by the fact that all 13 groups in both primary and middle school workshops were able to achieve the course objective and create a final project. Most

of their final products was very creative and showed that their design both creative and technologically challenging. One of the all-male middle school groups put two accelerometers on the arms of the TeeBoard to control patterns of flashing LEDs, thus LEDs that would flash in different colors and patterns that would change according to hand motion (Figure 4.8 (a)). One of the all-female middle school groups chose the creative way and created a smiley-face t-shirt, using two multicolored LEDs for cheeks, whose color was controlled by the readings from the accelerometer. When the face was patted lightly, moving the accelerometer along the z-axis, the color of the two LEDs changed to red, creating a "blushing" effect (Figure 4.8 (b)). One of the all-girl primary school groups also showed their creativity and designed an interactive picture t-shirt, moving an accelerometer to light up the LED embedded on the felt sun, and using the back and forth pattern of flashing LEDs between two felt ladybugs to represent their conversation (Figure 4.8 (c)). A sample program of their projects can be found in Appendix E.

(a) (b) (c)

Figure 4.8. (a) A middle school boy shows how to control the LED pattern by moving his hands. (b) A middle school girl explains the design of her t-shirt: a smiley-face with blush on its cheeks. (c) An interactive t-shirt is done by four primary school girls: sunrise and the two ladybugs' conversation.

4.2.3 Case Study 2: i*CATch Apparel Platform

More than two workshops used the i*CATch apparel platform; we selected one workshop for primary school students and one for middle school students as the samples of our research findings in using the i*CATch toolkit. In the primary school workshop, there were 9 girls, with an age range from 10 to 12, and come from the same school. It ran for 5 half days (around 15 hours in total) and used the Arduino IDE with our i*CATch functions library, a text-based programming environment, for students to write their programs on the i*CATch main boards. We let the students form their own groups of one to three students each, and had four groups

finally. In the middle school workshop, there were 18 boys and 4 girls, making 22 students in total, with an age range from 11 to 14, and academic backgrounds from both public and private schools. It took place over 5 full days (around 27 hours in total) and used i*CATch IDE, a hybrid text-icon-based programming environment, for students to write their programs. Students were encouraged to form their own groups of approximately four students each: four all-male groups and three mixed.

The theme of the two workshops was e-fashion and intelligent clothing. The overall objective for both workshops was to introduce the students to programming and electronics through wearable computing. More descriptions of the syllabus of the workshops are shown in Table 4.2 and Appendix A (B).

Instructors' Feedback

Based on our instructors' observation, the students were excited, both all male and female groups from primary and middle school workshops, when they were introduced to the i*CATch toolkits. They immediately asked the functionality of the electronic devices and tried to plug the electronic devices on the i*CATch jacket to observe the effects. In the bus-based i*CATch toolkit, the circuitry is abstracted into the design of the bus and the interface sockets, and therefore the students did not spend time to learn about the electricity and circuitry concepts. In addition, the design of the interface sockets with the male and female snap fasteners prevents the user from plugging in a device backwards, and therefore we did not find that the students plugged in the devices on the jacket substrate in a wrong way.

In doing the assigned tasks, the instructors reported that the students were very interested in working on multicolored LEDs and buzzers. The students who had music background, they like to compose a melody; the other students without music background like to play a tone or a sound from the library. The students were also excited to experiment with the input sensors. In one of the tasks, the students were assigned to use an ultrasonic sensor to create an "intelligent stick" for blind people. They put on their programmed jacket and walk along the corridor with their eyes closed. They could apply their newly learned programming skills into other input sensors such as light sensor as well. In general, they did not have much difficulty to

program light sensor to turn on a light in the dark. In the final project, both the boys and girls engaged in designing their intelligent jackets. The majority of the boys group focused on functionality design and simple abstract pattern. In contrast, most of the girls group spent more time integrating felt decoration into electronic devices than writing program, except one 11-year-old primary school girl who was enthusiastic about science and technology and already had much programming experience on MicroWorlds, NXT robot, Scratch, Flash and 3ds Max programming tool. Due to the time limitation, we did not introduce the usage of IR sensor to the students, but that girl begged us to give her a remote control to test the result of the IR sensor used in her intelligent hiking jacket.

Around half of the middle school students were observed that were eager to modify the text code in the i*CATch IDE, it was likely that they had prior programming experience. It was also reported that most of the primary school students could cope with the text-based Arduino IDE augmented the i*CATch functions library, which is highly related to all of them having typing training and programming background in text-based programming environment such as MicroWorlds.

Survey Results

A survey was conducted at the end of the lessons to evaluate the workshops and the i*CATch toolkit. Figure 4.12 shows the results of the level of difficulty of our workshops for primary and middle school students. Only close to 10% of the primary or middle school students felt it was difficult to complete our tasks. This result is encouraging as it indicates that the tasks associated with the bus-based i*CATch computational platform is suitable for a wider range of students to learn computational concepts. We also asked the students about their feelings on our courses. Almost 95% of the students liked the topic of e-fashion and intelligent clothing and using wearable computing as a teaching approach of learning computational concepts (see Figure 4.13). Most of them said they learned the concepts of smart clothing, clothing design and programming concepts.

(a) (b) (c)

Figure 4.9. (a) A middle school boy decorates their group' jacket. (b) A multi-function casual wear is created by a group of two middle school girls: the pink felt heart shows the wearer's emotion, the purple felt clock tells time and the yellow felt flower massages the wearer's stomach. (c) A girl demonstrates their hi-tech hiking jacket which could play a song by using a remote control.

Students' Projects

All 10 groups of the primary and middle school students were able to generate fashionable and intelligent jacket in the final project. One of the all-male middle school groups designed their smart clothing for blind people with four functions: aesthetics – generating the colorful lights associated with the colorful check pattern on the jacket; call for help – flashing white light in the Morse code pattern SOS with a long tune; money checker – detecting the color of banknotes, thus the denominations of banknotes, through a light sensor; blind man's cane - alerting distance through an ultrasonic sensor into vibration felt by the wearer. The first three functions were controlled by a switch, and the last function was available when the jacket was turned on (Figure 4.9 (a)). In general, the all-female groups' projects were more creative. In the following two examples, they not only considered the aesthetic factor, but also the functional factor of their smart clothing. One of the all-female primary school groups designed a multi-function casual wear: using a multicolored LED integrated into the pink felt heart, each color represented an emotion such as blue was sad and green was sick, and whose color was controlled by the joystick; using a buzzer embedded into the purple felt clock, every hour played with different sounds; using a vibration motor hidden in the yellow felt flower acted as a massage machine, creating a series of vibrations when the switch was pressed (Figure 4.9 (b)). Another all-female primary school groups made a hi-tech hiking

jacket with four functions: illumination through the integrated LEDs into two arms of the jacket; neck massage through a vibration motor at the back of the neck; follower detection through an ultrasonic sensor at the back, playing "iPod" music through a buzzer controlled by a switch, even an IR remote control, thus the wearer's companies were able to choose other music to play (Figure 4.9 (c)). A sample program of their projects can be found in Appendix F.

Design Progress

We encouraged students to start with a draft of the design before they developed their projects as a reference, particularly to the e-fashion design project. We also designed a method to have a deeper analysis of the computational construction platform for smart clothing, because the concepts of e-textiles and wearable computing are comparative new to children. The method measures the creativity space of a construction platform by comparing the number of the similarity of the ideas between four learning stage (see Figure 4.10). We supposed that if the difference of the degree of similarity between the transitional stages is small, and the difference of the number of ideas between the initial and final stage is small, the construction platform will support certain level of creativity space for children to design.

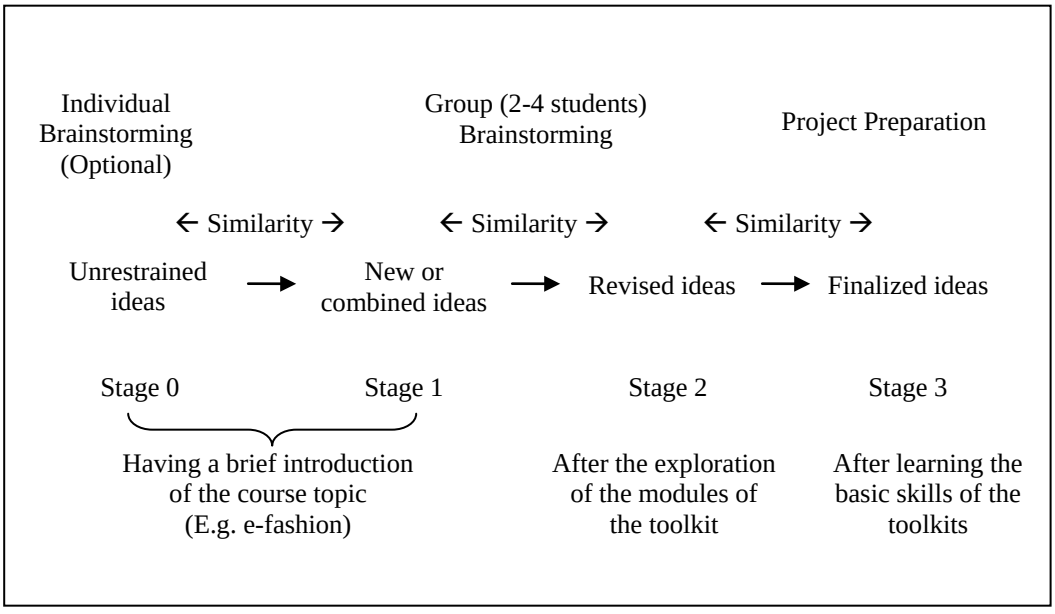


Figure 4.10. Illustration of the measurement of the creativity space of a construction platform

We applied this method in a 5-half-day wearable computing workshop for primary school students held in 2010 summer. The class began with the introduction to the latest applications of the e-textile and wearable computing, after that we asked each student to design their dream e-fashion and draw it down. We recorded the students' unrestrained ideas in this moment (stage 0). Then, we let them form a group to have further discussion on their designs. In this "group brainstorming" time, students combine or generate some new ideas for their projects (stage 1). The "individual brainstorming" process is optional if the children are assigned to work in a group, not individually. In the beginning of the second day of the workshop, the students were asked to revise their design. In this "revising" stage (stage 2), the students had already known the basic modules of the provided wearable computing toolkit i*CATch. On the fourth day of the workshop, the students were asked to finalize their design of the final project. In this "finalizing" stage (stage 3), the students learned the basic programming skills to build functions through the toolkits.

We counted the number of ideas in each stage and the number of changes of the ideas between the two stages of each group. The number of idea was based on the students' draft and presentation to list out the features of the e-fashion in terms of aesthetic and functionality. Aesthetics referred to flashing light pattern; functionality referred to some daily used electronics such as music player, clock, heater, fan, temperature and GPS map, and some fantasy applications such as friend detector and emotion-face display. From stage 0 to 1, the changes of the ideas were mainly due to the effect of brainstorming in group. From stage 1 to 2, the students were introduced to the provided electronic modules that were more basic and simple. The limited modules were not able to fulfill all the students' ideas. Thus, some ideas obviously had to be given up such as fan, GPS map and friend detector. However, some students still insisted in their ideas. On the other hand, some students used some available modules in unexpected ways to create new ideas. For example, vibration motor was used to be a massage machine; friend detector was changed to follower detector by using ultrasonic sensor to detect obstacles. The inspiration of the available modules is another reason to increase the number of ideas. From stage 2 to 3, the students learned the basic skills on programming the electronic modules.

Some insisted students confirmed that some ideas were impossible, such as temperature; some students wanted to challenge themselves to add some new functions, such as light sensor to detect the brightness of environment to change light flashing pattern. Therefore, the number of ideas increased and decreased in each stage. Finally, all groups were able to use the limited modules to represent their creative ideas. For examples, music player - the students programmed a buzzer to play two melodies and a joystick to switch the music; clock – they programmed a buzzer to play 12 different tones respectively after each period of time; emotion display – they changed to use different colors instead of faces to represent different emotions.

Table 4.3 shows the degree of the similarity of the ideas between each stage of four groups. The result indicates that the number of changes of the ideas from stage 0 to stage 1 is slightly more than the other transitional stages. A possible explanation for this difference is that the group brainstorming tends to produce more effective ideas than the individual brainstorming. The result also shows that the degrees of the similarity of the ideas between the transitional stages are almost the same, and the difference of the number of ideas between the initial and final stage is small. Therefore, according to our hypothesis and the quality of the students' projects, the i*CATch platform for apparel design supports enough creativity space for students to design.

Group	# of Students	Stage 0 to 1			Stage 1 to 2			Stage 2 to 3		
		# of Ideas		Similarity	# of Ideas		Similarity	# of Ideas		Similarity
		*S0	S1		S1	S2		S2	S3	
1	1	7	+4	3	4	4	3	4	3	3
2	2	**(2, 2)	2	0	4	3	3	3	3	2
3	3	(2, 2, 2)	5	2	5	6	5	6	5	4
4	3	(2, 2, 4)	4	4	4	4	4	4	3	2

* S, where S is the abbreviation of stage

** (x, x), where x is the number of the individual idea

\+ As there was only one student in a group, one of our instructors discussed with the student during the group brainstorming section.

Table 4.3. The degrees of the similarity of the ideas between each stage

4.2.4 Case Study 3: i*CATch Robotic Platform

More than two robotic workshops using the i*CATch platform were carried out, we selected one workshop for primary school students and one for middle school

students as the samples of our study. There were 12 girls and 25 boys in the primary school, with an age range from 8 to 12, and there were 3 girls and 27 boys in middle school, with an age range from 11 to 14. All students were enrolled from both public and private schools. The primary and the middle school workshop ran for 5 half days (around 15 hours in total) and 5 full days (around 27 hours in total) respectively. Both workshops were available the i*CATch IDE, a hybrid text-icon-based programming environment, for students to write their programs. Students were encouraged to form their own groups of approximately four students each: three all-female groups and six all-male groups in the primary school workshop; and one all-female group and seven all-male groups in the middle school workshop.

The theme of the two workshops was basically the same: green robots. The theme was called “green” robots as the provided low-tech construction materials were recycle materials such as carton paper, magazine paper and plastic bottles instead of the standard unit blocks like LEGO. Making robots with use of different materials provides more challenging for construction and more space for creativity. The objective for both workshops was to introduce the students to programming and electronics through robotic computing and make their functional and aesthetic robots. More descriptions of the syllabus of the workshops are shown in Table 4.2 and Appendix A (C).

Instructors’ Feedback

In constructing the robotic cars session, our instructors reported that boys and girls, primary and middle school students concentrated on their craft work. Some of them even kept building their cars during the break. They did not feel that cutting the carton paper or plastic bottles to build their robotic chassis was troublesome, but they felt it was challenging and had fun with it. Most of the primary school students encountered difficulties to construct their robotic cars such as the servo wheel mounting problem that the wheel was easily detached from the servo; or the size of wheels was too small to support the load of the car’s weight and electronic devices, one reason may be because they had fewer experience on constructing models compared to the secondary school students. Despite the difficulties of construction,

most of them still enjoyed the construction process. Some of them also cut out some magazines to decorate their cars. This indicates our approach that craft activity could attract both boys and girls to learn and work on.

In performing the programming tasks, in general, the male groups always finished faster than the female groups in both primary and middle school workshops. Instructors observed that the programming ability of the primary school students was weaker than the secondary school students. It is obvious that the construction time of the robotic car was almost the same in the primary and secondary school workshops; the primary school students had fewer time to solve the problems and practice programming skills. On the other hand, this could be because the middle school students had more programming experience. Basically, the assigned tasks were similar to the general NXT workshops such as following a black line and moving backward when too close to a wall, but the big difference was that the students had to consider how to fix the sensors on the carton car, which is more difficult than using the standard-size blocks. For example, one all-male group demonstrated their car to move backward when the obstacle was detected, but they were not successful at the first time even though their program was correct. The boys did not understand why they could not achieve the task, until they found that the position of the sensor had shifted. A sample program of their programming tasks can be found in Appendix G.

Survey results

A survey was conducted to collect the students' feedback at the end of the workshops. Figure 4.12 shows that the primary school students encountered more difficulties than the middle school students. Of 24 primary school student's feedback, over 20% had difficulty in the tasks, whereas among the 29 middle school student's feedback, less than 5% had difficulty. Figure 4.13 indicates that the primary school students were less interested in our course than the middle school students. Almost 75% of the primary students were interested in our course using recycled materials to construct robotic cars and the assigned tasks, whereas more than 95% of the

middle students were interested in it. It is apparent that there was a strong correlation between students' interest and the degree of difficulty encountered.

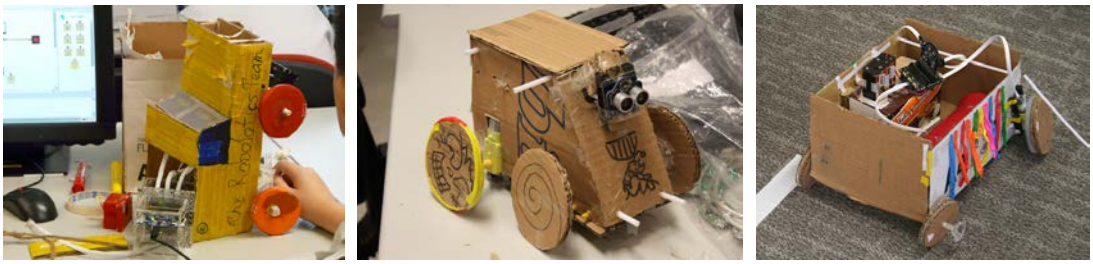


Figure 4.11. A yellow truck is created by a group of four middle school boys (left). A trapezium-shaped car with an ultrasonic sensor on the top is constructed by another group of four middle school boys (middle). An open-top bus decorated with colorful balloons on two sides is produced by a group of four primary school girls (right).

Student Projects

All 9 groups of the primary school workshop and 8 groups of the middle school workshop were able to construct their robotic car using the recycled materials, such as carton paper and plastic bottles. Despite their cars' wheels sometimes were not fixed well during the task demonstration, they could achieve the task after the repair. Figure 4.11 shows the examples of the features of the robotic cars created by the middle school students and the primary school students. The variety of the design in terms of structure and size was found in the middle school student's cars. For examples, a group of four middle school boys created a yellow truck; another group of four middle school boys created a trapezium-shaped car with an ultrasonic sensor on the top. In contrast, the similar design was found in primary student's cars. Most of the structure of the car was a rectangular box. For example, an open-top bus in rectangular structure was produced by a group of four primary school girls. The structure of the middle school student's car was relatively sturdier than that of the primary student's one. Figure 4.11 also shows the examples that the all-male groups usually focused on the structure design while the all-female groups considered the decoration for the cars more than the car structure. It was difficult to find an example of the all-male groups' cars with decorative detail that was similar in effort to a group of four primary school girls' open-top bus with colorful balloons on two sides.

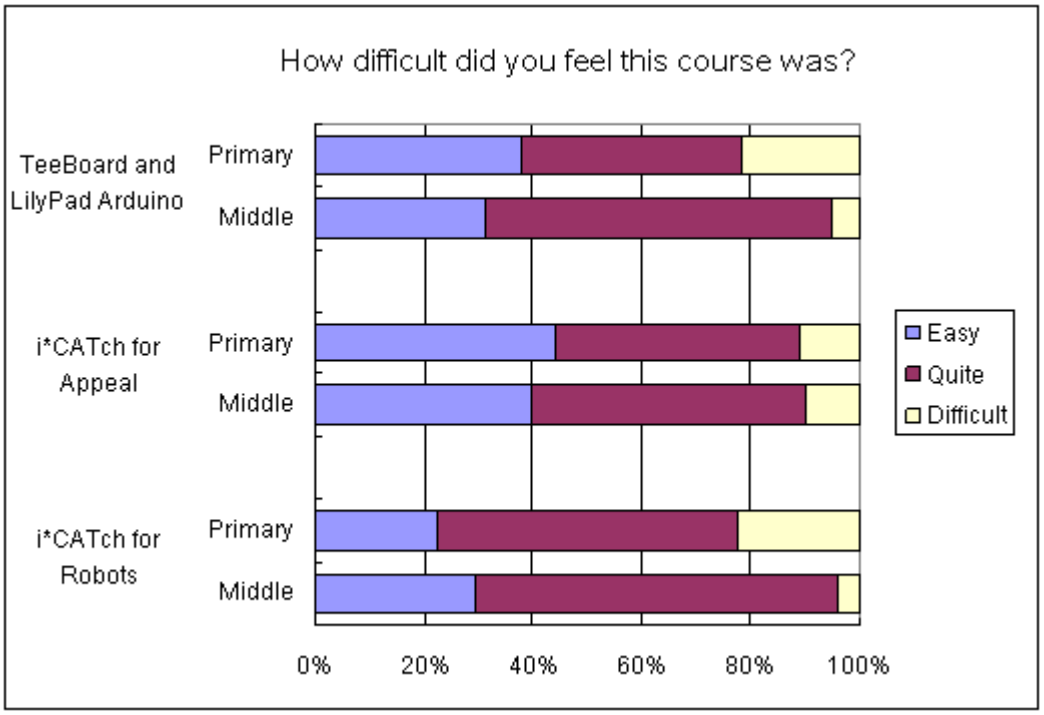


Figure 4.12. Summary of the results of the student's feedbacks from three kinds of workshops showing the student's difficulty on the course

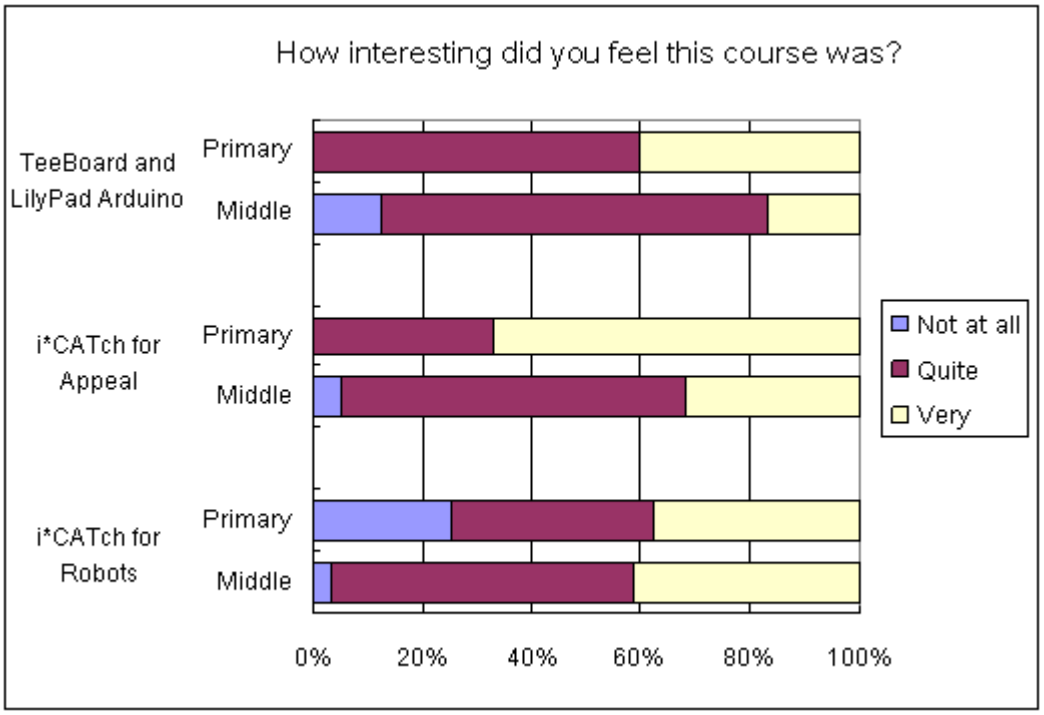


Figure 4.13. Summary of the results of the student's feedbacks from three kinds of workshops showing the student's interest on the course

4.3 Discussion

Overall three kinds of computational toolkits for crafts making had positive feedback from the students, and all of the groups successfully achieved the course objectives to create an interactive garment or complete challenging tasks. In fact, there were some elements of the computational toolkits for learning and creation that were different from each other. This section analyzes the computational toolkits from four different angles: construction interface, project theme, computer programs and engagement factors.

