Let’s Make a Digital Patchwork

Designing for Children’s Creative Play with Programming Materials

Ylva Fernaeus

Stockholm University
For the Children
List of Publications

Included Publications


Related Publications


All of the papers appended in this thesis, as well as the actual studies, were done in close collaboration with others. Most importantly, this work would have taken a completely different form without my close collaboration with Jakob Tholander, with whom I shared not only office and research projects, but also nearly all my thoughts with during the first three years of this project. Even though we are no longer at the same department, he is still the person that I usually turn to when I need advice in my work. An important part of our work included repeated workshops and activities with groups of children, whose enthusiasm, interests, and creative ideas constantly forced our research efforts into new directions. I have no ideas where this project had taken us without them. One of the papers is also the result of collaboration with Mikael Kindborg in Linköping, whose strong interest in representational forms for children’s programming has inspired me into new ways of thinking about computational structures. Other discussions that have formed much of this work include intense email conversations with Kevin McGee and Ken Kahn regarding conceptual as well as practical issues of children’s programming. Yet other pieces have been provided by a series of master students, as well as my colleagues Martin Jonsson, Jesper Holmberg and Johan Mattsson, who have been of invaluable help at crucial parts of the technology explorations. The very idea of a digital patchwork arouse as part of my collaborations with Ulla West, and our ongoing explorations around the themes of mixing checker patterns, pixel images, interaction, and textile technologies. I have got lots of good advice lately from Martin Jonsson, at the weirdest hours – thank him also for the stickers. Many thanks also to Jenny Hellström, who made the illustrations for the cover of this book.

This process had been impossible without the support from the senior researchers around me, perhaps especially those in my ‘supervisor committee’ (Jakob Tholander, Kia Höök, and Calle Jansson), and Jonas Löwgren for an excellent ‘predoc’ seminar. I’m also grateful to everyone who has given feedback on my writing, Magnus Boman, Klas Karlgren, Tessy Ceratto Pargman, Mikael Kindborg, and everyone else in the labs at SICS and at DSV who I have in different ways followed me in this work, Hillevi Sundholm, Martin Jonsson, Anna Ståhl, Jarmo Laaksolaati, Petra Sundström, Tony Lindgren, Fatima Jonsson, Johan Mattsson, Maria Croné, Martin Svensson, Mona Påhlman, Johan Eliasson, Lisa Browsers, Patric Dahlqvist, Karin Hansson, Peter Lönnkvist, Petter Karlström…
Thanks also to my family and my wonderful community of friends, for being with me during this sometimes quite difficult time. Thank you so much for places to stay, amazing cocktails, dinners, dance moves, parties, coffees, lunches, holidays, and so many great moments, evenings, days… Emma Rapp, Jenny Helström, Åskar Brickman, Madeleine and Nicklas Norling, Karl and Anne Grandin and the rest of Sveriges Kulturnorge - you are the best!
## Contents

Preface ........................................................................................................................................ 13

1 Introduction .......................................................................................................................... 15
  1.1 Problem .......................................................................................................................... 16
  1.2 Approach ....................................................................................................................... 18
  1.3 Contributions ................................................................................................................ 19
  1.4 Division of Work ........................................................................................................... 20
  1.5 Structure of thesis ......................................................................................................... 21

2 A Patchwork Approach to Children’s Programming ...................................................... 22
  2.1 Media-Sensitive Design ............................................................................................... 23
  2.2 Making a whole out of ‘pieces’ ................................................................................... 25
  2.3 Tangibles for Social Interaction .................................................................................. 27

3 Exploring the Design Space through Observation, Intervention and Tool Design ............... 30
  3.1 Overview of the Process .............................................................................................. 30
  3.2 Design of new behaviour structures ......................................................................... 35
  3.3 Children’s Design with Low-Fidelity Materials ......................................................... 38
  3.4 Role Plays with Groups of Children ........................................................................... 40
  3.5 Explorations of Tangible Technologies ..................................................................... 42

4 Interacting with Patcher .................................................................................................... 46
  4.1 Media-Sensitive Design .............................................................................................. 48
  4.2 Making a whole out of pieces .................................................................................... 49
  4.3 Tangibles for Social Interaction .................................................................................. 51

5 Conclusions ........................................................................................................................ 54
  5.1 The use of tools ............................................................................................................ 54
  5.2 Programming as creative play ..................................................................................... 55
  5.3 Understanding the Digital Material ............................................................................ 56

6 Summary of Papers ............................................................................................................. 58
   Paper I: Computational literacy at work ......................................................................... 58
   Paper II: “Looking At the Computer but Doing It On Land” ......................................... 59
   Paper III: Finding Design Qualities in a Tangible Programming Space ....................... 59
Preface

Using the word *Patchwork* in the title of a doctoral thesis might seem somewhat peculiar, since in science, a ‘patchwork’ is otherwise used to describe a construction, or theory, made up of inconsistent ideas, with parts combined in a seemingly thoughtless manner. However, rather than in such sense, the term patchwork is here intended to refer more literally to a personal interest in the specific affordances of textile materials and crafting.