4.3.1 Construction Interface

Construction interface is the tangible interface of the computational craft toolkit that provides a space for user to make their interactive crafts. This interface element is coupled with the creativity, course design and learning process. We considered whether the construction interface supports creativity in two directions: one is the creative projects and conceivable ideas; another one is iterative design on construction and aesthetics (Table 4.4). From the view of the project outcomes, it is clear that three types of computational construction interfaces supported students to produce their creative projects. However, from the view of conceivable ideas, the flexible construction interface of the i*CATch apparel platform had better support on creative design than that of the TeeBoard/LilyPad platform significantly. Despite no measurement of the creativity space in the TeeBoard workshop, we could imagine if we applied the students' ideas from the i*CATch apparel platform to the TeeBoard/LilyPad platform, the idea of the final product could be greatly different from the initial one. For example, a hi-tech hiking jacket created by a group of three primary school girls from i*CATch apparel workshop, which consisted of four functions by using nine modules including three multicolored LEDs, one buzzer, one vibration motor, one switch, one joystick, one IR sensor and one ultrasonic sensor. It is impossible to achieve their ideas using the TeeBoard/LilyPad platform which only has 6 analog and 13 digital pins. Regarding the i*CATch robotic platform, it is difficult to compare it to the TeeBoard/LilyPad or i*CATch apparel platform directly, because the nature of tasks is different: robot is goal-oriented, and e-fashion

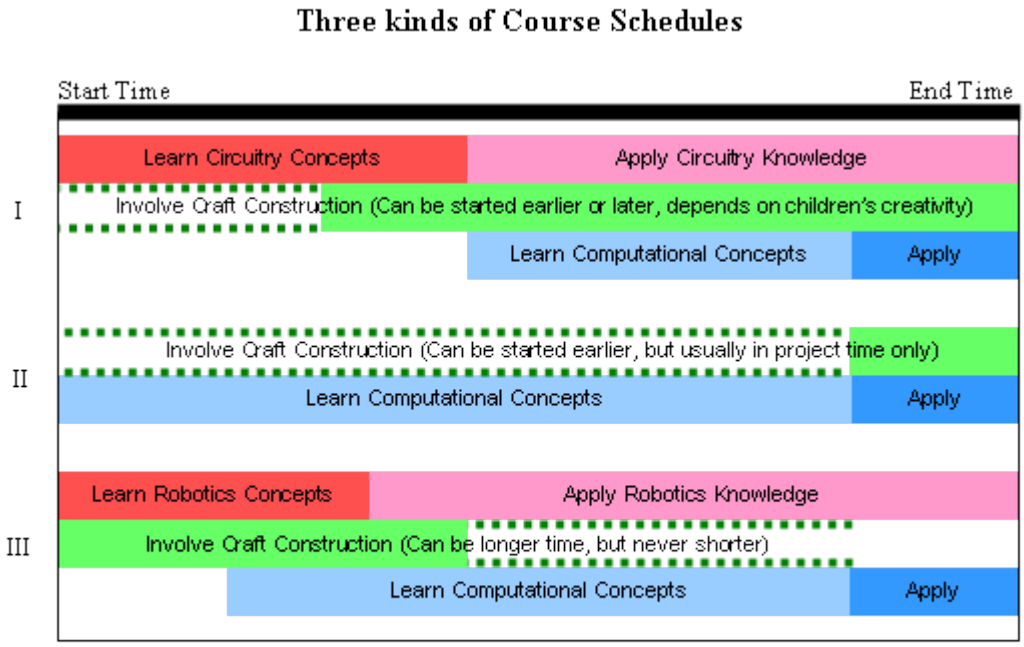
is design-oriented. If the i*CATch robotic platform is compared to the standard construction bricks of NXT platform, the design of the car structure of i*CATch is often more innovative than NXT, because of the unlimited shapes of the low-tech craft materials. From the perspective of the iterative design, the TeeBoard/LilyPad and i*CATch apparel platform support iterative construction design of electronic devices well, while the i*CATch robotic platform is not that flexible, due to electronic devices that are fastened by glue or tape on the carton paper. The surface or the structure of the robotic chassis may be damaged while reconstructing the position of the electronic devices. On the other hand, there was one unexpected usage of TeeBoard by the students, which could not be observed in other workshops. The primary school students, especially the girls, liked to use the colorful ribbon wires to make some patterns while completing their circuitry tasks. Figure 4.14 shows an example of the circuitry design by a group of girls aged 9-10. This observation shows that TeeBoard also supports iterative design on aesthetics without any extra craft materials.

<table>
<tr><th colspan="2">Computational Platforms for Crafts Making</th><th>TeeBoard with LilyPad</th><th>i*CATch for Apparel</th><th>i*CATch for Robots</th></tr>
<tr><td rowspan="2">Creative Projects</td><td>Initial Conceivable Ideas</td><td>Limited by point-to-point architecture, requires circuitry knowledge</td><td>Bus-based architecture supports complex ideas</td><td>Constrained by functionality and mechanics</td></tr>
<tr><td>Final Outcomes</td><td>Still like a prototype</td><td>More resembles final product</td><td>Can complete traditional robotic challenges</td></tr>
<tr><td rowspan="3">Iterative Design</td><td rowspan="2">Construction</td><td colspan="2">Plug-n-play facilitates iterative construction</td><td rowspan="2">Integrity of body parts limits iterative construction</td></tr>
<tr><td>Dependency issues may arise from modifications because of point-to-point architecture</td><td>Bus-based architecture enables modifications without affecting previously placed modules</td></tr>
<tr><td>Aesthetics</td><td>Functional elements (e.g. wires) part of decoration</td><td>Effects of functional modules are part of aesthetics, supports attachment decorative elements</td><td>Limited by mechanics</td></tr>
</table>

Table 4.4. A summary of the creativity space supported by three kinds of computational construction interfaces for making computational crafts.

Figure 4.14. A girl shows her circuitry design with a symmetric pattern for lighting up a series of LEDs at the back.

As three kinds of construction interfaces support different dimensions of creativity, it leads the course syllabus or schedule to have some variations. Figure 4.15 summarizes three possible course schedules of three computational construction platforms. The TeeBoard and i*CATch robotic platform requires students to start from learning electronic circuitry concepts and robotic concepts such as car construction and motor control respectively while the i*CATch apparel platform does not require students to learn any science concept before learning computational



I. The TeeBoard and LilyPad Arduino platform
II. The i*CATch apparel platform
III. The i*CATch robots platform

Figure 4.15. A summary of the three kinds of course schedules

concepts. The TeeBoard/LilyPad platform requires students to construct a simple circuit; or the i*CATch robotic platform requires students to build two basic wheels attached to two motors on the chassis before writing a program to control those modules; however, the i*CATch apparel platform does not require students to construct anything, except plugging the electronic modules on the jacket substrate. In other words, the i*CATch apparel platform allows have more time on programming practice during the course; or it is more flexible to adjust the course content to involve more craft design.

In the learning process, three kinds of construction interface bring different challenges to students. In TeeBoard workshop, although the students could use the colorful ribbon wires to create some creative shapes, they spent much time on debugging non-working circuits, many of which had typical problems such as open circuits and LEDs plugged in wrong directions, even at the final project session. In contrast, the students in the i*CATch apparel workshop did not encounter these problems, since the i*CATch apparel interface enforced correct attachment of the electronic modules. The students in the i*CATch robotic workshop also did not encounter these debugging problems, but they had to spend time to repair their cars sometimes such as wheel mounting. In addition, the bus-based architecture of the i*CATch apparel platform allows the electronic modules plugged in anywhere and programming would still remain the same, this encourages good programming practices of code reuse, divide-and-conquer coding strategies and modular design. From our observations, the students would work on developing the functionality of one or two actuators and sensors at a time, saving the program in a separate file, working on another couple of modules, and then finally integrating everything into their final product. In comparison, the students in the TeeBoard/LilyPad and i*CATch robotic platform tended to write a new program for the modules to solve the task every time. This reveals that programs for a bus-based architecture are inherently more portable and reusable than programs for point-to-point architectures.

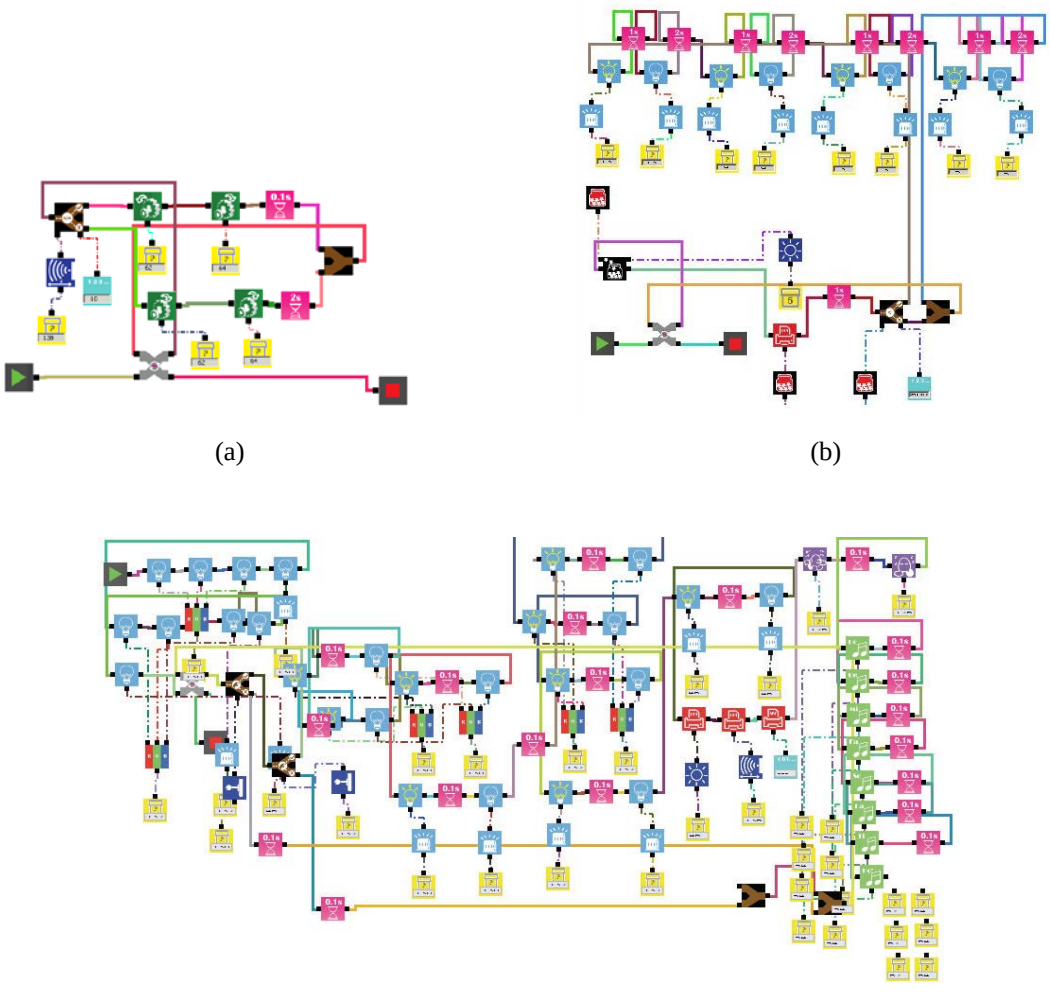
4.3.2 Project Theme

Obviously, the direction of the theme of the students' project in the workshop is highly related to the objective of the computational toolkits, such as constructing robotic car by robotic computational toolkit and creating interactive garments by wearable computational toolkit. But we focus on the discussion on possible themes of the projects or workshops that can be developed through a specific objective of computational toolkits. In the robotic workshops, the students seemed to limit themselves to design the physical appearance, movement and interaction of a robotic car. In contrast, in the wearable computing workshops using TeeBoard/LilyPad or i*CATch apparel platforms, the students had slightly a wider range of design on their garments. In aesthetic side, the students made light flashing pattern design (e.g. Figure 4.8 (a)), light embedded geometric-shaped or cartoon-pattern design (e.g. Figure 4.8 (b) and (c)) for their e-fashion projects. In functionality side, the students designed multi-functional garments for special usage, such as intelligent clothing for the blind (e.g. Figure 4.9 (a)) and hi-tech hiking jacket (e.g. Figure 4.9 (b)), etc. This illustrates that the project theme of wearable computing is more neutral and wider than robotics and inspires students to learn and design, which implies that it is able to improve the diversity of the student pool by introducing functionality or aesthetics design.

4.3.3 Complexity of Computer Programs

To understand how much computational knowledge the students learned through three tools, we analyzed the actual computer programs that the students created during the final project session. According to our observations, we did not find that the students had too many difficulties on using different kinds of programming environments. It is probable that we based on the student's programming experience to provide the relevant type of environments for them to use, so we will not consider the programming environment factor in the following discussion. Our hypothesis is the students will create longer and complex programs by using the i*CATch apparel platform, the next is TeeBoard/LilyPad, and the last is i*CATch robot. This is because the three computational platforms have different objectives of final projects,

nature of tasks and design of construction platforms. In robotic workshop, the aim of the final project was to program a robot to solve a number of complex tasks. In contrast, in the apparel workshop, the aim of the final project was to design and create an interactive garment. As expected, the nature of robotic tasks was goal-oriented, which required students to spend more time on solving the assigned challenges; and the nature of apparel tasks was design-oriented, which required more time on fashion and functionality design. In addition, the i*CATch robotic and TeeBoard/LilyPad platform are point-to-point architecture, which are less flexible to manipulate and reconstruct the modules than bus-based architecture.



(a) (b)

(c)

Figure 4.16. (a) A representative program by i*CATch robot students, (b) by the TeeBoard/Lilypad students and (c) by the i*CATch apparel students. The i*CATch programs for interactive garments tend to be longer and more elaborate, and use more functions and programming constructs.

Figure 4.16 shows three representative programs by students using three types of platforms as examples. The student programs show that our hypothesis is generally valid. From the view of program length, we measured it in terms of the number of icons programmed in i*CATch IDE (program bricks or statements of other two kinds of IDE were converted into program icons.). On average, the programs written by the i*CATch apparel students had around 100 icons; the programs written by the TeeBoard/LilyPad students had around 50 icons; and the programs written by the i*CATch robot students had around 20 icons. There was a slight difference in the variance of the programs generated by the students who worked on the apparel tasks due to various design ideas for their garments, while there was no noticeable difference in the variance of the programs generated by these who worked on the same robotic tasks. However, the results are still significant and show that the overall programs written by the i*CATch apparel students were relatively complex either in primary or middle school workshops.

In terms of program complexity, we measure it in terms of the number of logical constructs (i.e. conditional and looping constructs). On average, the programs written by the i*CATch apparel students had around three to four logical constructs in total, while the programs written by the TeeBoard/LilyPad and i*CATch robot students had around two to three logical constructs in total. The reason is because point-to-point construction on the TeeBoard/Lilypad and i*CATch robot platform involves more dependencies than bus-based construction on the i*CATch apparel platform. In the TeeBoard/Lilypad platform, students had to construct complex circuits if they wanted to write a complex programs, and in the i*CATch robotic platform, students were limited by the six sockets interface box and the fact of the assigned tasks, while in the i*CATch apparel platform, students were free to construct the modules, and thus they focused on program design more.

4.3.4 Engagement Factors

To investigate the engagement of the students to the computational craft activity, we analyzed our observations in our workshops from three angles: gender, learner styles, and types of programming environments. In our robotic or apparel workshops, the

majority of our students were male, and more than half of the students, especially the boys, had experience on robotic NXT construction toolkit before. Although more girls joined our apparel workshops than our robotic workshops, the number of enrollment was still lower than our expectations. The reason was probably related to our promotion of our apparel workshops that emphasized wearable computing and intelligent clothing, instead of other less technical themes such as fashion design.

Based on our observations on all workshops, some students found it easier to work with other students than others. It seems that is related to student's learning style: active (enjoy working in groups) or reflective (prefer working alone or with on or two familiar partners). To have more detailed observations, we found that more students had problems working with other students in the robotic workshops, maybe because the construction interface of the i*CATch robotic platform gives less support for collaborative learning than the TeeBoard/LilyPad and i*CATch apparel platform. Obviously, the physical construction area of i*CATch robotic platform is much smaller than the other two wearable computing platforms, which makes it difficult for more than two students work together. On the other hand, in performing assigned tasks, we observed that some students were eager to solve the programming tasks and some were more interested in the craft work. It seems that this working behavior is also related to students' learning style in sensing (prefer practical work and problem-solving) and intuitive (enjoy innovative work and design). Thus, these two kinds of students worked together in our workshops, their projects were usually better than the same styles of students. We also found that they had satisfactions on their work and could learn from their teammates' merits. Therefore, the wider domain of tasks attracts more kinds of learning styles of students to learn technology.

Most students enrolled into the workshop were interested in science and technology. Even though the students had different levels of programming experience, no students reported that they had big difficulties to use our text-based or graphical-based programming environments. Thus, we believe if the design of programming environment is user-friendly, programming environment will not be a factor to reduce the student's interest to learn computational concepts.

Overall, the engagement factors of students learning computational concepts through computational construction toolkits are associated with the intrinsic factors: workshop theme, construction interface and nature of tasks, and the extrinsic factor: learner characteristics.

4.4 Summary

In this section, we presented our investigation on how children learn through three types of the computational toolkits for crafts making: *Lilypad Arduino with TeeBoard platform, i*CATch apparel platform* and *i*CATch robotic platform*. Our aim was to have a better understanding of the children's learning experience of using computational toolkits to create computational crafts under different kinds of construction interfaces and assigned tasks or projects. According to our studies, the overall results confirm in the evaluative surveys and feedbacks from students and instructors, which shows that computational toolkits for craft activity attracts both boys and girls, and therefore they have the potential to broaden the population in learning technology and computing; computational toolkits for crafts making can provide a space for children to exercise their creativity and practice their programming skills. In conclusion, different construction interfaces of computational craft toolkits support different levels of creativity; wearable computing platform widens the themes of project, and thus broadens the diversity of students to learn computational concepts; the flexible construction interfaces encourage students to write complex programs; and the engagement factors are associated with workshop theme, construction interface, nature of tasks and learner characteristics.

Chapter 5 Expression through Story Creation

Children like playing bricks and making crafts and also like telling stories. It is common for children to receive stories through parents, friends and electronic media. Meanwhile, children also tell stories for communication and self-expression. According to Vygotsky, it is important for children learn through imaginative play and social interactions [128, 129]. Story creation or storytelling is a way to mediate the construction of meaning and a child's organization of knowledge, and also is a social activity that supports positive interaction among students and teachers [16]. Storytelling has been successfully applied as a mode of communication in digital and tangible media, which facilitates children in expressing their thoughts and ideas relating to surrounding objects [31, 35, 51, 126] and becomes a vehicle for children to learn languages and even programming concepts [2, 42, 45, 91, 108, 111, 118].

Currently, technology, especially in wearable computing and computational textiles, focuses not only on the improvement of existing functions, but also on style and aesthetics [80]. It has been shown that advanced e-textiles and computational technologies can positively impact the way children learn and create [18], and there has also been work on methods to "lower the bar" on user design interface [94, 95], which allows high-tech fashion to be usable not only by professionals, but by children. These developments lead to formation several e-textiles and wearable computing workshops [19, 20, 70, 77, 94, 95] for children to learn about technology and programming while allowing them room to express their personal style via apparel. In particular, apparel is a kind of personal object, and wearable computing supports to create some sorts of special effects. Some researchers have already taken the role of wearable computing in performance [116, 117, 121], but the systems developed are only based on the performer's body movements or gestures to display the images on the body or project the images on a screen, but which are not available for children to program some effects for performance on their own.

To take advantage of storytelling with the personal nature of computational apparel created through the "lower the bar" computational apparel construction platform, we proposed to combine these two modes of expressions to be a new form

of expression media (i.e. storytelling augmented computational apparel) for storytelling and for children to learn computational concepts. This chapter aims to explore the potential of our proposed expression medium – storytelling augmented with computational apparel – supporting children to create stories in some certain ways, and examine whether this expression mode of storytelling media has potential to be a new programming domain for children to learn computational concept. There are three basic aspects for analysis:

- Electronic devices on computational apparel representations: How do children associate electronic devices with their ideas to create stories?
- Roles of computational apparel in performing a story: How do children use the computational apparel to perform the stories?
- Computational learning: How do children learn the computational concepts through this new form of storytelling?

The chapter is organized into five sections. Section 5.1 reviews the forms of storytelling and the development of storytelling media. Section 5.2 describes the research methodology, including the introduction of wearable computational toolkit and two syllabi of wearable computing workshops for children. The findings of the stories created by the children are presented in section 5.3, and the discussion in section 5.4 examines the various ways in which children deployed the technologies and the programs in their stories and computational apparel to support children to learn computational concepts. A summary is given in section 5.5.

5.1 Storytelling and Storytelling Media

The art of storytelling has been around as long as there has been civilization. It was originally a method of passing knowledge from one generation to the next; it can also be a powerful tool for communication, collaboration, creating imagery, expressing emotions, and understanding of events through the interaction between storytellers and audiences [52, 78, 91]. Stories can be expressed in different forms of media: oral interpretation combined with gestures and expressions; visual form as a graph or movie; textual forms as a poem or novel [10, 78]. In all forms of storytelling, storytellers employ approaches or tools to enrich the plot: traditional

and adaptive pantomime, character imagery, draw talk, puppetry, felt board, chant, balloon, musical and group role play [10] are examples.

The rapid development of new technologies, such as virtual reality, ubiquitous computing and tangible user interfaces, offers storytellers a richer variety of tools to use. In recent years, storytelling has been successfully enriched with these technologies, such as animation creation using graphical programming environments: Alice [2] and Scratch [111]; electronic puppets: SAGE [126], Rosebud [51] and PET [35]; and physical or tangible interactive platforms: StoryMat [109], StoryRooms [91], StoryToy [45] and PageCraft [17]. The objectives of telling stories through these interactive tools are to engage children in technology learning while supporting the development of their cognitive and language skills, as well as to assist them in expressing their design ideas.

Graphical programming environments have been shown to be able support storytelling and computational concept learning by children [73]. However, these environments do not usually provide the same level of support for social interactions compared by physical or tangible environments. The few tangible or physical programming environments involve simple computational concepts and most of them have some defects. For example, StoryRooms' switching modes confused children to learn [91] and StoryToy's non-linear content seemed to be too hard to younger children [45]. Researchers have already researched the role of wearable computing in performance [116, 117, 121]. On the other hand, current "lower the bar" wearable technology and applications allow children to learn technology and programming with positive results [20, 94, 95]. These reasons motivate us to explore computational apparel that may be potential tangible interactive media for children's storytelling and learning of computational concepts.

5.2 Methodology

To explore the proposed medium: storytelling augmented with computational apparel to be a potential expression medium and to learn computational concepts, two syllabi of the workshops were designed to conduct a comparative study. Observations were conducted in six wearable computing workshops over the past

two years: two were targeted at primary school students and four for middle school students. The data gathered was supplemented through interviews, observations and video and photographic documentation.

5.2.1 Two Syllabi of Workshops

The objectives of two syllabi were the same to introduce young students to computational concepts via the proposed innovative media. Both workshops introduced students to basic programming concepts such as sequentiality, repetition and conditionals, and provided electronic devices through a series of small exercises such as getting the lights to flash in particular order or patterns, or emitting sounds in response to a sensor trigger. Towards the end of the workshop, the students were asked to create a project, which involved telling a story on the provided computational apparel with the given electronic devices and craft materials.

The main differences between the two syllabi were the extra time for introducing storytelling techniques (e.g. storyboard, image theatre and pantomime [82, 130]) and the requirement for the forms of storytelling (drama or pantomime) used in the project. The syllabus with the extra introduction of storytelling techniques and project requirements (or it was labeled as syllabus II) also contained some exercises with image theater technique (holding a static pose to form an image) and electronic effects to express emotion, such as holding a crying pose with the moving blue LED lights as tears to express the sad emotion (see Figure 5.1). Two syllabi were designed in owing to compare the usage of computational apparel in expression and storytelling by children and to explore the differences in children's learning of computational concepts with or without extra storytelling information and practices. Our hypothesis is that children are able to program computational apparel to express a certain level of a story, whereas they can create a more complex and substantial story with more practice on storytelling skills. There is room for improvement in expression through technology and computational knowledge through learning.

Table 5.1 summarizes two syllabi of the workshops (see the details in Appendix B). The same syllabus was used for three times for different subject groups: one was

organized for primary school children and two for middle school children. In order to take the younger children into account, the length of the workshops for primary school was usually shorter than that of the secondary school ones, but the contents of the syllabus were roughly the same. The syllabus I primary school workshop was run for 4 half days (around 12 hours in total), and two middle school workshops was run for 4 full days (around 24 hours in total). The syllabus II primary school workshop was run for 5 half days (around 15 hours in total), and two middle school workshops was run for 5 full days (around 30 hours in total).

Figure 5.1. A primary school girl holds the crying pose with the moving blue LED lights as tears to express the sad emotion.

		Syllabus I	**Syllabus II**
Learning Contents		**Example Tasks**	
Storytelling techniques		Not included, assumed to be prior knowledge	Forms of storytelling (e.g. pantomime and image theatre), story elements, storyboarding
Electronic devices		Output devices (e.g. LEDs, buzzers, etc.) Input devices (e.g. Light sensors, switches, etc.)	
Computational Concepts	**Sequential logic**	Blink a multicolored LED in rainbow color sequence; play a melody on buzzer, etc.	
	Conditionals	Turn on a LED or play a tone by a switch, etc.	
	Repetitions	Set LEDs to flash repeatedly in reciprocal pattern on the clothing, etc.	
Project		Various forms of storytelling	Drama / Pantomime

Table 5.1. The two syllabi of the workshops (see the details in Appendix B)

Participants

All workshops were held in a school environment. In the syllabus I workshops, there were nine girls (aged 9 -12) from a primary school, and there were 40 children (aged 12- 15) from various middle schools, who were assigned to two workshops, making 21 children (4 girls and 17 boys) and 19 children (4 girls and 15 boys) in total respectively. In the syllabus II workshops, there were 34 children (19 girls and 15 boys) from various primary schools, and 49 children from various middle schools, who were assigned to two workshops, resulting in 17 children (9 girls and 8 boys) and 32 children (12 girls and 20 boys) in total respectively. In all of the workshops, the children were asked to work in a group of three to four. As they were allowed to form groups on their own, most of the groups were of single gender, with only a few mixed groups. An informal survey of the students showed that a few of them had experience in programming and/or storytelling and performance. All workshops were taught by undergraduate computer science students. The ratio of instructors to students was about 1:6.

5.2.2 Wearable Computing Tools and Materials

The materials used in the workshops included both high-tech devices, such as wearable computing components, and low-tech materials, such as paper, felt and tape. The i*CATch wearable computing system [88] was deployed in the study. The i*CATch system consists of a construction platform, which is usually a garment, and a set of components: a main control board, and peripheral devices such as sensors and actuators. It has a plug-and-play construction interface, which supports iterative or trial-and-error design and development. Table 5.2 shows the materials provided in the wearable computing workshops. The high-tech devices were provided slightly differently in two syllabi of the workshops because of the improvement of the switch controls and the process of creating sound effects in the syllabus II workshops. According to the findings on the syllabus I workshops, the students always relied on numbers of switches on different positions to trigger electronic effects, and spent too much time on programming sound effects, and therefore we made a few changes outlined in Table 5.2. The electronic devices included actuators that were specially

engineered to be attention grabbing, such as extra-bright LED lights and extra-loud speakers. The construction platform was a long-sleeved jacket with pockets (and a hood appeared in syllabus II workshops only), with connection sockets running down the front and back of the jacket, down the sleeves (and on the top and front of the hood). For more details on the process of building the garment construction platform, refers to Section 3.3.1 Technical Setup in Chapter 3). The students were also supplied with low-tech materials such as felt scraps and paper that they could use to decorate their jackets or make some simple props. The i*CATch IDE was chosen to be a programming environment for children to write programs, which allows graphical drag-and-drop programming with source code generation (To review the details of the i*CATch IDE, refers to Section 4.1.2 Programming Environments in Chapter 4).

	Syllabus I	**Syllabus II**
Low-tech materials	Felt scraps, paper and tapes	
High-tech devices	Microcontroller LEDs Buzzers Vibration Motor Switches Light Sensors Ultrasonic Sensor	Microcontroller LEDs Buzzers (with sound library) Vibration Motor Switches Joystick Light Sensors

Table 5.2. Materials used in the workshops

A Sound Library for Expression

The i*CATch buzzer is capable of emitting sounds specified by frequency and duration. However, based on our observations in the syllabus I workshops requiring students to program in their sound effects note by note would be very tedious, not a good use of their time, and would lead to spaghetti code. To come up with this problem, a sound library with 11 sound functions was designed (Table 5.3). These sounds were encapsulated into functions and could be used by the children as an atomic unit. The children could also choose to emit single tones by specifying the note (pitch), octave and beat (duration).

The design of the sound effects was inspired by some previous research from fields such as robotics. It has been shown that even simple sounds, such as beeps or tones, can effectively convey a variety of meanings, such as notifications to the user,

the state of the machine, or mimic confidence levels [49, 75]. We therefore expect that if children are given the opportunity to effectively make use of technology, they may find it appealing to integrate sound effects into their own stories to express different events, objects or even signaling another cast member or synchronization between cast members.

Types of Sound	Effects in Sound Functions
Common sounds	The siren of an ambulance or police The beating of a heart A car alarm
Non-specific single-tone sounds	A long beep A short beep A rapid sequence of beeps
Non-specific multi-tone sequences	Four slow ascending tones spanning an octave (do-mi-so-do) Four descending tones in a minor key (la-so-fa-mi) Three rapidly ascending tones (do-mi-so) Three rapidly descending tones (so-mi-do) A rapidly descending tone sequence of 20 notes

Table 5.3. The 11 sound effects encapsulated into functions in the sound library

5.3 Findings

This section delves into the results of six user studies, highlighting the most interesting observations of the students' stories and performance. The findings include the use of electronic devices on computational apparel to represent some elements in their stories, the role of the computational apparel in the performance of their stories and children's computer programs.

5.3.1 Electronic Devices on Computational Apparel Representations

Based on the results in the syllabus I workshops, we expected that students would be more familiar with using electronic devices on computational apparel to express their ideas. This means that there may be more number of devices and representations in a story when they were introduced to story techniques and had more practice on their expressions through the electronic devices.

Table 5.4 shows the statistics on the device and representation in the stories in each workshop under the conditions by the storytelling techniques with more

introductions and practices (syllabus II) and without any detail on storytelling skills (syllabus I). The fluctuation of the average number of devices used between two syllabi does not consist with our expectation of more number of devices in a story strongly. Based on our observation, children could program each electronic device to represent zero or more than one idea in different moments of a story. In general, actuators such as LEDs and buzzers were mainly used to express various and interesting ideas. In contrast, sensors such as switches and joysticks were usually used for triggering events rather than representations. However, there are some exceptions, such as using a switch to represent a gun trigger and using an ultrasonic sensor to represent the process of the collision. Furthermore, children could program each representation using more than one electronic device to express, such as a flashing white-blue LED with a buzzer emitting the siren sound to represent delivering a patient to a hospital. Therefore, the number of devices was not a suitable indicator to reflect our expectation of familiarity with using electronic device for expression. On the other hand, the number of representations in the students' stories in each syllabus II workshops was slightly more than the number of the corresponding levels of the workshops (see Table 5.4), which supports our expectation of more number of representations in a story. However, there was one exceptional case in the primary-school syllabus II workshop. One all-boy group always did not focus on their works and played with each other all the time, so when we gave children an option to choose using devices or not in their stories, they chose the easiest way by not using any device to perform their stories. Except them, most students were interested in programming the electronic effects integrated into their stories.

Upon closer analysis on all children's stories, the representations using the deployed electronic devices could be categorized into four areas: to denote abstractions and actions; to denote concrete objects; to denote emotions; and to denote themselves in intelligent clothing, in which only *to denote emotion* category appeared in the syllabus II workshops, which was obviously related to the storytelling practice on the emotion topic (see Table 5.5). This result shows again that more practices facilitate students to use electronic devices to express ideas better.

Even though the concept of using electronic devices to convey ideas seem to be not so intuitive, most of the students could use the devices to represent abstractions, actions and concrete objects in their stories in both workshops. Therefore, general students enable to associate the properties of electronic devices with some story events or objects, such as death and fire. A more descriptive analysis of each category of representations is presented below.

	Workshops	Number of Devices			Number of Representations		
		Avg	Min	Max	Avg	Min	Max
Syllabus I	Primary (N=3)	6.30	5	8	1.67	1	2
	Middle (N=13)	6.31	4	10	2.00	1	5
Syllabus II	Primary (N=8)	4.25	0	8	1.75	0	4
	Middle (N=13)	6.00	4	9	3.31	3	5

** N = Number of groups*

Table 5.4. Statistics on the device and representation in the students' stories with free forms of storytelling (syllabus I) and limited forms on drama or pantomime (syllabus II)

Representations	Syllabus I			Syllabus II		
	Primary (N=3)	Middle (N=13)	Overall (N=16)	Primary (N=8)	Middle (N=13)	Total (N=21)
Abstractions and Actions	1 (33%)	11 (85%)	13 (82%)	2 (25%)	11 (85%)	13 (62%)
Concrete Objects	1 (33%)	7 (54%)	8 (50%)	2 (25%)	10 (76%)	12 (57%)
Emotions	0 (0%)	0 (0%)	0 (0%)	3 (38%)	7 (53%)	10 (48%)
Themselves	1 (33%)	2 (23%)	3 (19%)	3 (38%)	1 (8%)	4 (19%)

** N = Number of groups*

Table 5.5. The total number and the percentage of groups in each category of the representations of electronic devices

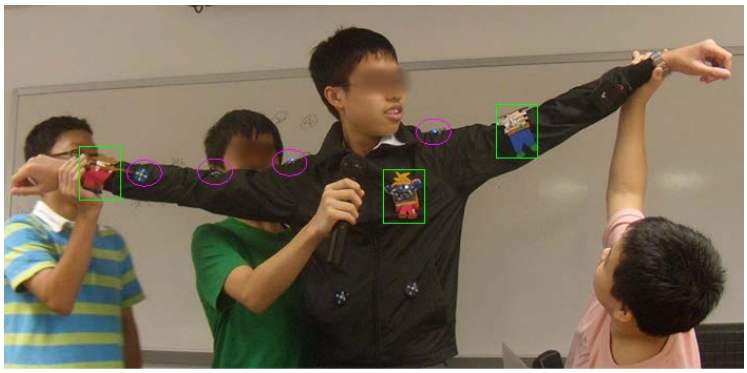


Figure 5.2. A story about filming: a director (center) and two gunmen (left and right) to make a movie (squared in green) and the trajectory of flying bullet (circled in red)

To Denote Abstractions and Actions

The most common usage of the electronic devices was to represent abstract concepts or events in both the syllabus I and II workshops, such as explosions, stock market fluctuations, or illnesses. There was an impressive example in the syllabus I

workshop. Four middle-school boys illustrated a story about filming by creating a director (microcontroller) who directs a movie about two gunmen, with using felt scraps embedded with a switch (see Figure 5.2, squared in green). A sequence of blue LEDs was attached along the top of the arms of the jacket. As the trigger is pressed, the blue LEDs turn on one by one to represent the trajectory of the flying bullet (see Figure 5.2, circled in red). Another example is the story *No Pain No Gain*, performed by three middle-school boys in the syllabus II workshop (the brackets indicate how the electronic devices were integrated into the story):

> *"… Two farmers bargain the price with the salesman. One of the farmers suddenly has an idea (Yellow LED on the head is on) to make the salesman reduce the price, and the bargain is deal finally…"*

They used a flashed LED placed on the head to indicate that a farmer had an idea in their story (see Figure 5.3).

Figure 5.3. Turning on an LED positioned on the top of the boy's head (circled) to indicate coming up with an idea.

The use of the LEDs to illustrate the trajectory of the flying bullet or reinforce the concept of a happy ending was unique to these two particular groups. However, the use of electronic devices to represent events such as explosion, stock market rallies and crashes, and major life events (such as injury and death) was a fairly common theme, as shown in the following story *Looking for a Planet to Live* (see Figure 5.4), told by a group of middle-school boys in the syllabus I workshop:

> *"Originally, all planets were beautiful and healthy (LEDs light up). However, man messed up Earth (LED embedded in the Earth felt turns off)…In the end, all the planets explode (vibration motor turns on and buzzer issues a "BOOM" sound)".*

Figure 5.4. The *Looking for a Planet to Live* story shows an example of using vibration motor to express one of the common story events – explosion.

Another example is *The Three Sons* story, performed by a group of middle-school boys in the syllabus II workshop (Figure 5.5):

> *"… three sons' father passed away (siren sound), ...The eldest son invested the money in the stock market, making a lot of money at first (slow ascending tone sequence). However, the market suddenly crashed (rapidly descending 20-tone sequence) and he lost all his money…The third son presses a button on the smart jacket (long beep), and their father's virtual image appears to tell them his wish..."*

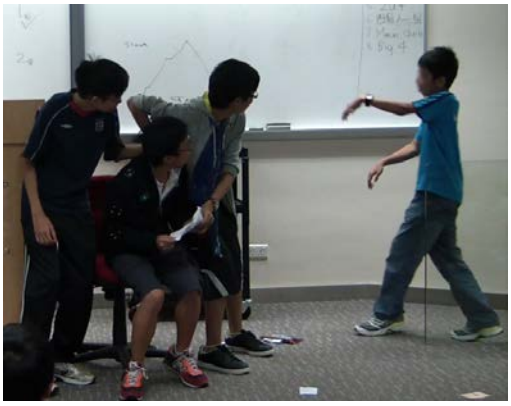


Figure 5.5. A scene of *The Three Sons* story: three sons (left) are watching their father's virtual image (right) projected from the smart jacket worn on the youngest son (sitting in the middle) and listening to his wish.

Closely related to abstractions are the representations of actions. For example, the fluctuations of the stock market can arguably be considered actions, as can be the event of the father dying in the previous story. In another example, there were many abstractions (e.g. injury and heaven) and actions (e.g. bleeding and delivering to a hospital) in *Siu Ming has an Accident* story, narrated by a group of middle-school boys in syllabus workshop I:

"Siu Ming goes into the forest and is listening to mp3 (a song is played from the buzzer). He doesn't notice a tree in front of him and walks into it (walks towards a desk in front of him, which is detected by the ultrasonic sensor), and he is injured and bleeding (the ultrasonic sensor triggers the turning on a series of red LEDs). An ambulance arrives, and he is sent to hospital (buzzer plays the siren of an ambulance). However, it is too late, he sees a flash of bright light (all white LEDs on the jacket turn on), and God receives him to heaven."

One more example about beating action is in the *Bullying* story, told by a mixed group of two middle-school girls and a boy in the syllabus II workshop (Figure 5.6):

"…A naughty boy played a nasty trick on a girl classmate by pulling out a chair from under her…The girls decide to take revenge on the boy and beat him up (violet LEDs flash, car alarm sound plays)…"

Figure 5.6. A scene of two girls bullying a boy: two violet LEDs on the front of the boy's jacket lights up and the buzzer plays the car alarm sound.

Many syllabus II workshop students relied on the provided sound effects, and few of them conducted a simple sound to represent an action. For example, a group of middle-school boys used a slow series of a low tone to represent the sound of plowing in their *No Pain No Gain* story:

"… Two farmers drive back to their farm and plow their fields (a slow series of a low tone)..."

To denote abstractions and actions was the most popular usage of the electronic devices both in two kinds of syllabi workshops, with 13 of the 16 groups (82%) in the syllabus I and 13 of the 21 groups (62%) in the syllabus II choosing to deploy the devices in this fashion (see Table 5.4). The middle school children were more able to use electronics devices to represent abstractions than the primary school students (see Table 5.5), owing to the stage of cognitive development. The primary school children 8-12 aged have not developed their abstract thinking ability well.

Many of the children used the sound effects from what they had previously seen from cartoons, movies or computer games (for example, using the siren to indicate the delivery to a hospital, using a descending sequence of tones to indicate a stock market crash, or using the car alarm tone, which was a rapid alternating sequence, to indicate a fight with fists going back and forth).

In this category, there was an even use of LEDs in the syllabus I workshops while there was comparatively less use of the LED lights in the syllabus II workshops. One reason was the students in syllabus I workshop, who worked without sound library, had some difficulties to convert a series of tones to express their ideas, and therefore some gave up using the sound. Therefore, the usage of buzzers in the syllabus I workshops was less than that of the syllabus II workshops. In other words, the sound library enriched the children to express some abstract ideas and actions. Even though the usage of LEDs was relatively less in the syllabus II workshops, there were quite a number of creative usages that were found in several groups using LEDs. For example, a flashed LED placed on the head to indicate that the wearer had an idea; a LED light flashing in colors of the rainbow is used to reinforce the concept of a happy ending in *Adapt to Change*.

The other types of electronic devices such as vibration motors and sensors were seldom used; only four stories used vibration motors to indicate explosions, earthquakes and fart, and only one story used ultrasonic sensor to indicate the process of collision in the syllabus I workshops. The main reason is probably related to the nature of vibration motors having less various output and unobvious effects in performance, and sensors having no output effect, not intuitive to be used for representations. In other words, using sensors to denote some meanings seems to require children to have a more creative mind.

All these observations show that actuators, especially buzzers and LEDs, were intuitive to students to express abstractions and actions in many forms of storytelling, regardless of having extra storytelling skills practice or not.

To Denote Concrete Objects

Another common usage for the electronic devices was to represent concrete objects.

In the syllabus I workshops, there were quite a number of groups that used the microcontroller to represent characters in their stories. This usage was only in the syllabus I workshops, since the children were free to choose their forms of storytelling. There were a numbers of groups that used puppetry to perform their stories, and therefore the children decorated the robot-faced microcontroller into their characters. Figure 5.7 shows a story with a movie director represented by a microcontroller as an example. In this story, besides a director, there were two gunmen created from felt scraps embedded with a switch. When the switch is pressed, the blue LEDs are turned on one by one along the arms (see Figure 5.2), which is similar to pressing a gun trigger, and the blue bullet is fired from the gun. In our observation, it is common for student to use switches to represent a button of a real object, such as a gun trigger, a play button of MP3 player and a button of a panel to order food (see Figure 5.8), as switches are common in our daily appliances.

Figure 5.7. A microcontroller (center) dressed up as a movie director, alongside two gunmen actors.

Figure 5.8. A girl presses a switch to trigger a light to indicate the food order.

In the syllabus II workshops, it was more common to use the LED lights, which could easily be programmed with different colors to represent different objects. An example is the cautionary tale *Don't do Drugs* (Figure 5.9), performed by a group of middle-school girls:

> *"One night, a girl got drunk at a disco (multicolored LED flashes with different colors, speakers emit the short ascending and descending sequences in succession) and*

somebody slipped her some drugs. Leaving the disco on a drug high, she crosses the road (multicolored LED flashes red, yellow and green, representing traffic lights). She passes by a hawker selling scarves, starts to look through the scarves, and finally to take off with one of them. However, the police come (multicolored LED flashes white and blue, speaker emits siren sound), and they arrest the girl."

One of the girls in the group of performing the *Don't do Drugs* story wore the computational apparel and stood on one side to produce the light effects of a disco ball, the red-yellow-green color of a traffic light and the flashing the white and blue lights of a police car to represent the scenes in the disco, street and the final arrest scenes respectively. There was another primary-school girls group in the syllabus I workshop who also used three colors of lights to represent a traffic light in their story. Traffic light seems to be a common object in children's story.

Figure 5.9. A girl standing on the other side (left) controls the lights on the arm to change different colors and the buzzer to play different tones to represent the disco ball and music

In our observations on both syllabi workshops, representing actual objects was a fairly common mode of usage for the electronic devices, especially for the LED lights. Perhaps the students used lights as paint to simulate the color of objects. Another example in the syllabus II workshops is *The Exam*, illustrated by two boys and a girl from the primary school section:

"Three students are taking an exam which is invigilated by a very harsh examiner. One boy is so nervous (LED flashes red like a beating heart) that he starts sweating (another LED flashes blue like a drop of sweat). The boy see that the invigilator has fallen asleep, so he grabs his classmate's paper and starts copying from it (blue LED goes off). The invigilator wakes up and discovers that the boy is cheating, and decides to report this incident. The boy gets really nervous again (a blue LED starts flashing again)..."

8 out of 16 (50%) stories in the syllabus I workshops and 12 out of 21 (57%) stories in the syllabus II workshops incorporated the use of electronic devices to

represent concrete objects. More examples of concrete objects that were represented were: the sirens of ambulances or police cars, the headlights of cars, traffic lights, blood (red flashes), tears, sky (blue steady light), lightning (flashes of white light), or heartbeats (flashes of red lights, accompanied by sounds from buzzer). As the representations were for concrete objects, the majority of the effects were straightforward and easy to understand. For example, the siren of the ambulance or police car is easily recognized, and two white lights on opposing sides of the wearer's body also readily recall us to associate them with the headlamps of a car (Figure 5.10). Using red lights to represent the heart or blood, and blue lights to represent tears or sweat were also fairly common choices.

Figure 5.10. A scene of the No Pain No Gain story: Two farmers drive (left) a car to the store (right). Two LED lights on the front of the boy's jacket (left) are turn on to represent headlamps of a car.

To Express Emotions

One of the most interesting usages of the electronic devices was to represent emotions. As expected, this usage was only found in the syllabus II stories, because there was an extra practice on an emotion expression through a static pose with the aid of the electronic devices. This practice in the syllabus II workshops was added because was to examine the way children used electronic devices to represent ideas was not a random result in the syllabus I workshops, but was also reasonable for children to apply. If the use of electronic device was not intuitive to represent various meanings, the children would be biased to use it for expressing emotions only, not for other representations. Using emotion topic but not the other, basically it was based on our experience that this category was not found in the syllabus I stories, and it was also inspired by some designers and researchers used lights in apparel to express emotion [27].