Apart from this, my fundamental interest in interactive technologies comes partly from childhood. This resulted in university studies in computer and systems sciences at Stockholm University (as well as pre-academic arts and media studies in Bristol and in Sweden), ending with an MSc in interactive multimedia at University of Westminster. The specific interest in physical resources for interaction did however develop especially during two years work with conceptual design and programming of various interactive systems for museum contexts. In such work it becomes obvious how moving away from the standard input devices and to more broadly consider the physical setting is an important factor when designing interactive systems. This said, my understanding is still that the design of interactive properties of a system, i.e. those that define interactivity, always boils down to the practice of programming. Even though this can never be done without considering also the physical properties of the interactive resources, a lot of details will not get their final shape until they are formed and re-formed in the actual material of expression. This means that to me, programming is viewed as a highly creative and expressive activity, rather than only a means for learning, problem solving, or implementation of design ideas that are ‘already there’.

A constant theme throughout my time as a research student has also been a struggle with epistemological conflicts, with divergent perspectives that somehow had to fit together. Examples include the use ethnomethodological theories, emphasising what is strictly observable in recorded data, and perspectives that emphasise personal and sensory experience. This also includes general challenges of merging ‘theory’ with ‘practice’, academic writing with playful activities with children, and attempts at combining research with the design of new artefacts. As focusing specifically on children’s interaction, there has been a constant tension between educational values, the values expressed by the children, and my own more specific interest in the actual form of interaction. This became especially evident dur-
ing the first three years of this work, which was performed in the context of two research projects running in parallel. The first project, SPLÄSH, a collaboration between my research ‘group’ (Jakob and me) at DSV and Stockholm International Toy Research Center (SITREC), had a focus on activities where children design, modify and play with their own and each others’ computer games. This was done from a rather open perspective, and aimed at exploring how children made use of programming and digital construction tools and how such activities could be taken into practice in formal as well as informal situations. The second project, WebLabs, was a project that investigated new ways for middle-school children from six European countries to collaboratively work with scientific ideas. The intention was to study how tools and activities could be designed to let young students express and communicate their own ideas in a dynamic and interactive way. Among the different science domain that the project worked with, our team focused on the domain of natural ecosystems, and how ideas around such phenomena could be expressed on the computer. While my primary interest was more in line with the first project, several of our studies and reports became strongly influenced by the ambitions of the second project. With the way I have chosen to summarise my work I hope that this standpoint is made clear.

Figure 1. A kind of ‘patchwork’ made from cut-outs of glossy magazines (made by my mum Ingegerd, some years ago).
1 Introduction

The core ideas that I will put forward in this thesis are framed around a perspective of children’s programming where it is explored as a craft in a similar sense as other materials are traditionally introduced in schools and play schools. A quite substantial part in those practices concerns getting familiar with techniques for creating with materials such as wood, textile, pearls, paper, colour, and clay. Central to such activities are how to make use of media-specific qualities of the material with which one create.

My work has especially concerned the forms of visual expression afforded by computer screens and other digital display surfaces. The dynamic and interactive properties that are specific to this media do in many ways attract and interest children, and is also relevant since the computer ‘screen’ is often tightly associated with ‘the computer’, especially among children. Much can happen on a computer screen that is not possible to achieve with physical materials; things can be made to animate, objects may change shape and size, they may be copied, deleted, modifications can be undone, and much more. Moreover, in the context of writing up this thesis, computing technology (including mobile phones, video game consoles, online applications) are an integral part of many people’s everyday social activities, and the specific properties of expressions that are allowed by such media, for instance online sharing of information, rich possibilities for animation, and interaction, are also extensively and increasingly used for educational purposes. In that sense, screen-based interactive systems are quite a substantial part of our culture, and a basic assumption is that many children, just as anyone else, want to participate in the production of cultural artefacts. It also means that computational media may be seen as implying a special form of literacy (see e.g. diSessa, 2000; Snyder, 2002), concerning not only the ability to understand the work of others, but also to be able to create ones own media expressions.

These can be seen as some of the core motivations behind a stream of research efforts that concern the design of tools, materials and activities that aims at supporting children in making their own computer programs.
1.1 Problem

An aspect that is commonly discussed when engaging children in design is how well a certain material affords constructive and creative activities (see e.g. Eisenberg et al., 2002; Roth, 1996). If you put traditional Lego bricks in the hands of children they often immediately start building imaginative constructions, whereas if you provide them with a powerful programming environment, such as Logo or ToonTalk, there is a higher threshold before they start to build anything. Moreover, children who design their own computer systems often develop rich and elaborate ideas for what to build, but utilise only a subset of the expressive power of the programming resources when it comes to actually implement the ideas on the computer (Rader et al., 1997). An aspect that cannot be disregarded in this case is children’s prior expectations of what computers can do. Naturally, there is often a gap between the children’s initial ideas and what can be realised with a particular programming tool, especially since all programming environments targeted at children are designed to be easy to learn and to use, often at the cost of computational expressiveness (Ioannidou et al., 2003). In settings where children do create with computers, it is still more common to make static and sequential media, such as videos, animations, graphic designs or music, than to engage in programming.