As a result, the extra practice only assisted, but not biased the students in their final project, and around half of them used the electronic devices to represent emotions. Looked into the ratio of the representations in the primary school workshop (see Table 5.5), the number of groups using emotions in their stories is slightly more than using abstractions and concrete objects. Owing to the age of the primary school students and the shorter duration of the syllabus, their stories were noticeably with fewer number of electronic device representations than the secondary school's one, and it would be reasonable for the primary students to follow some ideas from the prior practices in their final stories. The following is an example of denoting emotion, the storyline of *A Thief*, presented by a group of middle-school boys (Figure 5.11):

> *"...Outside hides a thief, nervously looking in (heartbeat sound plays) the store. The thief conceals his real motivations from the shopkeeper and the customer, so he come in happily (three red-green-blue lights flash repeatedly). Suddenly, he takes action: he demands money, discovers a thousand bucks in the cash register, and beats up the storekeeper. The customer, in the meantime, has witnessed the whole scene and calls the police. The police arrive and give chase to the thief, and finally shoot him (red LEDs turn on)."*

Figure 5.11. A thief (right) looks happily and come into the store. Three multicolored lights flash repeatedly to indicate the happy emotion.

Nervousness seemed to be a common emotion in the children's stories, as were happiness, frustrations and sadness, as seen in the story *Jenny and the Cat*, played by 5 primary school girls:

> *"A little girl called Jenny is frustrated at school and very depressed about it (mournful four-tone descending sequence plays, blue LED turns on)...The next day, Jenny comes up with the correct answer to a question (four-tone ascending sequence plays). She is overjoyed (red LEDs turn on)..."*

10 of the 21 groups (48%) represented the electronic devices, only actuators, in this usage. The use of the devices was divided fairly evenly between the lights and the sounds. Some of the usages resembled (or perhaps were inspired from or inspired) the concrete and abstract usages previously described. For example, four groups used a blue light to express sadness, which is reminiscent of the use of blue light to represent the concept of tears or crying. Among the colors, red, white and blue were the most often used, with red most commonly used to represent "active" emotions, such as anger, disgust or happiness, and blue to represent the more "passive" emotions, such as worry, sadness and nervousness. The sound effects were relatively more varied than the light effects. The most common usage was to use the heartbeat sound to represent nervousness. Other usages included using ascending sequences to represent positive emotions, such as successes or happiness, and the descending sequences to represent negative emotions, such as anger or sadness.

To Denote as Themselves in Intelligent Clothing

Interestingly, there were very few usages where the wearable computing devices represented themselves in both syllabi stories. In other words, very few students used the concept of smart jacket in their stories. Even when the concept of a smart clothing appeared in the story, it tended to take a minor role, such as the inherited jacket in the story *The Three Sons*. Another two examples of the stories in which one jacket played a role was an intelligent blind stick with smart functions worn by a blind in the story *The Foolish Blind and The Smart Blind* performed by a group of middle-school girls in the syllabus I workshops; another jacket played a role was a cheerleading jacket with lights worn by a fan in *The Volleyball Match*, performed by a group of primary-school girls in the syllabus II workshop. This usage was found on 3 of the 16 groups (19%) in the syllabus I workshop and 4 of the 21 groups (19%) in the syllabus II workshop.

The Overall Usage of Electronic Devices for Representations

The actuators were widely used in the student's stories. Among the actuators, the LED lights were the most popular in the stories, with all but two of the groups in the

syllabus I and II workshops respectively choosing to deploy them in the storytelling process. Both the white and the multicolored LED lights were used, with different patterns of flashes or colors invoking meanings or special effects in the story.

Confirming our expectations, the sound library was also widely used by the students in the syllabus II workshops. The most commonly used functions were the ascending and descending tones; many of the students tended to associate descending tones with unhappy events and ascending tones with happy ones, which was consistent with observations in previous work [75]. Rapidly alternating tones and sounds invoking real-life objects (such as the ambulance siren and the heartbeat) were also popular. Perhaps due to the children's music background or the time-consuming action on composing a sound effect, very few syllabus II children explicitly composed a song note by note in their stories.

Compared with the other two types of actuators, the vibration motor was very rarely used in both syllabi workshops. As expected, the final project required telling or performing a story in front of an audience and the vibration motor is not seen or heard. It is reasonable that the students found it less effective for expression compared with the lights and the sounds.

In contrast to actuators, sensors and microcontroller were not used to represent any meaning in the syllabus II stories, but few cases in syllabus I stories such as switch as a button of an appliance and microcontroller as a main character in a puppetry show. This result indicates that the use of electronic device representations tends to be related to how children to perform their stories. There is more discussion in the following subsection 5.3.2.

5.3.2 Roles of Computational Apparel in Performing a Story

In all the workshops, the students were provided with a jacket as a substrate for their computational garment construction. In the wearable computing and storytelling workshops (syllabus I), they were free to use their own ways to tell or perform their stories, and their stories could be categorized into four forms of storytelling: pictorial, puppetry, drama and pantomime. Based on the children's performance of the syllabus I workshops, it shows that computational apparel could be more

relevant to theater-based performance and other forms of storytelling could be use other forms of computational textiles such as bags and quilts instead. Therefore, in the technical stage workshops (syllabus II), the children were assigned to perform their stories in forms of drama or pantomime. As a result, Table 5.6 shows that the syllabus II middle-students had a little wider range of usage on the computational apparel for performing a story overall. However, the result in the syllabus II primary-students is not significant to show the advantage of drama or pantomime format compared to that of the syllabus I primary-students. This may be related to the class size and the ratio of instructors and students. The following is analysis on the roles of computational apparel in terms of stage, stage effects, control panel, costumes and cast members, which is similar to some kinds of theatrical elements such as costumes, music and lighting.

	Syllabus I			**Syllabus II**		
Role of Computational Apparel	**Primary (N=3)**	**Middle (N=13)**	**Overall (N=16)**	**Primary (N=8)**	**Middle (N=13)**	**Total (N=21)**
Stage*	2 (67%)	10 (77%)	12 (75%)	N/A	N/A	N/A
Stage Effects	2 (67%)	11 (85%)	13 (81%)	5 (63%)	12 (92%)	17 (81%)
Control Panel	2 (67%)	8 (62%)	10 (63%)	3 (38%)	11 (85%)	14 (67%)
Costumes / Props	1 (33%)	2 (15%)	3 (19%)	2 (25%)	2 (15%)	4 (19%)
Cast Members**	0 (0%)	0 (0%)	0 (0%)	1 (13%)	0 (0%)	0 (0.05%)

* Stage role was only found in pictorial or puppetry forms of storytelling
** Cast members role was only found in drama or pantomime
*** N=Number of groups

Table 5.6. Summary of the students' stories and the role of the computational clothing modules

As a Stage for Pictorial and Puppetry

The pictorial form of storytelling presents the characters in a static tableau, which is used in picture books and gives both teller and listener a good idea of the setting of the story. In contrast, the puppetry form of storytelling presents a dynamic set of characters that appear and disappear from the field of view of the listener as they enter and exit from the storyline. The children used the garment construction platform as a presentation board or we called it a stage for puppets to tell these two forms of storytelling. The main difference of puppetry from pictorial is that the students detached the characters from the stage, moved to other locations, or even thrown away (which usually signified the character's death) during the telling of the

story. Figure 5.12 shows *The Little Match Girl and the Frog Prince*, an example of such a story, developed by a mixed group in the syllabus I middle-school workshop:

> *"…she suddenly saw something bright under the snow (white LEDs flash) -- it was a glass shoe! (Sticks felt cutout of glass shoe onto the jacket). She tried to put on the shoe. Right at that moment, the Frog Prince came by in his carriage (sticks felt cutout of prince onto the jacket) and saw this beautiful girl who had the shoe that he was looking for..."*

In our experiments, the pictorial and puppetry formats were the most common forms of storytelling, with a total of 12 of the 16 groups (75%) choosing to use the computational apparel construction platform as a stage to tell in these two formats. The students who chose to use the puppetry form tended to be related to how they learn to present a story from schools according to our informal survey in the syllabus I workshops.

Figure 5.12. The story of *The Little Match Girl*. The narrator sticks a felt cutout of a match onto the "stage" of the jacket.

As Stage Effects for Performing a Story

As discussed in subsection 5.3.1 elaboration, those representations were used as traditional stage effects in the all forms of storytelling performed by the children. The traditional stage effects including spot light, sound and fake blood capsules, etc. may serve doubly as an indicator of mood or characterization in the children's stories. In the pictorial form or puppetry, the students used sound effects to show on the stage (the computational jacket) and light effects to show on the felt puppets or objects to make them lively, or to indicate the key or active character, which is similar to the spot light function. For example, in *The Little Match Girl* story (Figure 5.12):

> *"…She felt very cold, so she lit a match to keep warm (sticks match onto jacket, red LEDs flash). It burned for a while, and then went out (removes match from jacket). She*

was cold, so she lit match after match (repeats the process). When she got to the last match, she thought she would die of the cold. However, she suddenly saw something bright under the snow (white LEDs flash) -- it was a glass shoe..."

In drama or pantomime, most students wore the computational jacket in front of the stage to produce the effects and some students put it at the backstage or one side to produce the effects. Three groups exchanged the computational jacket (without any effect of electronic device to show) to wear between their group mates to indicate the key character at that moment of the story. For example, in *The Birth of Paper* story, performed by a mixed group in the syllabus II primary-school workshop:

"...The tree (a boy wears a jacket with blue lights) is so worried and scared and hopes itself won't be chopped. But the tree is still chopped by a man and is sent to a paper making factory. A piece of paper is produced (a girl wears the jacket) and narrates her life..."

On the other hand, one all-boy and one all-girl groups in the pantomime performance also utilized these stage effects, using sound usually, for cueing other cast members, such as synchronizing a movement within the group in *The Three Sons* story after the siren sound (representing a scene of father's death) (see Figure 5.13).

Figure 5.13. The boys are moving after the siren sound cue in *The Three Sons* story.

The use of computational apparel to produce the stage effects was very common, over 80% of all groups in average. Particularly, the middle-school children used the stage effects more than the primary-school children. This result reflects that producing stage effects may not be that easy to primary-school students because it is not concrete knowledge and is related to children's experience on watching dramas or movies in their lives.

As a Control Panel

To facilitate to show the stage effects on a particular moment or event, children used a joystick, a switch or other sensors to trigger the effects instead using the timed delay function to display the effects. Therefore, sensors used as a control panel of a stage. For example, in the *Bullying* story, a middle-school boy used a joystick to change the lights to blue to indicate his sad feeling after bullying (Figure 5.14). There were 10 of the 16 groups (63%) in the syllabus I and 14 of the 21 groups (67%) in the syllabus II using sensors as a control panel to trigger effects in a particular event. More syllabus II children used sensors to trigger effects, in particular to joysticks, as perhaps the joysticks were available in the syllabus II workshops, which was predictable and intuitive of the device for this trigger events purpose. However, programming sensors to trigger event was seemed to be more difficult to the primary-school children, so fewer primary-school groups used sensors than middle-school groups.

Figure 5.14. A boy controls a joystick to change the lights to blue on the front of the boy's jacket to show his sad feeling.

As a Costume

It is rare for the children to use computational apparel as costume in their performance, being found in seven groups (19%) in all workshops. Only one primary-school girls group and one middle-school mixed group in the syllabus II workshops used some felt scraps to have extra decorations on the computational jacket, and the rest mainly programmed the LEDs to produce different flashing patterns to make the jacket beautiful or futurist, such as a fan's cheerleading jacket in *The Volleyball Match* (Figure 5.15 (a)), or they programmed the sensors to create a smart jacket as a prop for their dramas such as a smart blind's smart jacket in *The*

Foolish Blind and The Smart Blind (Figure 5.15 (b)). This result shows that most children (even girls) are more interested in programming the electronic devices to produce effects rather than designing a costume with craft materials.

(a) (b)

Figure 5.15. (a) A fan wears a cheerleading jacket with lights and star-shaped felt accessories. (b) A smart blind girl uses a light sensor (circled) on her smart jacket to check the banknotes.

As a Cast Member

There was an unexpected and interesting finding in the syllabus II primary-school workshop. The role of the computational apparel that have been described so far have mainly used as stage effects: to accentuate or to convey a point of information to the audience and as a control panel: to trigger events. One all-girl group used the computational jacket with a sound to be a cast member – magic mirror. Figure 5.16 illustrates their *The Girl and The Magic Mirror* performance: A girl stands on the left who wishes to be beautiful. Another girl stands on the right who holds up a jacket and controls the sound of the magic mirror. Every time when the girl asks the mirror if she was beautiful, the mirror answers her "No" with a mournful sequence plays. This interaction is similar to a communication of two cast members on the stage.

Figure 5.16. A scene of *The Girl and The Magic Mirror* story

5.3.3 Computer Programs

In addition to the usage of the computational apparel for performing a story, another thing was the potential for this media to facilitate students' computational learning. Given that most of the students did not have any experience in programming, this is a good indicator as to whether the use of new media such as storytelling augmented with computational apparel which can also be used effectively to teach computational concepts. To investigate this aspect, two hypotheses were established: I) the programs for the primary-school workshops were shorter and simpler than the middle-school workshops, because the age of primary-school students was smaller and the duration of the primary workshops was shorter; II) the programs for the syllabus I workshops were also shorter and simpler than the syllabus II workshops, because the extra practice on expression through electronic devices in the syllabus II workshops. Three methods of program measurement were used for analysis: the number of line of code, the complexity of program and the quality of program. Most of the results support the hypotheses.

Line of Code

The length of the program is measured by the number of lines of text codes generated by drag-and-drop icons in the i*CATch programming environment. The average number of line of code in the syllabus I primary-school workshop was 40.7 while the average number of line of code in the syllabus II primary-school workshop was 18.7 (see Table 5.8). This difference was statistically significant. It violates the hypothesis I because most syllabus I primary-school girls liked music so much and conducted their own music note by note; as a result, a lengthy program was found. In contrast, most syllabus II primary-school children relied on the provided sound library, and thus their program length was much shorter.

The average number of line of code in the syllabus I middle-school workshops was 25.8 while the average number of line of code in the syllabus II middle-school workshops was 38.0, which is more than that of the syllabus I middle-school workshops. This is because the number of line of code would be proportional to the number of electronic device representations in the story if students do not create

their own sound effect with a series of notes. A longer program is required for more number of representations. This result is consistent with the results of the number of representation discussed in section 5.3.1 and the hypotheses in this section. In other words, if children have more practicing on storytelling skills with electronic devices, students would be able to design a longer program.

Program complexity

To measure complexity, the number of complex operations (iterations and conditionals) in a program was added up. The average complexity of the syllabus I primary-school children's programs was 1.7, as same as the syllabus II primary-school children's programs (see Table 5.8). There is no significant difference between two syllabi in primary-school workshops. This result is reasonable as there was only three-hour learning time difference between two primary-school workshops. In fact, the iterative and conditional concepts are often abstract for younger children to learn. Most of the children in our workshops only programmed one conditional statement for controlling a switch or used an infinite loop construct to repeat the effects of actuators. In contrast, the average complexity of the syllabus I and II middle-school children's programs were 2.5 and 5.5 respectively. This result supports our hypothesis I and II that program complexity in the middle-school children's programs was more complex than the primary-school children's programs, and program complexity in the syllabus II middle-school children's programs were more complex than the others.

A closer analysis of the programs from middle-school children indicated that most of them could apply conditional constructs for triggering events with sensors and iterative constructs for repeating sound effects or flashes of light. In syllabus II middle-school workshops, most children were able to write a multilevel of conditional construct for triggering events with a joystick (that was only provided in syllabus II workshops) while this multilevel construct was not common in the syllabus I workshops, and the conditional constructs were inserted into different parts of sequential statements, when children used switches only. This children application shows that a five-directional joystick is convenient for children to trigger

stage effects during the performance and also facilitates children to learn conditional concepts. Furthermore, some children wrapped this multilevel of conditional constructs into an infinite loop construct. They realized that this approach was more efficient instead of restarting the program by resetting the i*CATch main control board (and waiting for the requisite 3-5 seconds as it resets itself). This finding also explains why the syllabus II middle-school children's programs are more complex than the syllabus I students' programs.

Program quality

To measure the program quality, the scoring system was designed (see Table 5.7), which was defined as the level of the linkage of a program and a story and the number of program errors. Programs with a score of 1 contained a weak linkage to the story, or there were many unnecessary program statements or logical errors. For example, in a story about a policeman fighting with four fighters (created by one all-boy group in the syllabus I primary-school workshop), the boys only programmed the light flashing pattern for the jacket to indicate the policeman from the other four fighters. This program was only related to a costume which took a minor role in the story, so the score of this program was given 1 point. Programs with a score of 2 contained a strong linkage to the story, or there were few unnecessary program statements or logical errors. For example, some extra codes were found in some children's program, which was not used for the story, because the students forgot to delete those codes after the device was plugged out from the jacket; the common logical error was found due to a lack of using timed delays after the codes for actuators, and thus the effect of the actuator could not work properly. Programs with a score of 3 contained a strong linkage to the story with no significant program error. The programs of the *Don't do Drugs* and *Siu Ming has an Accident* story are examples which contain numbers of program statements to produce effects to enrich the story plot without any program error.

The average score of program quality of the syllabus I and II primary-school children's programs were 1.3 and 1.7 respectively. In contrast, the average scores of program quality of the syllabus I and II middle-school children's program were 2.1

and 2.3 respectively. The score of program quality in the primary-school children's programs is lower than that of the middle-school children's programs, and the score of program quality in the syllabus I workshops is slightly lower than that of the syllabus II workshops; this result verifies the hypothesis I and II again.

Score	Score Explanation
1	The program has a weak linkage to the story, or there are many unnecessary program statements or logical errors.
2	The program has a strong linkage to the story, but there are few unnecessary program statements or logical errors.
3	The program has a strong linkage to the story with no significant program error.

Table 5.7. The scoring system for the program quality

Syllabus	Workshop	Project	Line of Code	Complexity	Program Quality
I	Primary	Various forms of storytelling	40.7	1.7	1.3
	Middle		25.8	2.5	2.1
II	Primary	Drama/ Pantomime	18.7	1.7	1.7
	Middle		38.0	5.5	2.3

Table 5.8. Summary of the average of three kinds of program measurements

5.4 Discussion

Overall, the findings show that children are able to create and perform a story through the suggested media storytelling augmented with computational apparel media; meanwhile, children can learn some computational concepts through this platform. In addition, children can create more interesting stories or performances with more practices on storytelling and programming skills, which confirms our expectations on the potential for this expression domain and storytelling media to facilitate children's learning of computational concepts.

5.4.5 Computational Apparel Media for Creating a Story

In our studies, most students could program electronic devices to link to or to illustrate key points in the stories and could use the computational apparel to perform their stories through different ways, particularly through drama or pantomime. However, we may still want to ask how the computational apparel and electronic devices critical to the stories, or whether the students were incidental and used only because we requested the children to do so. According to the instructors' observations, most children were motivated to interact with the computational

apparel platform, especially the children in the syllabus II workshops, who made their storyboards to design which scenes should add what effects or which scenes without effects, which is similar to stage effects design in a traditional drama. Only few children (usually boys) reported that they found it difficult to use the computational apparel and electronic devices to create a story. The main reason was that they thought a smart jacket could not be used for other purposes. As a result, most of these groups usually used the computational apparel as a costume with a very few linkage in their stories, or programmed one to two electronic devices to produce several simple effects for their stories. The children's story design process indicates that most stories did not come from incidental and most design was critical to the story.

Actually, the concepts of electronic effects from computational apparel are similar to the traditional stage effects. One 13-aged girl from the syllabus II workshops was an active member in the school drama club for at least 6 years. She reported that using the clothing with lights and sounds was similar to the lighting and sound production on the stage, and only the position was different -- the traditional lighting was from outside pointing to a stage or characters while the computational apparel method was that the lighting directly lighted up on the clothing. Compared to the traditional theatre methods such as lighting and sound design, computational apparel platforms have the advantage of supporting children to design stage effects quickly and flexibility by programmability. The light and sound effects from computational apparel enriched the visual appeal during performing a story, whereas particularly in a pantomime, those effects also provided more information for audiences to associate the effects with the cast members' body gestures, which helped to understand, also left enough space to imagine. For example, a scene in *The Three Sons* story: the eldest son's hands held some paper (with slow ascending tone sequence), and then he threw all the paper from hands (with rapidly descending 20-tone sequence). This scene with sound effects was clear to the audience to know that the eldest son lost all his money in the stock market. However, we may have gotten the impression that the eldest son was so happily throwing away all the paper if no sound was supplemented.

In addition, an unforeseen contribution of computational apparel media was that electronic devices helped the students to present their ideas more comfortably. Since they were given only three hours to create a story, to build and program their computational clothing creation and to rehearse their story (even they were given three hours more in the syllabus II middle-school workshops), it is understandable that many of them should be nervous about presenting in front of a crowd. However, the electronic devices helped to keep the audience's attention, and therefore reduced the children's nervousness when speaking or moving their bodies in front of the audience. A case in point is the group of students in the syllabus I middle-school workshops who created the movie director story. They were too shy to even speak up when asked a direct question during the workshop, but when they were asked to present their story, they were very eager to see the audience's reaction on the LED "bullet". Their feedback was very enjoyed the process of telling the story.

5.4.6 Task Characteristics

To program a story through a computational apparel platform is different from traditional robotic programming, even traditional storytelling programming. We found that some children had difficulties in writing a program for storytelling while some children were engaged in programming for performing stories easily. The following analysis focuses on three aspects to discuss.

Problem-Oriented versus Design-Oriented

Traditional computational tasks for children to learn are related to robotics, which are often to solve problems related to science and mathematics such as line-following and obstacle detection. This kind of tasks focuses on training children's problem understanding and using the divide-and-conquer approach to solve the problem. In contrast to robotics, storytelling tasks on computational apparel platform are often to design storyboards and story elements, which is similar to the tasks on graphical programming environment. In the syllabus I workshops, most of the children expected to learn computing and technology, as the theme was related to smart clothing. The gender ratio of females to males was 1 to 1.89, which was much

higher than that of the general robotic workshops (almost no girl there). By comparison, in the syllabus II workshops, the children clearly knew that the theme was related to theater and performance, and few children even unexpected to learn programming. As a result, the gender ratio of females to males was 1 to1.08, which was almost even. This statistics indicates that the nature of storytelling tasks attract children of both gender and with different learning styles.

Screen-based versus Theater-based

Storytelling tasks on computational apparel platforms are also related to creating stories, but its design focus is different from story creation in graphical programming environments. According to the findings, children programmed the stage effects to express the mood of characters or indicate some objects or events, which were tied in the voice with puppetry or body gestures. During the performance, the children had to remember when to trigger the effects or what effects would produce, and then what actions the students needed to do. In addition, each performance seems to trace a program once, which should facilitate children to understand more of programming concepts. Compared to graphical programming environments, children often programmed the movements or dialogues of virtual characters to create animations [73, 106]. If children only focus on the screen to design and program, they will miss the chance of learning through bodily engagement and interaction with tangible objects and peers, which is a good starting point to motivate children to gain knowledge about abstract concepts [55].

Products and Performances

In general, researchers mainly consider what kinds of project aspects (e.g. related to robotics, e-fashions or animations) can motivate children to learn abstract computational concepts or how to improve construction platforms to support collaborative learning and create their computational products effectively (e.g. Tern bricks [56] and Storytelling Alice [73]), but few researchers consider what will happen after the children’s projects come out. The robotic projects usually are used to solve the assigned challenges; the e-fashion projects usually are to express

children's design and creativity; the animations usually are to display children's ideas and stories. Most of the outcomes of these kinds of projects can be demonstrated by one child, and the rest of the group members may stand near to the presenter and observe the results. In contrast, performing a story through computational apparel often require more than one student: in puppetry, usually one child was a narrator and others moved around the puppets or triggered the effects on the stage or used their voices to act out the puppets; in drama or pantomime, most children conducted a role play in front of the audience and some were responsible for triggering the stage effects. In other words, the storytelling through computational apparel platforms support children to prepare their stories collaboratively (due to the fact that the i*CATch computational platform was suggested to use in the study, and its construction interface supports to work collaboratively mentioned in section 4.3.4 engagement factors in chapter 4), and even performing their outcomes of their stories. This result inspires us that collaboration design for learning tools is not only related to the construction user interfaces, but it is also related to the presentation of their project outcomes.

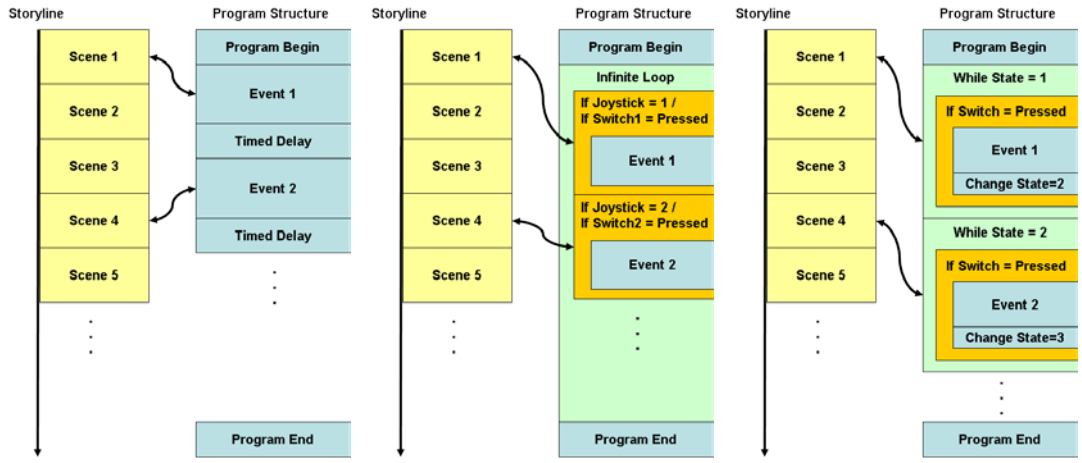
5.4.7 Support for Computational Learning

In all workshops, students learned some basic computational concepts such as sequentiality, iteration and conditionals and worked with the wearable electronic devices before they wrote a program to create a story. In the project session, we did not teach them to use a particular program structure to create their stories, but we encouraged them to utilize the sensors to generate the effects of actuators instead of the effects running all the time. According to the statistical findings in section 5.3.3 above, the program length, program complexity and program quality indicate that the children were able to apply the basic computational concepts to program some effects or representations for their stories.

Interestingly, one unexpected observation of the children's programs was that they basically deployed three types of program structure to link up their stories (see Figure 5.17): 1) Sequentiality on events and timed delays; 2) Infinite loop with joystick or switches as triggers; 3) Sequentiality on states with a switch trigger.

Sequentiality on events and timed delays is the simplest logic to link up to a story, as its structure is close to the sequential storyline. Thus, at the very beginning, most students used this structure. After a while, most students discovered the problem that they were required to predict the exact time of each scene to use this structure, which is difficult in practice. As a result, only one group of students used this program structure in the end. *Infinite loop with joystick or switches triggers* was used by most students, due to its flexibility in triggering events at a particular time. Time control is important to a performance. Without a lengthy rehearsal, the length of each scene is not clear and this flexibility becomes a critical issue that allows students to operate the effects or representations in particular scenes. It is also convenient to allow children to test their programs because they can test the outputs of each event individually without waiting to run through all the previous events and timed delays. *Sequentiality on states with a switch trigger* is the most complex logic to link up to a story, as this structure requires the use of an extra variable and while constructs to keep checking the state of each event. Therefore, this structure has the advantage of flexibility similar to the second type of program structure; while being close to the sequential structure of a story. Furthermore, while performing a story, the children did not need to remember which directions of the joystick or which switches trigger which events, since they only needed to press one switch to run each event in order. However, this complex structure was only deployed by one syllabus II middle-school all-boy group, in which the boys had programming experience on NXT. This is likely the reason why they could construct a complex and relevant program structure (see Appendix H).

While the storyline was not intended use of infinite loop or while loop with states, the first that the children could figure out how to construct their programs to be more flexible show that they do have a grasp of the computational concepts of sequentiality, iteration and conditionals. Again, this is additional evidence that shows that the storytelling argument with computational apparel has potential to be a new media for children to learn computational concepts.



(a) (b) (c)

Figure 5.17. Three kinds of program structures found in the workshops (a) Sequentiality on events and timed delays (b) Infinity loop with joystick or switches triggers (c) Sequentiality on states with a switch trigger

5.5 Summary

This chapter presents a study to reveal the potential of storytelling augmented with computational apparel media for children to perform stories and learn computational concepts. By comparing two syllabi of the workshops, the results give evidence that children were able to use wearable electronic devices and computational apparel for expression and performing stories, which was not an accidental, and there was room for children to improve their stories through more practices on storytelling and programming electronic device skills. In addition, the analysis of the children's computer programs shows that our proposed storytelling through computational apparel platform is possible to be a new tangible interactive storytelling method for children to learn computational concepts.

Chapter 6 Analysis of Design of Tangible Computational Construction Kits

The previous three chapters describe three case studies from the three major categories of tangible computational construction platforms. Each study indicates that the corresponding platforms have potential for assisting in children's computational concepts learning. These positive results lead to two questions: Are they all efficient in children's computational concepts learning at same level? Any characteristics of these kinds of platforms have an advantage over each other?

To address these questions, we suggest an analysis of the three major categories of tangible computational construction kits via five factors that were informed by our experience of the studies described in previous chapters: coupling of computational concepts, construction interfaces, domains of tasks, learners' characteristics and learning environments.

6.1 Coupling of computational concepts

In general, children find it difficult to learn abstract computational concepts. According to Piaget's theory of cognitive development, children at the concrete operational stage require concrete objects to understand abstract concepts [98]. Ullmer and Ishii have proposed that tangible user interfaces (TUIs) [125] bridge between the abstract digital world of computational concepts and the real world, in which tangibles can carry physical state, with their physical configurations tightly coupled to the digital state of the systems they represent. Some tangible programming systems are influenced by this concept, such as AlgoBlocks [120], Programming bricks [89] and Tern [56]. However, in our i*CATchBadges study, we found no significant difference in children's computer programs between programming in GUI or TUI environment within a 2-hour period of time. No significant differences were also found in Horn et al.'s study in science museum [58] that confirms our results. Actually, the nature of programming structures is one instruction (or one function) followed by one instruction, which is similar to brick

by brick, and therefore the design of tangible programming language usually uses a physical brick to represent an instruction or function to form the basic programming constructs (e.g. sequentiality, repetition and conditionals). However, computational concepts are dynamic fashions instead of static objects, and therefore it is difficult to have a strong coupling of passive physical bricks or other artifacts with computational concepts. Even though bricks can be embedded with electronics to generate dynamic simulations like i*CATchBadges, SystemBlocks [135] and Topobo [104], the scalability and the versatility of TUIs often support simple tasks only; it is still difficult for the design to apply to more complex tasks compared to GUI environments. The TUIs tend to encourage children's sensory engagement and collaborative interactions [58, 133] rather than facilitating children to assimilate the abstract concepts directly. This is discussed in section 6.2.

Concreteness does not only indicate objects that can be grasped with hands, but it can also refer to physical actions [29]. Building physical microworlds is a cognitive approach to link the abstract world and the real world [38]. Programming a Mindstorms robot [90] to have different movements and computing a bicycle's speed by Tangible Programming Bricks [89] are some of the representative examples. In our study, the i*CATch robots and apparel systems also support building physical microworlds. For example, programming a robot to move in a square shape can use a series of instructions in sequence or a repetition construct; turning on a light in the dark and producing a sound when too close to a wall can be solved by a conditional construct. In our observations, children were eager to count the number of movements and experiment the responses by triggering the light and ultrasonic sensors. Through these explorations, most children were able to link the tasks of building physical microworlds, which is similar to the researchers' studies in robots and e-textiles [19, 20, 89]. Furthermore, building physical worlds is flexible to form simple and sophisticated tasks for children to solve with computational concepts. Therefore, the building physical microworlds approach is effective to assist children in assimilating the concepts through exploration of the physical world.

However, it is difficult to motivate girls to learn computational concepts through building physical microworlds, and researchers therefore use storytelling that is a natural way for children to express ideas to bridge the abstract computational world such as the graphical programming environments Alice [2] and Scratch [111], and a physical programming environment StoryRooms [91]. Even though storytelling is a natural way of expression, it does not match well to computational constructs. Some researchers found that children used fewer loops in Storytelling Alice programs than generic Alice programs and commented that loop construct does not match well to storytelling [73]. In our experience in storytelling through computational apparel, children also used fewer loops compared to conditionals, even in situations where they wanted to repeat the effects of actuators (e.g. two-tone siren sound). They relied on sensor triggers in an infinite loop instead of using several loop constructs for each effect, which means that they have to hold down a switch or keep the joystick in one direction for a while to generate repeated effects (e.g. a series of siren sound). However, this is not to imply that the loop construct is not suitable for storytelling, since repetition of events or effects generally occurs in stage performance. If children could implement sensor-triggers embedded in a number of while-loops in sequence like the group of middle-school boys in the syllabus II workshop, three basic constructs would be used in a program to perform a story. Some hints are suggested to be given to children when designing their programs, which should be helpful for children to match computational constructs with storytelling. Therefore, we believe storytelling can be considered a potential method to assist children in understanding computational concepts.

6.2 Construction interfaces

The construction interface of a computational toolkit needs to consider inputs and outputs. Input interface is used for programming, which can be a mouse, keyboard or tangible platform. The design should prevent learners' frustration on program syntax errors and provide a scaffolding layer for transitioning to a more complex system (e.g. text-based). Output device is programmed by an input interface, which can be a robot, e-fashion and story, and may have a reconstruction interface for

learners to design their artifacts flexibly. In addition, there is evidence that working together aids learning [32, 129], and therefore collaborative construction interfaces are also preferable.

In tangible programming systems, the design of input construction interfaces are as important as the design of output interfaces, since its rationale is to let learners focus on their hands-on physical blocks to solve computational tasks, which means writing a program by manipulating a set of tangible objects while the output can be displayed on a separate device or on the same input bricks. Usually, the design is a one-to-one mapping between (functional) bricks and program instructions. As a program is often formed by a series of bricks (e.g. Tangible Programming Bricks and Tern), and learners do not worry about the syntax error problem. In addition, the space of tangible interface often supports collaborative construction [58, 133]. In our study involving the age 16 male subject who could not distinguish between the action flow and the data flow, the subject always made the connections in the wrong place until we drew two squares for action connections and two circles for data connections onto the action badges. This additional clarification indicates that the perceived affordance of the construction interface prevents making errors, and therefore children can focus on manipulating the computational concepts. On the other hand, our tangible programming simulation has a one to multi-semantics mapping between (functional) bricks and a number of representations or instructions (e.g. smart clothing's functions or story's characters) which expands the possible aspects of tasks. Even though the one-to-many mapping approach is more flexible and facilitates children's association ability, it still can be only used to solve simple tasks. The children also had the initiative to work collaboratively in the game booth activity or the classroom workshop through our tangible programming simulation.

In computational crafts construction platforms, the rationale is reversed, in which the programming part often uses in graphical or text-based programming environments (the input is not necessarily tangible) to control a physical construction artifact (the output is physically constructible). Unlike text-based programming environments, graphical programming environments often free learners from having to worry about the syntax, but it may be too simple for the

learners who have programming experience [26, 110]. To fill in the gap between graphical and textual programming environments, a hybrid graphical-text-based programming environment is suggested such as the Bricklayer and the i*CATch IDE in our study. This kind of interface supports a wider range of learners' experience in programming and a transition to text-based programming environment. However, it does not really support learners to collaborative work while the physical output of the computational crafts platforms may or may not support collaborative work. There are two examples which show that the physical output of the computational crafts platforms does not seem to support learners to construct collaboratively: the Arduino Lilypad platform supports construction by sewing, and the sewing process is quite individually; the i*CATch robotic platform in our study has a limited physical space of the interface box. In contrast, there are another two examples that support learners to construct the craft collaboratively: the TeeBoard/LilyPad and the i*CATch apparel platforms in our study both have plug-and-play electronic components and a wide physical area of a garment substrate for simultaneous construction. Therefore, computational crafts construction platforms support collaboration, which is mainly related to the design of the output construction interface.

The storytelling augmented computational apparel platform uses the i*CATch apparel platform, and it also uses hybrid graphical and textual programming environment and supports collaborative construction, which is different from the traditional graphical [2, 111] or physical storytelling [45, 91] platforms in which learners tend to work individually for both input and output parts.

6.3 Domains of Tasks

Traditional programming tasks that are related to robotics seem to attract mainly boys [108]. To broaden the potential computing student population, researchers develop toolkits to provide potential domains which are not limited to science topics to motivate children to learn programming, such as storytelling [73, 91, 106] and e-textiles [19, 20, 70]. In our study, the three major categories of computational construction platforms were used in different domains of tasks such as e-fashion,

wearable computing, robots and storytelling. Our findings show that children had different levels of engagement in different domains. To investigate children's engagement in a particular domain, three stages of children's responses to the domain are considered: recruitment period (before joining the activity), performing assigned tasks and after the activity.

In a recruitment period, children's decision is based on the introduction of advertisements on the news or the theme of a booth whether to join the workshops or activities. In this study, the gender ratio in the storytelling workshops and the game booth with the e-fashion theme was fairly even while the ratio of the boys in the robotic and wearable computing workshops (with craft making) was higher. It is clear that even though workshops involve other learning or teaching approaches such as crafts making, children are attracted to join the workshop by the theme, but not the content of the workshop, which may be due to the impression of science and arts subjects that influences their perception of the workshop.

In performing the assigned tasks, if children are engaged in their tasks, they are eager to work on their tasks, which may be reflected on the time-on tasks [15, 43, 88] and positive emotions or interests [113]. In the game booth study, the boys and girls exhibited different approaches in the e-fashion domain. The boys tried to form different combinations of badges to design functionality while the girls spent more time on decorations. Some participants, both boys and girls, obviously wanted to keep working on their tasks until we stopped them while some just attempted a minute. In our workshops, regardless of the themes, the boys and girls also have their unique styles to express their engagements. While boys focused on programming and girls more on making the decorations, both were eager to work on their tasks even for both the craft part and programming part; they often requested to have more time to finish their tasks. There were some exceptional cases. One is in the primary-school technology workshop: the boys did not want to use the tangible programming system (i*CATchBadges) to tell a story, since they disliked creating stories and liked learning technology; another one is common to all workshops: there were usually one to two groups who lost their interests in their tasks after they failed in a number of trials. The first exception suggests that teaching or learning

approaches need to be with the theme of a workshop. The second exception reminds us to be careful about the level of difficulty of tasks and the instruction given to support them.

After the activities or courses, if children are still engaged in that domain or learning technology and programming, they will come back again to join the same activity or other technology courses or ask their friends to join them. It is difficult to follow up all children's participation in relevant science activities after our activities. Based on our enrollment records, there were a few children who joined our different workshops in different years. Based on the post-course surveys, if children have a positive feeling about the workshops (e.g. complete all tasks successfully; achieve their expected design), they will be likely to pursue other technology and computing courses, either boys or girls. In game booth study, one girl invited her friend to come to our booth, after finishing her e-fashion design. We were interested to know whether she wanted to learn writing programs to control electronic devices instead of plugging them on a jacket. Her reply was, "Yes, if we can make our own!" Even though the activity time was very short, around 3 to 5 minutes, it is still possible to motivate children to join relevant workshops. In the primary-school wearable computing workshop, there was a girl who learned many different kinds of computational platforms and also succeeded in all our challenges. She reported, "Hope to have an advanced class!" In the performance workshops, there was a primary-school girl who created a story *The Exam* with programming electronic devices to express emotions. She commented, "I am interested in programming i*CATch board!" There was another middle-school girl said, "She never thinks about learning programming and sciences, and now it is possible in the future." They were very positive to our workshops and tended to be engaged in technology and computing, but there is a countering example: one primary-school girl in the TeeBoard workshop said, "I am very much enjoyed this workshop, even though I may not choose science subjects to study in the future." Overall, we still believe that most children who have positive feelings have motivation to pursue learning technology in the future.

To conclude, the theme of a workshop is the most important step to engage with a particular group of children who may be interested in science, arts or both. The second step is to maintain children's engagement during the activities by developing instructors' awareness of the children's behaviors on tasks (e.g. feel too difficult or dislike the teaching or learning approaches). If children have positive experience in the activity, they will tend to pursue technology and programming. In other words, regardless of the categories of computational construction kits, if the kits support a domain that attract a group of children to join, it becomes a motivating way to learn technology and programming for that group of children.

6.4 Learner's Characteristics

In designing a construction kit, researchers usually consider the gender factor when deciding which programming domains or paradigms should be supported by a platform. Through a series of case studies, it was found that the girls can be more engaged and have better performance in their works than the boys. Therefore, children's interests do not only depend on gender. This finding is similar to Kelleher et al.'s findings in their study about children's performance and interests in programming [73], which depends on programming experience rather than on gender. Age range seems to be another design factor to decide the complexity of the systems because of the child's cognitive development [98]. However, there are now more children who have the opportunity of learning computational concepts through creating animations and games or constructing robots in workshops compared to the past. Some primary school students may have more experience in programming than middle school students, just like the primary-school girls in one of our wearable computing workshops. The discrepancy in students' programming experience is reflected in the zone of proximal development and not the mental age [129].

The main objective of three streams of the computational toolkits is similar, which is to broaden the potential children's population to learn programming by gender and age range. All three of the computational toolkits basically reached this objective. The exception is the computational crafts stream, but we believe that it will achieve this objective if the theme of a workshop was more neutral. Therefore,

we propose to use learning styles [40] to analyze the suitability of three categories of the computational toolkits for a particular group of children's learning styles.

Sensing versus Intuitive

The design of the computational crafts construction kit is clearly intended for children to build an artifact such as robots or smart clothing (functional design) to solve computational problems. It should be more suitable for sensing learners. In contrast, the design of our tangible programming system or storytelling argument with apparel platform requires children to associate concrete physical outputs with some meanings which may be related to functions or stories. Wearable computing platforms for e-fashion design also require children to have a creative mind to use electronic devices to express aesthetics. The association process is abstract to some children, and it should be more suitable for intuitive learners.

Visual versus Verbal

The programming environment of the three main computational construction kits can be tangible, graphical and textual. In our workshops, most children reported that they preferred programming in graphics which seems to be a kind of visual learning style. In contrast, few children, especially male children, preferred programming in text. These children should belong to verbal learning style. In our primary-school technology workshop, there is no significant difference on the performance of children using tangible or graphical interfaces. Horn et al. [58] also had a similar finding between tangible and graphical interfaces. Thus, we consider that tangible representations are similar to visual representations, which should be also suitable for visual learners.

Active versus Reflective

Most tangible construction kits have designs that tend to support collaborative construction. This takes advantage of collaborative learning. The computational construction platforms used in this study also support three to four children working together. The exception is robotic construction platform which is limited because of

the physical size. If the kits can support collaborative work, they can support individual work. Therefore, most tangible construction kits can satisfy the learning style of active learners and reflective learners while most graphical construction kits with one mouse or keyboard control cannot.

Sequential versus Global

The learning computational concepts approach of tangible programming systems focus on the flow of a dynamic system by constructing bricks one by one; the nature of tasks in the robotic or wearable computing (functional) domain tends to be action-oriented or problem-oriented, which requires children to have linear thinking process and to learn problem solving by divide-and-conquer strategies (i.e. sequential learners). In contrast, the nature of tasks in the e-fashion or storytelling domain tends to be design-oriented and event-oriented, which requires children to think in parallel and holistically (i.e. global learners).

Overall, tangible computational construction kits accommodate a wider range of learning styles. Tangible programming systems are suitable for intuitive, visual and sequential learners; computational crafts construction kits are suitable for most of the learning style of children, which depends on application domain; storytelling construction platforms are suitable for intuitive, visual or verbal and global learners.

6.5 Learning Environments

An appropriate learning environment allows children to become expressive using new technologies, and this expressive experience offers them the opportunity to explore basic concepts [8, 105]. Children usually learn computational construction kits in a workshop carried out in a classroom, and the workshop is similar to the general outreach programs with a curriculum and an instructor [19, 73]. This learning environment provides less opportunity for children to take an active role. Some researchers propose a learning method that is related to the “Atelier” or “Studio” style of working [22, 76]. This learning method aims to facilitate a learning environment that engages learners in active learning and positive social interaction

[70]. On the other hand, some researchers have proposed learning computational construction kits through online communities such as Instructables [66] and LilyPond [81] that introduce how to build construction artifacts, where can be shared with and discussed by others. These supportive social communities support children to have Constructionist Learning [96] and Zones of Proximal Development (ZPD) [129], which encourage creativity, problem solving and engagement. Furthermore, some researchers believe that science museums assist children in learning scientific or computational concepts [9, 59]. Science museums offer a constructivist, self-guided learning environments, where allow children to learn through constructing their own knowledge by hands-on exploration and correlating with social aspects of engagement [9, 59]. Tangible programming systems allow children to explore computational concepts through constructing physical structures, often using a set of bricks. The complexity of the tasks is flexible, some of which can be finished in around 5 minutes and some others are much longer. This flexibility allows tangible programming systems to be used as a teaching tool in a classroom or science museum. Usually, tasks are related to the demonstration of dynamic behaviors or systems [134, 135], and they can be used to construct an artifact such as our game booth's activity that was to design an e-fashion. However, the artifacts are too simple, and the learning concept of tangible programming systems focuses on the process of constructing bricks to form a flow of program, which is similar to playing Scrabble to form a numbers of words. Therefore, it does not seem suitable to organize an online community for tangible programming systems to engage children in learning science and technology by sharing their products, but it is possible to form a club to appeal to a group of children playing with tangible programming systems after school or in leisure time, much like a Scrabble club or chess club. In computational crafts construction platforms, children have to use a programming environment to build their computational artifacts. Since the construction platforms involve the connection between the software and hardware components, this connection is quite complex when children encounter it for the first time. The learning process should be more efficient if children were to learn through an instructor giving an introduction in a classroom or guiding in studios, but there are

some examples of older children who were successful in self-learning through online tutorials [84]. The online communities work well for children to share their artifacts, even source codes, with their friends or to ask for questions with some experienced learners. Constructing a computational artifact usually requires at least an hour or above, hence computational crafts platforms seem not to be suitable for children to learn in science museum. Tangible storytelling platforms often require a number of ubiquitous settings and physical props, and those settings are usually new to children, which tend to be suitable for learning in a classroom. This applies to our proposed storytelling augmented computational apparel platform as well. Regardless of individual narration or collaborative performance, the children in our study reported that they often preferred to obtain feedback from their audiences in real time. Even though children can take a video as a record and upload it onto an online community, this sharing tends to be for entertainment, rather than to facilitate learning computational concepts. Tangible storytelling which is similar to traditional storytelling tends to be a classroom activity, which is different from creating a virtual story that can be done at home on the computer. Hence, tangible storytelling platforms are not suggested to be an online community. The time for creating a story usually is more than an hour, hence tangible storytelling platforms also seem to be as suitable for science museums or exhibitions.

The five factors of a computational construction kit discussed in this section are interrelated. It is difficult to comment which factor is more effective for learning, since it depends on the objective of a toolkit. It is also difficult for a tool to obtain the optimum level of each factor (see Table 6.1). Based on the analysis of these five factors, there are four suggested guidelines that we believe to be useful for designers and researchers to develop a new computational construction toolkit for children to learning computational concepts.

	Coupling of Concepts	Construction Interfaces		Domains of Tasks (Engagement)	Learners' Characteristics	Learning Environments
		Scaffolding	Collaborative			
Tangible Programming Systems	Passive physical bricks are mapped to dynamic computing concepts	Difficult to transit to text-based systems	Construction domain supports collaborative building	Monotonous tasks, short engagement	Diverse	Exhibitions / Museums, Classrooms
Crafts Platforms	Dynamic behavior of artifacts maps to dynamic computing concepts	Easy to transit to text-based systems	Depends on the platform types	Variety and challenging tasks, long engagement	Depends on the theme of projects	Classrooms, Studios, Online Community
Storytelling Platforms	Dynamic story flow / stage effects map to dynamic computing concepts	Easy to transit to text-based systems	Plug-n-play, bus based platform allows multiple learners to collaborate	Variety forms of storytelling, long engagement	Diverse	Classrooms

Table 6.1. Summary the five factors of the three mainstreams of computational construction kits

Determine places of learning

It is important to determine places of learning first, since it is directly related to the complexity of tasks (measurement by time) and domains of tasks (appealing to both genders). If a tool is designed for learning through public environments such as exhibitions or science museums, tasks should be finished within a short period of time and domains should appeal to males and females, similar to the tangible programming system's approach. Otherwise, designers have more freedom to determine the complexity and the domain of tasks, similar to the computational crafts or tangible storytelling's approach.

Support diversity of domains

In this thesis, the i*CATch construction platform was used in most case studies through different aspects to represent the three categories of computational construction toolkits. In our analysis, the three categories of computational construction toolkits are suitable for different learning styles of children. If a toolkit, such as i*CATch, can be applied into different domains for learning programming, instructors can flexibly design teaching materials or the theme of a workshop to

support different learning styles of children. It may be easier to find a domain which appeals to both genders.

Support simple and challenging tasks

It seems obvious that simple tasks appeal to children. But if tasks are too simple and monotonous, children will feel bored, and therefore simple tasks cannot encourage children to learn. However, if tasks are too challenging, children will be frustrated by repeated failures. As a result, challenging tasks also cannot motivate children to learn. To maintain children's learning motivation, computational construction toolkits should be flexible to support different complexity of tasks, and therefore children can learn computational concepts through accomplishing tasks from simple to complex. The i*CATch construction platform is an example, which contains a set of modules to form different combinations to design different levels of tasks.

Provide hybrid programming environments

To develop different kinds of computational construction kits, the main objective is to appeal to children, especially girls, to pursue the study of computer science in the future. Ultimately, learners still have to learn textual programming instead of tangible or graphical programming if they choose computer science as their major. To bridge the gap between learning textual and graphical or tangible programming, a hybrid programming system is proposed for each computational construction kit. This approach assists in transiting from tangible or graphical version to textual programming with increased complexity, and also entertains diversity of children with different levels of programming knowledge in an outreach program by its flexibility. Our studies used two hybrid graphical and textual environments, BrickLayer and the i*CATch IDE, which show the advantages of hybrid programming environments.

Chapter 7 Conclusion and Future Work

7.1 Conclusion

This thesis explores how children learn computational concepts through these platforms, by focusing on three types of tangible computational platforms: tangible programming systems, computational toolkits for crafts making, and computational interfaces for story creation. Our motivation for choosing these aspects is based on the hands-on learning theories proposed by three influential educators: Montessori, Froebel and Vygotsky.

In the study on tangible programming systems, we investigated how children understand the abstract concepts by building block functions to simulate functions of smart clothing and story flows, as well as computational concepts such as looping and branching. Two paper prototypes and a simulation were designed. The paper prototypes studies helped us to determine some important design dimensions of a tangible programming language for children to learning computational concepts, including knowledge of computational concepts, increment of abstraction, space of imagination and function-based expression. To find more evidence on the design dimensions, the i*CATch construction platform was modified into the i*CATchBadges tangible programming system to simulate a tangible programming system for further study. The results show that the children were able to successfully grasp some basic computational concepts through the tangible programming system to solve some simple tasks. Furthermore, the system was examined in two different learning environments. The results indicate that tangible programming systems which have input and output on the same interface support simple installation, which makes them suitable for different learning environments including indoor museums and outdoor exhibitions. Hence, it has potential to draw the attention of a diversity of children engaged in computing and technology through different learning environments.

To study the computational toolkits for crafts making, three computational craft platforms were adopted as case studies, including Lilypad Arduino with TeeBoard platform, i*CATch apparel platform and i*CATch robotic platform. The focus was

on crafts making in robotic and apparel domains, integrating traditional materials such as paper and cloth with the electronic devices. Through these three different design approaches of the computational craft platforms, we gain a deeper understanding of how children use computational platforms for creating crafts. Different construction interfaces of computational craft toolkits support different levels of creativity. The wearable computing platform is able to facilitate the learning of computational concepts by a wider diversity of children via enabling a wider selection of project themes. The flexible construction interfaces encourage students to write complex programs. Overall, the results confirm that computational crafts toolkits attract both boys and girls and provide a space for children to exercise their creativity and practice their programming skills.

To study the computational interfaces for story creation, a storytelling expression media was proposed: storytelling augmented with computational apparel for children to learn the computational concepts through story creation. The i*CATch wearable computing platform was used as the storytelling media. Two syllabi of the workshops were designed to compare how children use computational apparel platforms to tell a story. The results show that computational apparel can be a media for children to tell a story, and indicate that there is room for children to improve their stories through more practices on storytelling and programming electronic device skills. Furthermore, through the analysis of the children's computer programs, we notice that the children could figure out different programming approaches to construct their programs; hence it shows that this proposed storytelling through computational apparel can function as a new tangible interactive storytelling method for children to learn computational concepts.

The exploration of each aspect has adopted different research methodologies, curricula and evaluation approaches, which can be used as a reference for running other case studies or workshops. The findings of several empirical user studies have also helped us to understand the characteristics of the three types of platforms and how they support children's learning of computational concepts.

To this end, this thesis also proposes five factors for a cross-evaluation between these three categories of computational construction toolkits. The five factors

include coupling of computational concepts, construction interfaces, domains of tasks, learners' characteristics and learning environments. The cross-evaluation of three types of tangible computational construction kits reveals fundamental differences related to children's learning of computational concepts. It concludes with four suggested guidelines: determine places of learning, support diversity of domains, support simple and challenging tasks and provide hybrid programming environments. We believe that the guidelines should be useful for designers and researchers who wish to develop a new computational construction toolkit for children to learning computational concepts.

7.2 Future Work

There are still some the state-of-the-art technologies and innovative methodologies which have not been applied into tangible construction platforms for children to learn computational concepts. The suggested factors and guidelines facilitate to design and examine new platforms. There are a few directions for possible future work.

Explore extensions of each mainstream of computational construction kits

Basically, the technology applied in our study is to extend the i*CATch wearable computational construction platform to simulate a tangible programming system and propose to be an interactive storytelling media to explore the major types of computational construction kits. Therefore, it should have potential methodologies to extend the current computational construction kits to investigate the strengths and weaknesses. Furthermore, there are more advanced technologies available, for example, paper computing [21] and skin input [54], which can be developed into a new interface of computational construction kits to arouse children's interests in computer science and technology.

Explore new streams of computational kits for learning computational concepts

There are more than three categories of computational construction kits. For example, tangible interactive games are common [85, 115, 132], even in commercial

products such as a dance mat and Wii. However, few of them address children's learning of computational concepts, nor do they propose to have a simple programming environment by interactions with the physical objects [41]. Actually, it may be possible for children, especially older children, to design their physical environment games or bodily interactive games by writing programs on physical or tangible modules. On the other hand, some researchers have proposed a concrete real-world cooking scenario for children to learn programming [122]. Furthermore, social network is so popular to nowadays, some researchers have also developed some prototypes for tangible social networks [69], which may also have potential as a new tangible platform for children to learn computational concepts.

Investigate potential learning environments

Learning computational concepts is usually regarded as learning with a computer in the indoor environment. Wireless technology has become mature enough to satisfy the needs of mobility and electronic devices are getting smaller. All these technologies are possible to move the classroom into the outdoor environment. Our study of a tangible programming system run in game booth is an example. On the other hand, some researchers advocate outdoor field learning in which children on field trip are encouraged to explore their surrounding environments to gain knowledge during the immersive learning experience [71]. Some researchers have developed a mobile device which supports children's collaborative artifact creation and play in outdoor environments [12, 28, 107]. It may be also possible to develop a mobile tangible construction platform for children to write programs to build simple functions to measure some data in outdoor environment such as air pollution, UV light and humidity.

Appendix A: Crafts Making Workshops Syllabi

(A) Syllabus of TeeBoard with LilyPad Workshops

Chapter 1: Introduction to E-textiles and Wearable Computing

Contents: Display latest applications of e-textiles and wearable computing such as smart clothing, performance dressing, and intelligent bags.

Sample task: Discuss the possible features of e-textiles and wearable computing.

Learning outcomes: Students should obtain some background in e-textiles and wearable computing.

Chapter 2: Electronic Circuit Theory

Contents: Electricity, electrical circuits and electrical resistance.

Sample tasks:

1) Create a complete circuit with four ribbons and one LED.
2) Create a serial circuit with five ribbons and two LEDs.
3) Create a parallel circuit with six ribbons and two LEDs.
4) Create a short circuit with adding one ribbon on the current circuit.

Learning outcomes: Students should understand basic electrical knowledge such as voltage, conductivity and resistance.

Chapter 3: T-Shirt Circuit Design

Contents: Introduction to the circuit design and a t-shirt breadboard.

Sample tasks:

1) Create a circuit with three LEDs: one LED is in the centre part of the t-shirt and another two are in two sleeves separately.
2) Create a circuit with two LEDs in parallel connected to one LED in series at the back of the t-shirt.

Learning outcomes: Students should understand the basics of circuit structures and the use of breadboards.

Chapter 4: Integrated Circuits (ICs)

Contents: ICs introduction and the output signals of ICs

Sample tasks:

1) Connect an IC with two LEDs to observe the flash pattern of LEDs on the t-shirt breadboard.
2) Connect an IC with four LEDs to observe the flash pattern of LEDs on the t-shirt breadboard.
3) Compare the difference between the results of connecting with two LEDs and four LEDs.

Learning outcomes: Students should understand the concept of ICs and the notion of a predefined logical output.

Chapter 5: Computational Platforms

Contents: Introduction to programming environment such as Arduino IDE, Bricklayer and i*CATch IDE, and hardware such as microcontrollers, actuators and sensors.

Sample tasks:

1) Write a simple program such as turning the microcontroller's LED on.
2) Compile and execute a program, and upload it onto a microcontroller.

Learning outcomes: Students should understand how to operate the programming environment and load programs onto a microcontroller.

Chapter 6: Introduction to Sequential Logic

Contents: Incorporate a microcontroller into the t-shirt breadboard and program a circuit consisting of actuators (e.g. LEDs) in sequential logic.

Sample tasks:

1) Blink a multicolored LED in rainbow color sequence.
2) Blink four LEDs one by one for 0.5 second each in order.

Learning Outcomes: Students should be able to write a program to solve a problem involving the sequential logic.

Chapter 7: Introduction to Repetitions

Contents: Incorporate a microcontroller into the t-shirt breadboard and program a circuit consisting of actuators (e.g. LEDs) and sensors (e.g. light sensors) in repetitions.

Sample tasks:

1) Blink a multicolored LED repeated six times in red, repeated four times in green and repeated two times in blue.
2) Use a light sensor to collect five readings after every one second.

Learning Outcomes: Students should be able to write a program to solve a problem involving repetitions.

Chapter 8: Introduction to Conditionals

Contents: Incorporate a microcontroller into the t-shirt breadboard and program a circuit consisting of sensors (e.g. accelerometers and light sensors), and actuators (e.g. LEDs) in the conditional logic.

Sample tasks:

1) Blink a multicolored LED in rainbow color sequence when in the dark.
2) Turn on two LEDs at the back of the t-shirt when moving both arms.

Learning Outcomes: Students should be able to write a program involving the conditional logic that reads in signals from sensors and sends simple signal to output devices.

Chapter 9: Project

Contents: Combine all the electronic circuit and computational concepts to design an interactive t-shirt.

Sample project: Design and construct an interactive t-shirt in sport theme.

Learning outcomes: Students should be able to exercise their creativity as well as their newly learned programming and electronics knowledge.

(B) Syllabus of i*CATch for Apparel Workshops

Chapter 1: Introduction to E-textiles and Wearable Computing

Contents: Display latest applications of e-textiles and wearable computing such as smart clothing, performance dressing, and intelligent bags.

Sample task: Discuss the possible functions of e-textiles and wearable computing.

Learning outcomes: Students should obtain some background in e-textiles and wearable computing.

Chapter 2: Introduction to Electronic Devices for Apparel

Contents: Input devices include switches, joysticks, light sensors, ultrasonic sensors; output devices include LEDs, buzzers and vibration motors.

Sample task: Explore and write down the physical properties of the electronic devices for apparel by providing a pre-programmed microcontroller.

Learning Outcomes: Students should learn some basic physical properties of each provided electronic device.

Chapter 3: Introduction to Programming Environment

Contents: Introduction to programming environment such as Arduino IDE and i*CATch IDE.

Sample tasks:

1) Write a simple program such as turning the microcontroller's LED on.
2) Compile and execute a program, and upload it onto a microcontroller.

Learning outcomes: Students should understand how to operate the programming environment and load programs onto a microcontroller.

Chapter 4: Introduction to Sequential Logic

Contents: Plug a microcontroller and actuators (e.g. LEDs and buzzers) into the i*CATch jacket and program them in sequential logic.

Sample tasks:

1) Blink a multicolored LED in rainbow color sequence.
2) Play the C major scale on a buzzer.

Learning Outcomes: Students should be able to write a program to solve a problem involving the sequential logic.

Chapter 5: Introduction to Repetitions

Contents: Plug a microcontroller, actuators (e.g. LEDs and buzzers) and sensors (e.g. light sensors) into the i*CATch jacket and program them with applying the concept of repetition.

Sample tasks:

1) Blink a multicolored LED repeated six times in red, repeated four times in green and repeated two times in blue.
2) Use a light sensor to collect five readings after every one second.

Learning Outcomes: Students should be able to write a program to solve a problem involving repetitions.

Chapter 6: Introduction to Basic Conditionals

Contents: Plug a microcontroller, actuators (e.g. LEDs and buzzers) and sensors (e.g. switches and joysticks) into the i*CATch jacket and program them in the conditional logic.

Sample tasks:

1) Blink a multicolored LED in rainbow color sequence when a switch is pressed.
2) Control four lighting patterns by turning different directions of a joystick.

Learning Outcomes: Students should be able to write a program involving the conditional logic that reads in signals from simple sensors and sends simple signal to output devices.

Chapter 7: Introduction to Advanced Conditionals

Contents: Plug a microcontroller, actuators (e.g. LEDs and buzzers) and sensors (e.g. light sensors and ultrasonic sensors) into the i*CATch jacket and program them in the conditional logic.

Sample tasks:

1) Blink a multicolored LED in rainbow color sequence when in the dark.
2) Turn on two LEDs in the front of the jacket when someone is getting closer to the wearer.

Learning Outcomes: Students should be able to write a program involving the advanced conditional logic that reads in signals from sensors and send simple signal to output devices.

Chapter 8: Project

Contents: Combine all the computational concepts to design an interactive jacket.

Sample project: Design and construct an e-fashion smart jacket.

Learning outcomes: Students should be able to exercise their creativity as well as their newly learned programming knowledge.

(C) Syllabus of i*CATch for Robots Workshops

Chapter 1: Introduction to Electronic Devices for Robots

Contents: Input devices include switches, joysticks, light sensors, ultrasonic sensors; Output devices include motors, LEDs and buzzers; data transmission devices include USB cables and Bluetooth dongle.

Sample task: Explore and write down the physical properties of the electronic devices for robots.

Learning outcomes: Students should understand the physical properties of the electronic devices for robots.

Chapter 2: Introduction to Building a Robotic Chassis

Contents: Discuss some basic mechanical principles to build a robotic chassis such as the size of wheels and the weight of the car body.

Sample task: Design and build a robotic chassis incorporated with two motor wheels and the interface box.

Learning outcomes: Students should be able to design their own robotic cars which will be used for solving challenging tasks.

Chapter 3: Introduction to Programming Environment

Contents: Introduction to programming environment such as Arduino IDE and i*CATch IDE

Sample tasks:

1) Write a simple program such as turning the microcontroller's LED on.
2) Compile and execute a program, and upload it onto a microcontroller.

Learning outcomes: Students should understand how to operate the programming environment and load programs onto a microcontroller.

Chapter 4: Introduction to Sequential Logic

Contents: Incorporate a microcontroller and actuators (e.g. motors, buzzers and LEDs) into the interface box, and program the movement of a robot in sequential logic.

Sample tasks:

1) Move a robot forward for 0.5 second and give a sound.
2) Move a robot forward for 0.5 second and backward for 0.5 second, and finally blink a multicolored LED in rainbow color sequence.

Learning outcomes: Students should be able to write a program to solve a problem involving the sequential logic.

Chapter 5: Introduction to Repetitions

Contents: Incorporate a microcontroller, actuators (e.g. motors, buzzers and LEDs) and sensors (e.g. ultrasonic sensors and light sensors) into the interface box, and program the movement of a robot with applying the concept of repetition.

Sample tasks:

1) Control a robot to move in a square shape.
2) Print out the intensity of light through a light sensor in every one second.

Learning outcomes: Students should be able to write a program to solve a problem involving the repetition logic.

Chapter 6: Introduction to Conditionals

Contents: Incorporate a microcontroller, actuators (e.g. motors, buzzers and LEDs) and sensors (e.g. ultrasonic sensors and light sensors) into the interface box, and program the movement of a robot in conditional logic.

Sample tasks:

1) Give a sound when a robot hits the wall.
2) Turn on a LED when a robot moves in the dark.

Learning outcomes: Students should be able to write a program involving the conditional logic that reads in signals from sensors and sends simple signal to output devices.

Chapter 7: Solving Challenging Tasks

Contents: Combine all the computational concepts to solve a series of challenging tasks.

Sample challenging tasks:

1) Bombs detection: Detect the bombs by using light sensor to identify the color of objects (e.g. ball in red color represents a bomb.), and give a sound when a bomb is found.
2) Escape the maze: Detect the walls by using ultrasonic sensor to find out the way to escape out of the maze.

Learning outcomes: Students should be able to exercise their problem solving skills as well as their newly learned programming and mechanics knowledge.

Appendix B: Storytelling Workshops Syllabi

(A) Syllabus I of Storytelling Workshops

Chapter 1 to 6: Electronic Devices and Computational Concepts

May refer to syllabus of i*CATch for apparel workshops chapter 2 to 7 in Appendix A (B).

Chapter 7: Project – Storytelling

Contents: Combine all the computational concepts and students' own storytelling skills to create an interactive story.

Sample stories:

1) Design and construct an interactive jacket as a stage for telling a story.
2) Design and construct an interactive jacket as a costume to highlight the main character.

Learning outcomes: Students should be able to exercise their creativity and imagination as well as their newly learned programming and presentation skills.

(B) Syllabus II of Storytelling Workshops

Chapter 1: Introduction to Forms of Storytelling

Contents: Introduce the forms of storytelling (e.g. pantomime and image theatre), story elements (e.g. setting, plot, conflict and character), and storyboarding.

Sample tasks:

1) Image theatre: A group of students give a static pose to express a particular emotion (e.g. happy, angry or sad).
2) Pantomime: A group of students perform a number of actions without dialogue to express a scene in a particular venue (e.g. in a restaurant, in a classroom or on the road).

Learning outcomes: Students should learn some basic storytelling skills and should be able to express a scene in different forms of storytelling.

Chapter 2 to 7: Electronic Devices and Computational Concepts

May refer to syllabus of i*CATch for apparel workshops chapter 2 to 7 in Appendix A (B), but the tasks require students to associate the effects of the electronic devices with some meanings.

Chapter 8: Project – Drama / Pantomime

Contents: Combine all the taught computational concepts and storytelling skills to create a drama or pantomime.

Sample stories:

1) Design and construct an interactive jacket to produce stage effects for performing a drama.
2) Design and construct an interactive jacket to express the main character's emotion.

Learning outcomes: Students should be able to exercise their creativity and imagination as well as their newly learned programming and presentation skills.

Appendix C: Sample Program of i*CATchBadges Study in Game Booth

(Refer to *Section 3.4.1 i*CATchBadges Study in Game Booth*, page 54)

(A) A sample of the program without associations with felt icons

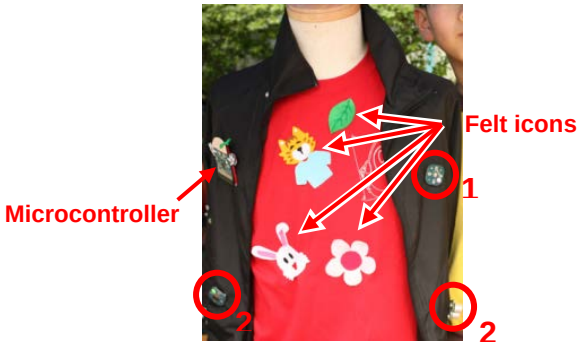


Figure C.1. An example of a boy's jacket with labels

Author: A primary-school boy

Steps of the program (Figure C.1):

1. *LED light [Flash in a multicolor light pattern]: Use as a decoration
2. Ultrasonic sensor (right) and vibration motor (left) [A pair of badges contains a condition function – when ultrasonic sensor detects something within 20 cm distance, the vibration motor turns on]: Function as an anti-stalker

* Read as: Badge type [Physical function]: Representation(s)

(B) A sample of the program with associations with felt icons

Figure C.2. An example of a girl's jacket with labels

Author: A primary-school girl

Steps of the program (Figure C.2):

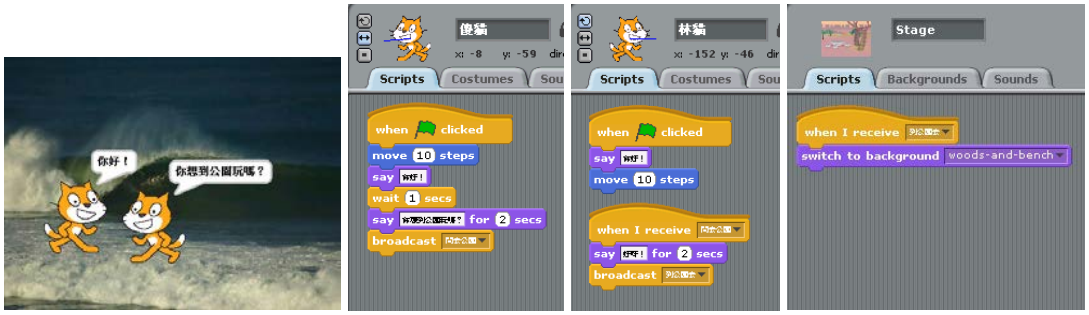
1. Buzzer with a fire felt [Play a C note sound repeated 3 times]: There is a fire accident on the sea.
2. LED light with a treasure felt [Flash in a multicolor light pattern]: A treasure sinks into the sea.
3. Buzzer with a shark felt [Play a roar sound repeated 2 times]: A shark threatens the clown fish near to the treasure, since the shark wants to get the treasure.
4. LED light with a clown fish felt [Flash in red color]: A clown fish is so afraid and swims away.

* Read as: Badge type [Physical function]: Representation(s)

Appendix D: Sample Program of i*CATchBadges Study in Technology Workshop

(Refer to *Section 3.4.2 i*CATchBadges Study in Technology Workshop*, page 55)

(A) A sample program with using a graphical programming environment (Scratch)



(a) (b) (c) (d)

Figure D.1. (a) A sample output screen of a story with using Scratch created by a mixed group of students (b) Crazy Cat's script (c) Lam Cat's script (d) Stage's script

Author: A primary-school mixed group (2 girls and 1 boy)

Storytelling Time: Around 1:30 minutes

Story script:

Crazy Cat met Lam Cat in a breach. Crazy Cat and Lam Cat greeted each other (Refer to program step 1) Crazy Cat waited a second and asked whether Lam Cat wanted to go to a park (Refer to program step 2). Lam Cat replied that that's good. Crazy cat replied to Crazy Cat that's good (Refer to program step 3).

Steps of the program (Figure D.1):

1. Crazy Cat and Lam Cat's program scripts run at the same time:
   when green flag clicked → move 10 steps → say "你好！"
2. Crazy Cat's program scripts run:
   wait 1 sec → say "你想到公園玩嗎？" → broadcast "問去公園"
3. Lam Cat's program scripts run:
   when I received "問去公園" → say "好呀！" → broadcast "到公園去";
   Stage's program scripts run:
   when I received "到公園去" → switch to background "woods and bench"

(B) A sample program with using a tangible programming environment (i*CATchBadges)

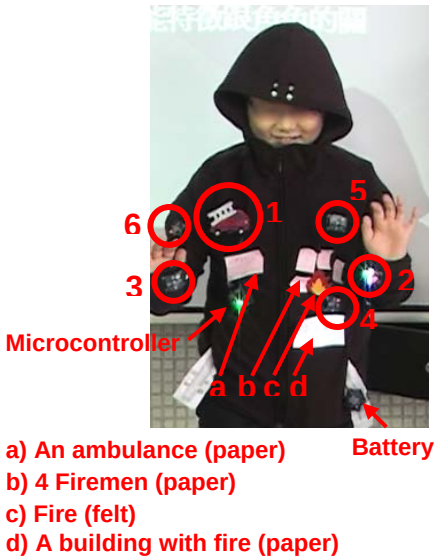


Figure D.2. An example of a story using i*CATchBadges created by a mixed group of students

Author: A primary-school mixed group (2 girls and 1 boy)

Storytelling Time: Around 1 minute

Story script:

One day, there was a fire accident in a building. Someone called the police, and after a while four fire-engines and one ambulance came to the accident scene. Luckily, all residents left the building safely. However, one fireman inhaled too much smoke and finally died after being taken to a hospital.

Steps of the program (Figure D.2):

1. Buzzer with a fire-engine felt [Play a siren sound repeated 2 times]: A siren of a fire engine
2. LED light [Flash in a multicolor light pattern]: Left front light of a fire engine
3. LED light [Flash in a multicolor light pattern]: Right front light of a fire engine
4. LED light [Flash in a multicolor light pattern]: For car body decoration
5. LED light [Flash in a multicolor light pattern]: For car body decoration
6. Vibration Motor [Generate a vibration pattern]: The vibration of a moving fire-engine

Appendix E: Sample Program of Using TeeBoard with LilyPad

(Refer to *Section 4.2.2 Case Study 1:TeeBoard with LilyPad*, page 72)

A sample code of the e-fashion project

Author: A primary-school-girl group (4 girls)

Theme: Zoo

Program code:

```
#include <sensor.h>
int main(void){
	init();
	int red = 0;
	Wire.begin();
	Serial.begin(9600);

	while(1){
		red = getLightSensorReading(5);
		Serial.println(red);
		delay(1000);

		if (red<600) { // when in dark environment, animals come out
			ledOn(13, LED);  // A bear felt with an LED: A bear wakes up.
			delay(1000);
			ledOff(13, LED);
			delay(2000);

			ledOn(9, LED);  // A cow felt with an LED: A cow wakes up.
			delay(1000);
			ledOff(9, LED);
			delay(2000);

			ledOn(3, LED); // A cat felt with an LED: A cat wakes up.
			delay(1000);
			ledOff(3, LED);
			delay(2000);

			ledOn(5, LED); // A rabbit with an LED: A rabbit wakes up.
			delay(1000);
			ledOff(5, LED);
			delay(2000);
		}
		else{
		}
	}
	while(1);
	return 0;
}
```

Appendix F: Sample Program of Using i*CATch Apparel Platform

(Refer to *Section 4.2.3 Case Study 2: i*CATch Apparel Platform*, page 76)

A sample code of the e-fashion project

Author: A primary-school-girl group (3 girls)

Theme: Hi-tech hiking jacket

Program code:

```
#include <Wire.h>
#include <sensor.h>
void setup()
{
   Serial.begin(9600);
   Wire.begin();
}
void loop()
{
   if (getUltraSonicSensorReading (136) <70)  // a follower detection function
   {
      ledOff (122, GREEN); ledOff (122, BLUE);
      soundLibrary (86, 11); ledOn (122, RED);
   }
   else if (getUltraSonicSensorReading (136) >70)
   {
      noSound (86, 11); ledOff (122, RED);
      ledOn (122, GREEN); ledOn (122, BLUE);
   }
   if (isIRDetected (56))  // play a song by using a IR remote control
   { // Mary had a little lamb
      playSound (80, "E", 3, 10); playSound (80, "D", 3, 10); playSound (80, "C", 3, 10);
      playSound (80, "D", 3, 10); playSound (80, "E", 3, 10); noSound (80, 1);
      playSound (80, "E", 3, 10); noSound (80, 1); playSound (80, "E", 3, 20);

      playSound (80, "D", 3, 10); noSound (80, 1); playSound (80, "D", 3, 10);
      noSound (80, 1); playSound (80, "D", 3, 20); playSound (80, "E", 3, 10);
      playSound (80, "G", 3, 10); noSound (80, 1); playSound (80, "G", 3, 20);
      noSound (80, 1);

      playSound (80, "E", 3, 10); playSound (80, "D", 3, 10); playSound (80, "C", 3, 10);
      noSound (80, 1); playSound (80, "D", 3, 10); noSound (80, 1);
      playSound (80, "E", 3, 10); noSound (80, 1); playSound (80, "E", 3, 10);
      noSound (80, 1); playSound (80, "E", 3, 10);

       playSound (80, "C", 3, 10); playSound (80, "D", 3, 10); noSound (80, 1);
       playSound (80, "D", 3, 10); playSound (80, "E", 3, 10); playSound (80, "D", 3, 10);
       noSound (80, 1); playSound (80, "C", 3, 30); noSound (80, 1);
   }
```

```
  if (getJoyStickReading (44) ==1) // a neck massage function
  {
     vibrationMotorOn (92);
     delay (6000);
     vibrationMotorOff (92);
  }
  if (getJoyStickReading (44) ==2) //
  {
     onBoardLedOn (12); onBoardLedOn (13);
     delay (6000);
     onBoardLedOff (12); onBoardLedOff (13) ;
  }
  if (getJoyStickReading (44) ==3) // an illumination function
  {
     ledOn (116, RED); ledOn (122, RED); ledOn (110, RED);
     delay (3000);
     ledOn (110, BLUE); ledOn (122, BLUE); ledOn (116, BLUE);
     delay (3000);
     ledOff (110, RED); ledOff (122, RED); ledOff (116, RED);
     delay (3000);
     ledOn (110, GREEN); ledOn (122, GREEN); ledOn (116, GREEN);
     delay (3000);
     ledOff (110, BLUE); ledOff (122, BLUE); ledOff (116, BLUE);
     delay (3000);
     ledOn (110, RED); ledOn (122, RED); ledOn (116, RED);
     delay (3000);
     ledOff (110, GREEN); ledOff (122, GREEN); ledOff (116, GREEN);
     ledOff (110, RED); ledOff (122, RED); ledOff (116, RED);
  }
  if (getJoyStickReading (44) ==4) // play a song by using a joystick
  {
     ledOn (98, LED); // an LED attached with an “iPod” music player
      // Row Row Row Your Boat
     playSound (86, "C", 3, 10); noSound (86, 1); playSound (86, "C", 3, 10);
     noSound (86, 1); playSound (86, "C", 3, 15); playSound (86, "D", 3, 5);
     playSound (86, "E", 3, 10); noSound (86, 1);

     playSound (86, "E", 3, 15); playSound (86, "D", 3, 5); playSound (86, "E", 3, 15);
     playSound (86, "F", 3, 5); playSound (86, "G", 3, 20);

     playSound (86, "C", 4, 5); noSound (86, 1); playSound (86, "C", 4, 5);
     noSound (86, 1); playSound (86, "C", 4, 5); playSound (86, "G", 3, 5);
     noSound (86, 1); playSound (86, "G", 3, 5); noSound (86, 1); playSound (86, "G", 3, 5);

     playSound (86, "E", 3, 5); noSound (86, 1); playSound (86, "E", 3, 5); noSound (86, 1);
     playSound (86, "E", 3, 5); playSound (86, "C", 3, 5); noSound (86, 1);
     playSound (86, "C", 3, 5); noSound (86, 1);playSound (86, "C", 3, 5);

     playSound (86, "G", 3, 15); noSound (86, 1);playSound (86, "F", 3, 5); noSound (86, 1);
     playSound (86, "E", 3, 15); noSound (86, 1); playSound (86, "D", 3, 5); noSound (86, 1);
     playSound (86, "C", 3, 20); noSound (86, 1);
     ledOff (98, LED);
  }
}
```

Appendix G: Sample Program of Using i*CATch Robotic Platform

(Refer to *Section 4.2.4 Case Study 3: i*CATch Robotic Platform*, page 83)

A sample code of robotic challenging tasks:

Author: A middle-school-boy group (4 boys)

Task: To detect obstacles: the robotic car cannot bump into the obstacles and move away from them

Program Code:

```
#include <sensor.h>
int main(void){
	init();
	Wire.begin();
	while(1){
		if (getUltraSonicSensorReading(138)<=46){
			servoAntiClockwise(62);
			servoClockwise(64);
			delay(500);
			delay(100);
			servoStop(64);   // stop servo
			servoStop(62);   // stop servo
			delay(500);
			servoAntiClockwise(64);
			servoAntiClockwise(62);
			delay(1000);
			servoStop(64);    // stop servo
			servoStop(62);    // stop servo
		}
		else {
			servoAntiClockwise(64);
			servoClockwise(62);
			delay(100);
			servoStop(64);      // stop servo
			servoStop(62);      // stop servo
		}
	}
	while(1);
	return 0;
}
```

* Remarks: Students may consider the mechanical problems such as stopping servos which help to stabilize the car to turn around.

Appendix H: Sample Programs of Program Structures in Storytelling Workshop

(Refer to *Section 5.4.7 Support for Computational Learning*, page 128)

(A) Using sequentiality on events and timed delays

Author: A middle-school-boy group (4 boys)

Story title: Debt

Program code:

```
#include <sensor.h>
int main(void){                                                    → Program Begin
    init();
    Wire.begin();

    onBoardLedOn(13);                                              ┐
    for (int i=0; i<6; i++){                                       │
        // yellow light means the debtor is so scared              │
        ledOn(118, RED);                                           │ Event 1
        ledOn(118, GREEN);                                         │
        delay(500);                                                │
        ledOff(118, RED);                                          │
        ledOff(118, GREEN);                                        │
    }                                                              ┘

    delay(5000);                                                   → Timed Delay

    for (int i=0; i<6; i++){                                       ┐
        // red light means the debtor is so nervous                │
        ledOn(116, RED);                                           │ Event 2
        delay(500);                                                │
        ledOff(116, RED);                                          │
    }                                                              ┘

    delay(5000);                                                   → Timed Delay

    for (int i=0; i<6; i++){                                       ┐
        // white light means the debtor gets a shot from one creditor │
        ledOn(118, RED);                                           │
        ledOn(118, GREEN);                                         │
        ledOn(118, BLUE);                                          │ Event 3
        delay(500);                                                │
        ledOff(118, RED);                                          │
        ledOff(118, GREEN);                                        │
        ledOff(118, BLUE);                                         │
    }                                                              ┘

    delay(5000);                                                   → Timed Delay
```

```
    onBoardLedOff(13);
   // a series of sound and light effects means the debtor died
    soundLibrary(82, 4); // a waiting sound
    soundLibrary(82, 2); // a heart beat sound
    for (int i=0; i<10; i++){
    // red light
         ledOn(118, RED);
         delay(500);
         ledOff(118, RED);
    }
    // white light
    ledOn(118, RED);
    ledOn(118, GREEN);
    ledOn(118, BLUE);
    delay(5000);
    ledOff(118, RED);
    ledOff(118, GREEN);
    ledOff(118, BLUE);
    soundLibrary(82, 8);  // a fail sound
    delay(1000);
    soundLibrary(82, 1); // a long beep
    soundLibrary(82, 1); // a long beep
    soundLibrary(82, 8); // a fail sounds
    delay(10000);
    soundLibrary(82, 3); // an ambulance siren
    delay(5000);
    // a rapid sound means the creditors had a discussion
    soundLibrary(82, 11);
    // yellow light means the debtor resurrects
    ledOn(118, RED);
    ledOn(118, GREEN);
    while(1);
    return 0;
}
```

Event 4

→ **Timed Delay**

Event 5

→ **Program End**

* Remarks: Not all the timed delays are used to separate the events, while some timed delays are used to slow down the program to make the LED's light visible to human or to make a pause to separate two sound-libraries.

(B) Using Infinite loop with using joystick or switches triggers

Author: a middle-school-mixed group (2 girls, 1 boy)

Story title: Bullying

Program code:

```
#include <sensor.h>

int main(void){                                                    → Program Begin
    init();
    Wire.begin();


    while(1){                                                      → Infinity Loop Begin


        if (getJoyStickReading(42) == 1) {                         → If Joystick = 1

          // violet lights and a rapid sound mean the boy was being bullied.
          ledOn(116, RED);
          ledOn(116, BLUE);
          ledOn(110, RED);
          ledOn(110, BLUE);
          ledOn(112, RED);
          ledOn(112, BLUE);                                        } Event 1
          soundLibrary(78, 11);
          soundLibrary(86, 11);
          soundLibrary(80, 11);
        }
        else {

            if (getJoyStickReading(42) == 2) {                     → If Joystick = 2

              // blue lights and a slow sound mean the boy felt sad.
               noSound(78, 1);
               noSound(86, 1);
               noSound(80, 1);
               ledOff(116, RED);
               ledOff(116, BLUE);
               ledOff(110, RED);
               ledOff(110, BLUE);
               ledOff(112, RED);                                   } Event 2
               ledOff(112, BLUE);
               delay(100);
               ledOn(116, BLUE);
               ledOn(110, BLUE);
               ledOn(112, BLUE);
               soundLibrary(78, 10);
               soundLibrary(80, 10);
            }
```

```
            else {

                if (getJoyStickReading(42) == 3){                    → If Joystick = 3

                  /* vibration, red lights and an ambulance siren
                     mean the boy felt angry. */
                  noSound(78, 1);
                  noSound(86, 1);
                  noSound(80, 1);
                  ledOff(116, BLUE);
                  ledOff(110, BLUE);
                  ledOff(112, BLUE);
                  delay(100);
                  vibrationMotorOn(92);
                  ledOn(116, RED);
                  ledOn(110, RED);
                  ledOn(112, RED);
                  soundLibrary(78, 3);
                  soundLibrary(86, 3);
                  soundLibrary(80, 3);
                }                                                    } Event 3
                else {

                  if (getJoyStickReading(42) == 4){                  → If Joystick = 4
                    // turn off all effects
                    noSound(78, 1);
                    noSound(86, 1);
                    noSound(80, 1);
                    ledOff(116, RED);
                    ledOff(110, RED);
                    ledOff(112, RED);                                } Event 4
                    vibrationMotorOff(92);
                  }
                  else {
                  }
                }
            }
        }
    }                                                                → Infinity Loop End
    while(1);
    return 0;
}                                                                    → Program End
```

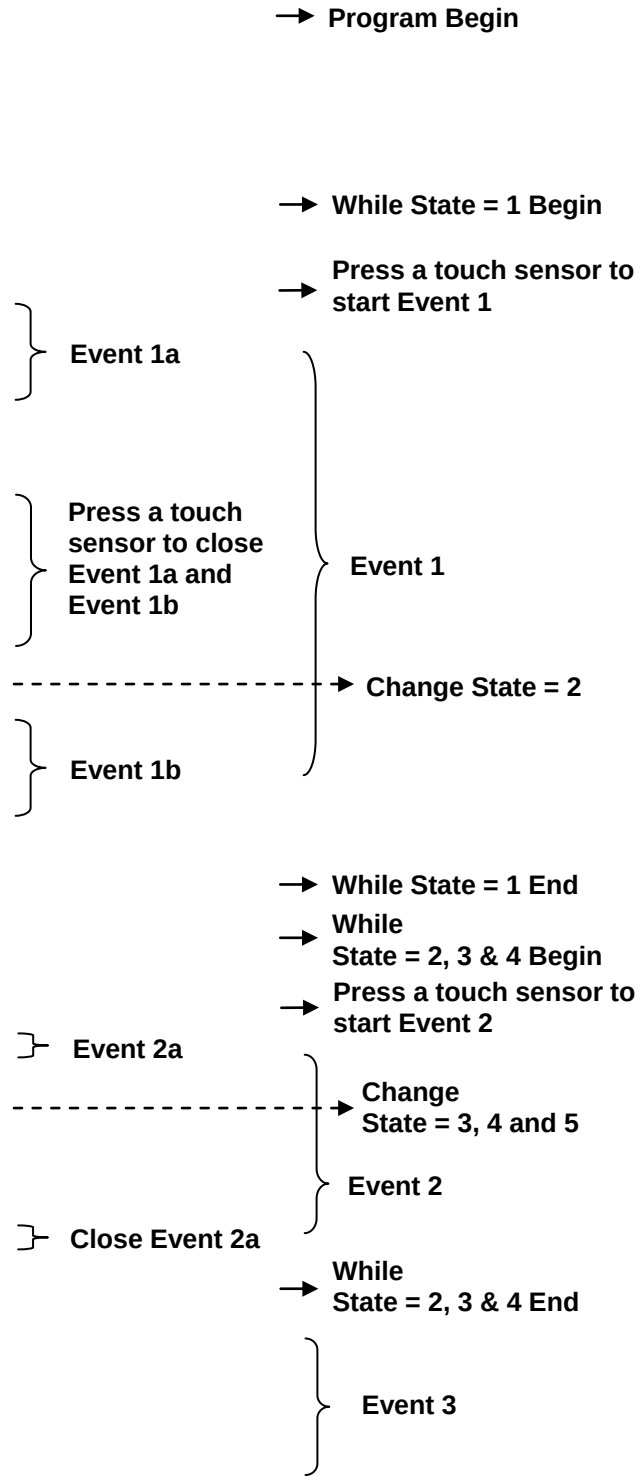
(C) Using sequentiality on states with a switch trigger

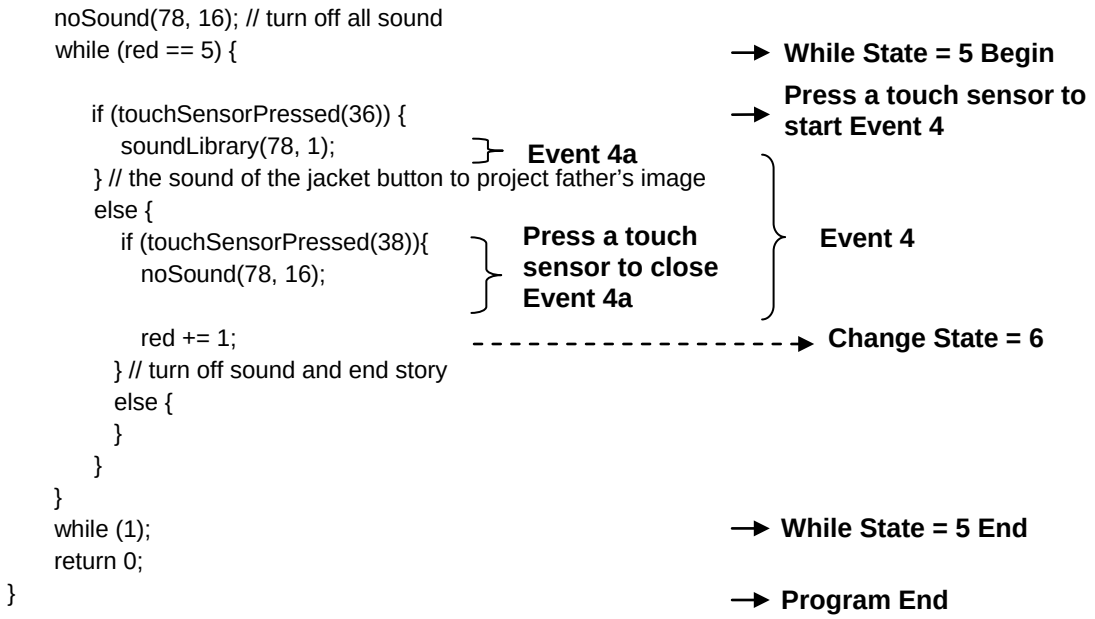
Author: A middle-school-mixed group (4 boys)

Story title: Three Sons

Program code:

```
#include <sensor.h>
int main(void) {
    init();
    int red = 1;
    Wire.begin();
    red = 1;

    while (red == 1) {

        if (touchSensorPressed (36)) {
          ledOn(112, RED);
          ledOn(114, RED);
          ledOn(118, RED);
          soundLibrary(78, 3);
        } // father was in great danger.
        else {
            if (touchSensorPressed (38)) {
              ledOff(112, RED);
              ledOff(114, RED);
              ledOff(118, RED);
              noSound(78, 16);

              red += 1;
            } // turn off all LEDs and sound
            else {
              soundLibrary(78, 1);
            } // father's death
        }
    } // end while-loop

    while (red <= 4) {

        if (touchSensorPressed(36)) {
            soundLibrary(78, 7);

            red += 1;
        } // stock was increasing.
        else {

            noSound(78, 16);
        } // turn off all sound
    } // end while-loop

    soundLibrary(78, 9);
    for (int i=0; i<3; i++) {
        soundLibrary(78, 8);
    } // stock was decreasing.
```





* Remarks: It is not a pure example of the program structure with using sequentiality on states with a switch trigger because: 1) States 3, 4 and 5 are used for repeating the sound effect rather than controlling the state to the next event; 2) Event 3 is not directly controlled by any state variable. However, the rest of the program is using this kind of program structure.

References

[1] Ali, A. and Shubra, C. Efforts to Reverse the Trend of Enrollment Decline in Computer Science Programs. 2010. In *Issues in Informing Science and Information Technology*, Volume 7, 2010, pp. 209-224.

[2] Alice. http://www.alice.org/

[3] Anscombe, I. 1996. *Arts and Crafts Style*. Phaidon Press.

[4] Antle, A.N. 2007. The CTI framework: informing the design of tangible systems for children. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (TEI '07). ACM, New York, NY, USA, 195-202.

[5] Arduino. http://www.arduino.cc/

[6] Arduino Softwware. http://www.arduino.cc/en/Main/software

[7] Amici. http://dimeb.informatik.uni-bremen.de/eduwear/category/development-software/

[8] Ackermann, E. 2004. Constructing knowledge and transforming the world. In *L. Tokoro M.; Steels (Ed.), A learning zone of one's own: Sharing representations and flow in collaborative learning environments (Vol. 1, pp. 15–37.)*. Amsterdam, Berlin. Oxford, Tokyo, Washington, DC: IOS Press 2004.

[9] Allen, S. 2004. Designs for Learning: Studying Science Museum Exhibits That Do More Than Entertain. *Science Education*, 88 (S1), Wiley Periodicals, S17-S33.

[10] Band, S. T., and Donato, J.M. 2001. *Storytelling in Emergent Literacy: Fostering Multiple Intelligences*. Delmar Thomson Learning, pp.27-45.

[11] Bandler, R., Grinder, J. 1979. *Frogs into Princes: Neuro Linguistic Programming*. Moab, UT: Real People Press.

[12] Benford, S., Rowland, D., Flintham, M., Drozd, A., Hull, R., Reid, J., Morrison, J. and Facer, K. 2005. Life on the edge: supporting collaboration in location-based experiences. In *Proceedings of the SIGCHI conference on*

Human factors in computing systems (CHI '05). ACM, New York, NY, USA, 721-730.

[13] Bonar, J. and Soloway, E. 1983. Uncovering principles of novice programming. In *Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages* (POPL '83). ACM, New York, NY, USA, 10-13.

[14] BrickLayer. http://etoy.comp.polyu.edu.hk/sites/default/files/bricklayer/index.htm.

[15] Brophy, J. 1983. Conceptualizing student motivation. *Educational Psychologist*, 18, 200-215.

[16] Bruner J. 1996. The culture of education. Harvard University Press, Cambridge, MA.

[17] Budd, J., Madej, K., and Stephens-Wells, J. et al. 2007. PageCraft: learning in context a tangible interactive storytelling platform to support early narrative development for young children. In *Proceedings of the 6th international conference on Interaction design and children* (IDC '07). ACM, New York, NY, USA, 97-100.

[18] Buechley, L., lumeze, N. and Eisenberg, M. 2006. Electronic/computational textiles and children's crafts. In *Proceedings of the 2006 conference on Interaction design and children* (IDC '06). ACM, New York, NY, USA, 49-56.

[19] Buechley, L., Eisenberg, M., and Elumeze, N. 2007. Towards a curriculum for electronic textiles in the high school classroom. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education* (ITiCSE '07). ACM, New York, NY, USA, 28-32.

[20] Buechley, L., Eisenberg, M., Catchen, J. and Crockett, A. 2008. The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (CHI '08). ACM, New York, NY, USA, 423-432.

[21] Buechley, L., Hendrix, S. and Eisenberg, M. 2009. Paints, paper, and programs: first steps toward the computational sketchbook. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction* (TEI '09). ACM, New York, NY, USA, 9-12.

[22] Butler, D., Strohecker, C. and Martin, F. 2006. Sustaining Local Identity, Control and Ownership While Integrating Technology into School Learning. In *Proc. ISSEP*, Springer, 2006, 4226, 255-266.

[23] Byckling, P and Sajaniemi, J. 2006. Roles of variables and programming skills improvement. *SIGCSE Bull*. 38, 1 (March 2006), 413-417.

[24] Carter, L. 2006. Why students with an apparent aptitude for computer science don't choose to major in computer science. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education* (SIGCSE '06). ACM, New York, NY, USA, 27-31.

[25] Carsten, B. 1989. *Power Conversion and Intelligent Motion magazine*, in a column “Carsten's Corner”, (p. 38) subtitled “Let's Define a Few Terms” November 1989.

[26] Cheung, J.C.Y, Ngai, G., Chan, S.C.F. and Lau, W.W.Y. 2009. Filling the gap in programming instruction: a text-enhanced graphical programming environment for junior high students. In *Proceedings of the 40th ACM technical symposium on Computer science education* (SIGCSE '09). ACM, New York, NY, USA, 276-280.

[27] Choi, Y., Pan, Y. and Jeung, J. 2007. A study on the emotion expression using lights in apparel types. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services* (MobileHCI '07). ACM, New York, NY, USA, 478-482.

[28] Chipman, G., Druin, A., Beer, D., Fails, J.A., Guha, M.L. and Simms, S. 2006. A case study of tangible flags: a collaborative technology to enhance field trips. In *Proceedings of the 2006 conference on Interaction design and children* (IDC '06). ACM, New York, NY, USA, 1-8.

[29] Clements, D. 1999. ‘Concrete’ Manipulatives, Concrete Ideas. *Contemporary Issues in Early Childhood*, 1, 1, 45-60.

[30] Cypher, A., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A. and Turransky A. (Eds.). 1993. *Watch what I do: Programming by Demonstration*. MIT Press, Cambridge, MA, USA.

[31] Di Blas, N., Paolini, P. and Sabiescu, A. 2010. Collective digital storytelling at school as a whole-class interaction. In *Proceedings of the 9th International Conference on Interaction Design and Children* (IDC '10). ACM, New York, NY, USA, 11-19.

[32] Dillenbourg, P. 1999. Collaborative Learning: Cognitive and Computational Approaches. *Advances in Learning and Instruction Series*. New York, NY: Elsevier Science, Inc.

[33] Dormer, P. 1997. *The Culture of Craft (Studies in Design and Material Culture)*. Manchester University Press.

[34] Dourish, P. 2001. *Where the Action Is: The Foundations of Embodied Interaction*. MIT Press, Cambridge, MA.

[35] Druin, A., Montemayor, J., and Hendler, J. et al. 1999. Designing PETS: a personal electronic teller of stories. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit* (CHI '99). ACM, New York, NY, USA, 326-329.

[36] Dunn, R. and Dunn, K. 1978. *Teaching students through their individual learning styles: A practical approach*. Reston, VA: Reston Publishing Company.

[37] Eisenberg, M. 2007. Pervasive Fabrication: Making Construction Ubiquitous in Education. In *Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops* (PERCOMW '07). IEEE Computer Society, Washington, DC, USA, 193-198.

[38] Eisenberg, M. 2003. Mindstuff: Educational Technology beyond the Computer. In *Convergence*, Summer 2003.

[39] Felder, R.M. and Silverman, L.K. 1988. Learning and Teaching Styles in Engineering Education. *Engineering Education*, Vol. 78, No. 7, pp. 674-681.

[40] Felder, R. M. and Spurlin, J. E. 2005. Applications, Reliablity, and Validity of the Index of Learning Styles. *Intl. Journal of Engineering Education*, 21(1), 103-112.

[41] Fernaeus, Y. and Tholander, J. 2006. Finding design qualities in a tangible programming space. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (CHI '06), Rebecca Grinter, Thomas Rodden, Paul Aoki, Ed Cutrell, Robin Jeffries, and Gary Olson (Eds.). ACM, New York, NY, USA, 447-456.

[42] Fernaeus, Y. and Tholander, J. 2006. Designing for Programming as Joint Performances among Groups of Children. *Interacting with Computers* 18, 1012–1031.

[43] Fisher, C., Berliner, D., Filby, N., Marliave, R., Cahen, L., & Dishaw, M. 1980. Teaching behaviours, academic learning time, and student achievement: An overview. In C. Denham & A. Lieberman (Eds.), *Time to Learn*. Washington, D.C.: National Institute of Education.

[44] Fleming, N.D. and Mills, C. 1992. Not Another Inventory, Rather a Catalyst for Reflection. *To Improve the Academy*, Vol. 11, page 137.

[45] Fontijn, W.F.J. and P. Mendels. 2005. StoryToy the Interactive Storytelling Toy. In *Proceedings of PerGames workshop*, Int. Conference on Pervasive Computing, Munich, Germany, 11 May 2005, pp.37-42.

[46] Frauenheim, E. Students Saying No to Computer Science. C | Net News.com, August 11, 2004, http://www.news.com/2100-1022-5306096.html

[47] Froebel Gifts. http://www.froebelgifts.com/

[48] Froebel, F. 1826. *On the Education of Man* (Die Menschenerziehung), Keilhau/Leipzig: Wienbrach.

[49] Funakoshi, K., Kobayashi, K., Nakano, M., Yamada, S., Kitamura, Y. and Tsujino, H. 2008. Smoothing human-robot speech interactions by using a blinking-light as subtle expression. In *Proceedings of the 10th international conference on Multimodal interfaces* (ICMI '08). ACM, New York, NY, USA, 293-296.

[50] Gardner, H. 1983. *Frames of mind: the theory of multiple intelligences*. New York: Basic Books.

[51] Glos, J.W. and Cassell, J. 1997. Rosebud: technological toys for storytelling. In *CHI '97 extended abstracts on Human factors in computing systems: looking to the future* (CHI '97). ACM, New York, NY, USA, 359-360.

[52] Goldman, L. R. 1998. *Child's play: Myth, mimesis, and make-believe*. New York, Berg Press.

[53] Gregorc, A. F. 1982. *An Adult's Guide to Style*. Maynard, Massachusetts: Gabriel Systems, Inc.

[54] Harrison, C., Tan, D. and Morris, D. 2010. Skinput: appropriating the body as an input surface. In *Proceedings of the 28th international conference on Human factors in computing systems* (CHI '10). ACM, New York, NY, USA, 453-462.

[55] Hashagen, A., Buecking, C., and Schelhowe, H. 2009. Learning abstract concepts through bodily engagement: a comparative, qualitative study. In *Proceedings of the 8th International Conference on Interaction Design and Children* (IDC '09). ACM, New York, NY, USA, 234-237.

[56] Horn, M. S. and Jacob, R. J. K. 2007. Tangible programming in the classroom with tern. In *Proceedings of CHI '07 extended abstracts* (San Jose, CA, April 2007). ACM Press, pp.1965-1970.

[57] Horn, M.S., Solovey, E.T. and Jacob, R.J.K. 2008. Tangible programming and informal science learning: making TUIs work for museums. In *Proceedings of the 7th international conference on Interaction design and children* (IDC '08). ACM, New York, NY, USA, 194-201.

[58] Horn, M.S., Solovey, E.T., Crouser, R.J., and Jacob, R.J.K. 2009. Comparing the use of tangible and graphical programming languages for informal science education. In *Proceedings of the 27th international conference on Human factors in computing systems* (CHI '09). ACM, New York, NY, USA, 975-984.

[59] Humphrey, T. and Gutwill, J.P. 2005. *Fostering Active Prolonged Engagement: The art of creating APE exhibits*. Exploratorium.

[60] i*CATch library for Arduino IDE
http://etoy.comp.polyu.edu.hk/sites/default/files/icatch/iCATchSensor.zip

[61] i*CATch hybrid text-graphical IDE
http://etoy.comp.polyu.edu.hk/sites/default/files/icatch/iCATch_v1.6.zip

[62] Jacob, R.J.K., Ishii, H., Pangaro, G. and Patten, J. 2002. A tangible interface for organizing information using a grid. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves* (CHI '02). ACM, New York, NY, USA, 339-346.

[63] Jung, C.G. 1971. *Psychological Types*. Princeton University Press, Princeton, N.J. (Originally published in 1921.)

[64] I2C Bus Specification.
http://ics.nxp.com/support/documents/interface/pdf/i2c.bus.specification.pdf

[65] I2C, A networking solution for integrated circuits.
http://www.nxp.com/documents/leaflet/75016900.pdf

[66] Instructables. http://www.instructables.com/.

[67] Ishii, H. and Ullmer, B. 1997. Tangible bits: Towards seamless interfaces between people, bits, and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '97), Steven Pemberton (Ed.). ACM, New York, NY, USA, 234-241.

[68] Johnson. G.W. 1997. *LabVIEW Graphical Programming: Practical Applications in Instrumentation and Control* (2nd ed.). McGraw-Hill School Education Group.

[69] Kalanithi, J.J. and Bove, V.M. Jr. 2008. Connectibles: tangible social networks. In *Proceedings of the 2nd international conference on Tangible and embedded interaction* (TEI '08). ACM, New York, NY, USA, 199-206.

[70] Katterfeldt, E., Dittert, N., and Schelhowe, H. 2009. EduWear: smart textiles as ways of relating computing technology to everyday life. In *Proceedings of the 8th International Conference on Interaction Design and Children* (IDC '09). ACM, New York, NY, USA, 9-17.

[71] Katz, L.G. and S.C. 1989. *Chard. Engaging Children' Minds: the Project Approach*. Ablex, Norwood, NJ.

[72] Kelleher, C. and Pausch, R. 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput. Surv.* 37, 2 (June 2005), 83-137.

[73] Kelleher, C., Pausch, R. and Kiesler, S. 2007. Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '07). ACM, New York, NY, USA, 1455-1464.

[74] Kolb, D.A. 1984. *Experiential Learning: Experience as the source of learning and development*. Englewood Cliffs, NJ: Prentice-Hall.

[75] Komatsu, T., Yamada, S., Kobayashi, K., Funakoshi, K. and Nakano, M. 2010. Artificial subtle expressions: intuitive notification methodology of artifacts. In *Proceedings of the 28th international conference on Human factors in computing systems* (CHI '10). ACM, New York, NY, USA, 1941-1944.

[76] Kuhn, S. 2001. Learning from the architecture studio: Implications for project-based pedagogy. *International Journal of Engineering Education*, 17 (4 & 5), 349–352.

[77] Lau, W.W.Y., Ngai, G., Chan, S.C.F., and Cheung, J.C.Y. 2009. Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students. In *Proceedings of the 40th ACM technical symposium on Computer science education* (SIGCSE '09). ACM, New York, NY, USA, 504-508.

[78] Lidwell, W., Holden, K. and Butler, J. 2003. *Universal Principles of Design: 100 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, and Teach Through Design.* Rockport Publishers, pp. 186 – 187.

[79] Lieberman, H., Paternò, F. and Wulf, V. 2006. *End User Development (Human-Computer Interaction Series).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[80] LilyPad Arduino. http://web.media.mit.edu/~leah/LilyPad/

[81] LilyPond. http://lilypond.media.mit.edu/about

[82] Livo, N.J., and Rietz, S.A.1986. *Storytelling: Process and Practice*, Littlejohn, Libraries Unlimited.

[83] LOGO Foundation. http://el.media.mit.edu/logo-foundation/logo/index.html

[84] Lovell, E. and Buechley, L. 2010. An e-sewing tutorial for DIY learning. In *Proceedings of the 9th International Conference on Interaction Design and Children* (IDC '10). ACM, New York, NY, USA, 230-233.

[85] Magielse, R. and Markopoulos, P. 2009. HeartBeat: an outdoor pervasive game for children. In *Proceedings of the 27th international conference on Human factors in computing systems* (CHI '09). ACM, New York, NY, USA, 2181-2184.

[86] Marshall, P. 2007. Do tangible interfaces enhance learning? In *Proceedings of the 1st international conference on Tangible and embedded interaction* (TEI '07). ACM, New York, NY, USA, 163-170.

[87] McCluskey, A.V. 2006. The difficulties of Swiss Computer Science studies. An example of the need for a new strategy. *SARIT conference*, March 13, 2006.

[88] McIntyre, D.J., Copenhaver, R.W., Byrd, D.M., & Norris, W.R. 1983. A study of engaged student behaviour within classroom activities during mathematics class. *Journal of Educational Research*, 77(1), 55-59.

[89] McNerney, T.S. 2004. From turtles to Tangible Programming Bricks: explorations in physical language design. *Personal Ubiquitous Comput*. 8, 5 (Sep. 2004), 326-33

[90] Mindstorms. http://media.mit.edu/sponsorship/getting-value/collaborations/mindstorms

[91] Montemayor, J., Druin, A., Chipman, G., Farber, A., Guha, M.L., 2004. Tools for children to create physical interactive storyrooms. *Comput. Entertain*. 2, 1 (January 2004), 12-12.

[92] Montessori Materials. http://www.montessoriedutoys.com

[93] Montessori, M 1912. *The Montessori Method*. New York Frederick Stokes Co.

[94] Ngai, G., Chan, S.C.F., Cheung, J.C.Y. and Lau, W.W.Y. 2009. The TeeBoard: an education-friendly construction platform for e-textiles and wearable computing. In *Proceedings of the 27th international conference on Human factors in computing systems* (CHI '09). ACM, New York, NY, USA, 249-258.

[95] Ngai, G., Chan, S.C.F., Ng, V.T.Y., Cheung, J.C.Y., Choy, S.S.S., Lau, W.W.Y. and Tse, J.T.P. 2010. i*CATch: a scalable plug-n-play wearable computing framework for novices and children. In *Proceedings of the 28th international conference on Human factors in computing systems* (CHI '10). ACM, New York, NY, USA, 443-452.

[96] Papert, S.1980. *Mindstorms: Children, computers and powerful ideas*. Basic Books, New York.

[97] Piaget, J. 1954. *The Construction of Reality in the Child*. Routledge and Kegan Paul Ltd. (Originally published in 1937.)

[98] Piaget, J. 1997. *The Principles of Genetic Epistemology*. Routledge and Kegan Paul Ltd. (Originally published in 1970.)

[99] Piaget, J. 1999. *Play, dreams and imitation in childhood*. Routledge and Kegan Paul Ltd. (Originally published in 1951.)

[100] PicoBoard. http://www.picocricket.com/picoboard.htm

[101] Portsmore, M. 1999. ROBOLAB: Intuitive Robotic Programming Software to Support Life Long Learning. *APPLE Learning Technology Review*, 26-39.

[102] Pratt, T.W. and Zelkowitz, M.V. 2000. *Programming Languages: Design and Implementation* (4th ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA.

[103] Price, S. 2008. A representation approach to conceptualizing tangible learning environments. In *Proceedings of the 2nd international conference on Tangible and embedded interaction* (TEI '08). ACM, New York, NY, USA, 151-158.

[104] Raffle, H., Parkes, A., Ishii, H., and Lifton, J. 2006. Beyond record and play: backpacks: tangible modulators for kinetic behavior. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (CHI '06),

Rebecca Grinter, Thomas Rodden, Paul Aoki, Ed Cutrell, Robin Jeffries, and Gary Olson (Eds.). ACM, New York, NY, USA, 681-690.

[105] Resnick, M. & Silverman, B. 2005. Some reflections on designing construction kits for kids. In *Proceedings of the 2005 conference on Interaction design and children* (IDC '05). ACM, New York, NY, USA, 117-122.

[106] Resnick, M., Maloney, J. et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (November 2009), 60-67.

[107] Rogers, Y., Price, S., Fitzpatrick, G., Fleck, R., Harris, E., Smith, H., et al. 2004. Ambient wood: designing new forms of digital augmentation for learning outdoors. In *Proceedings of the 2004 conference on Interaction design and children: building a community* (IDC '04). ACM, New York, NY, USA, 3-10.

[108] Rusk, N., Mitchel, R., Berg, R., Pezalla-Granlund, M. 2008. New Pathways into Robotics: Strategies for Broadening participation. *Journal of Science Education and Technology*, Volume 17, Issue 1, pp.59-69.

[109] Ryokai, K. and Cassell, J. 1999. StoryMat: a play space for collaborative storytelling. In *CHI '99 extended abstracts on Human factors in computing systems* (CHI '99). ACM, New York, NY, USA, 272-273.

[110] Schollmeyer, M. 1996. Computer programming in high school vs. college. In *Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education* (SIGCSE '96), Karl J. Klee (Ed.). ACM, New York, NY, USA, 378-382.

[111] Scratch. http://scratch.mit.edu/

[112] Shaer, O. and Hornecker, E. 2010. Tangible User Interfaces: Past, Present, and Future Directions. *Found. Trends Hum.-Comput. Interact*. 3, 1-2 (January 2010), 1-137.

[113] Skinner, E.A., & Belmont, M.J. 1993. Motivation in the classroom: Reciprocal effects of teacher behavior and student engagement across the school year. *Journal of Educational Psychology*, 85(4). p. 572.

[114] Snibbe, S. 2006. Three Drops. ArtNano: New Approaches for Visualizing the Nanoscale. http://www.nisenet.org/artnano/artists/snibbe/artwork/.

[115] Soler-Adillon, J., Ferrer, J. and Parés, N. 2009. A novel approach to interactive playgrounds: the interactive slide project. In *Proceedings of the 8th International Conference on Interaction Design and Children* (IDC '09). ACM, New York, NY, USA, 131-139.

[116] Sparacino, F., Wren, C., Davenport, G. and Pentland, A. 1999. Augmented Performance in Dance and Theater. *International Dance and Technology* 99 (IDAT99), Arizona State University, Tempe, AZ (February 25-28, 1999).

[117] Sparacino, F., Pentland, A. and Davenport, G. 1997. Wearable Performance. In *Proceedings of the 1st IEEE International Symposium on Wearable Computers* (ISWC '97). IEEE Computer Society, Washington, DC, USA, 181-182.

[118] Stanton, D., Bayon, V., Neale, H., Ghali, A., Benford, S., Cobb, S., Ingram, R., O'Malley, C., Wilson, J. and Pridmore, T. 2001. Classroom collaboration in the design of tangible interfaces for storytelling. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '01). ACM, New York, NY, USA, 482-489.

[119] Stross, R. 2008. What Has Driven Women Out of Computer Science? *The New York Times*.
http://www.nytimes.com/2008/11/16/business/16digi.html?_r=1

[120] Suzuki, H., Kato, H. 1995. Interaction-Level Support for Collaborative Learning: AlgoBlock-An Open Programming Language. *Proceedings of CSCL '95* (Bloomington, Indiana, October 1995), pp: 349 – 355.

[121] Tada, Y., Nishimoto, K., Maekawa, T., Rouve, R., Mase, K. and Nakatsu, R. 2001. Towards forming communities using wearable musical instruments. *Distributed Computing Systems Workshop, 2001 International Conference on* , vol., no., pp.260-265, Apr 2001

[122] Tarkan, S., Sazawal, V., Druin, A., Golub, E., Bonsignore, E.M., Walsh, G. and Atrash, Z. 2010. Toque: designing a cooking-based programming language for and with children. In *Proceedings of the 28th international*

conference on Human factors in computing systems (CHI '10). ACM, New York, NY, USA, 2417-2426.

[123] Tucker, A.B. and Noonan R.E. 2007. *Programming Languages Principles and Paradigms*, pp. 4-5. Second Ed.

[124] Torrance, E.P. 1984. *Torrance Tests of Creative Thinking. Norms*, Technical Manual Research Edition. Princeton, NJ: Personnel Press.

[125] Ullmer, B. and Ishii, H. 2000. Emerging frameworks for tangible user interfaces. *IBM Syst. J.* 39, 3-4 (July 2000), 915-931.

[126] Umaschi, M. 1997. Soft toys with computer hearts: building personal storytelling environments. In *CHI '97 extended abstracts on Human factors in computing systems: looking to the future* (CHI '97). ACM, New York, NY, USA, 20-21.

[127] Vegso, J. 2005. Interest in CS as a Major Drops Among Incoming Freshmen. *Computing Research News*, Vol. 17/No.3.

[128] Vygotsky, L.S. 1967. Play and its role in the mental development of the child. *Soviet Psychology*, 5(3), 6-18.

[129] Vygotsky, L.S. 1978. *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.

[130] Wardrip-Fruin, N. and Nick, M. 2003. From Theatre of the Oppressed. *The New Media Reader*. The MIT Press, pp.339-352.

[131] Wood, D.J., Bruner, J.S., & Ross, G. 1976. The role of tutoring in problem solving. *Journal of Child Psychiatry and Psychology*, 17(2), 89-100.

[132] Zhou, Z., Cheok, A.D., Li, Y. and Kato, H. 2005. Magic cubes for social and physical family entertainment. In *CHI '05 extended abstracts on Human factors in computing systems* (CHI '05). ACM, New York, NY, USA, 1156-1157.

[133] Zuckerman, O., Arida, S. and Resnick, M. 2005. Extending tangible interfaces for education: digital montessori-inspired manipulatives. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '05). ACM, New York, NY, USA, 859-868.

[134] Zuckerman, O., Grotzer, T. and Leahy, K. 2006. Flow blocks as a conceptual bridge between understanding the structure and behavior of a complex causal system. In *Proceedings of the 7th international conference on Learning sciences* (ICLS '06). International Society of the Learning Sciences 880-886.

[135] Zuckerman, O. and Resnick, M. 2004. Hands-on modeling and simulation of systems. In *Proceedings of the 2004 conference on Interaction design and children: building a community* (IDC '04). ACM, New York, NY, USA, 157-158.