Moreover, it is commonly noted how children when attempting to play with standard computer technology need to structure the interaction around turn taking or through some other division of labour (see Figure 2), often leading to frustrated situations where only one child can be actively participating at a time (Druin et al., 1997). In designing technology for children, an aspect that has been increasingly emphasised in recent years is therefore the importance of developing resources that may blend into everyday play patterns as they traditionally take shape in environments like schools and play schools (Crook, 1997; Druin, 1999). Important properties of such practices are the use of physical toys and other artefacts, which are practically endlessly open-ended in how they may be combined and used (see e.g. Wyeth, 2006). Other aspects are how physical materials, such as modelling clay, sand, snow, lego bricks, and textiles, all allow for action and interaction to be performed concurrently, and that physical manipulation can be conducted jointly as well as individually. Moreover, physical artefacts can be brought to and used in a range of different settings and activities. This poses a general challenge for interaction design, since the conventional ‘laptop’, ‘desktop’ or ‘personal computer’ is ultimately designed for individual activity.

An important focus in the area of children’s programming is thus to explore how programming structures and operations can be represented in tangible form (McNerny, 2004; Raffle et al., 2004). Physical manipulation of digital systems has been further elaborated on in the more general areas of tangible interaction and industrial design, where the rich sensory experience
afforded by physical objects are often emphasised (Buur et al., 2004). In the design of technology for children’s crafts, this has been especially prominent in work that concerns digitally enhanced construction kits and play materials (see e.g. Eisenberg et al., 2002; Raffle et al., 2004; Zuckerman et al., 2005). Other interesting work that explores the relationship between digital and physical crafting is the area of Computational Textiles (Buechley et al., 2006) where a focus has been on electronically enhanced fabrics, and different ways of adding interactivity to textile materials. A growing area in interactive technologies for children is concerned with technologies such as physical storytelling spaces (Montemayor et al., 2002), room-based (Eriksson, 2006; Stanton et al., 2001), and out-door interactive environments (Benford et al., 2005; Rogers et al., 2004). These technologies more extensively consider how to design for and make use of the physical space, and how this can support collaborative activities. However, only a subset of all approaches to tangible programming (e.g. Suzuki & Kato, 1995) concerns tools to support social interaction with on-screen materials.

Another problem that children have when making their own computer systems is the relationship between program creation and program execution (Repenning & Perrone, 2000; Smith & Cypher, 1999; Tholander, 2003). A conceptual difficulty is that the systems that children like to build, like richly animated games, often can be described at multiple ‘levels’, e.g., algorithms, observable behaviours, visible objects, instructions for use, and so on. To understand how the game one intends to make, one needs to grasp the differ-
ence between these levels and how they are related. The challenge is to create environments that support design of interesting behaviours through less formal and abstract programming practices.

1.2 Approach

In recent years, researchers seem to have shifted from looking at computer programming as something that by nature is theoretical, abstract, and primarily individual, to instead place more attention on the concrete and social aspects of how computer programs behave and come into existence. Examples include pair programming, tangible programming tools, and methods that attempt to more closely integrate design aspects into the programming process. Moreover the actual programs that are the result of the programming processes become increasingly multimodal, dynamic, and rich in how one may engage socially, aesthetically and emotionally with and through the systems (see e.g. McCarthy & Wright, 2004). A result of all this is that programming as such can no longer qualify as an activity characterised primarily of analytical work, but as a concrete craft of making artefacts that are perceived through our senses. From this standpoint my work is founded more generally on a socio-cultural perspective of human computer action, as exemplified for instance in activity theory (see e.g. Nardi, 1996).

As a way of framing my work, I make use of the use the folklore traditions of making textile patchworks, which is the traditional craft that I find most similar to the kind of programming that I have explored. There are several aspects of traditional patchwork technology that I find intriguing and fascinating, but as with all metaphors, only a few aspects will be used here to make a parallel to the activity of programming. There are especially three properties that I will be concerned with in the context of this thesis.

First, making a patchwork, whether it is created by paper, plastics or textile, is about being creative within a specific material, and within a specific form of cultural expression. The making of textile patchworks is well known in almost all human cultures and is also commonly practiced, for instance as a collaborative activity in schools. A digital patchwork in this case means switching from creating a physical artefact to design dynamic and interactive patterns on a digital display. Properties that are particular for this media include the possibility of making objects that change state, such as size and colour, to define how objects should move and interact with other objects, and to design for user interaction. The differences between the physical and the digital are then looked upon as being media or materials that can be shaped or moulded in different ways (see discussion in e.g. Landin, 2005).

Second, making a textile patchwork means combining pieces of different kinds of fabric into larger designs, and in that sense produce something new and valuable without having to buy or make all the resources needed to
weave new fabrics. Transferred to the activity of programming, this concerns a form of program construction based on reuse and altering of existing constructs, working programs and templates. This idea of ‘making a whole out of pieces’, also refers to the material, or constituent elements of what is being put together. In somewhat the same sense as a seamstress interacts with the materials of fabric, a programmer may need to deal with a range of different ‘programming materials’ in his or her practice. One could say that just as there are special textiles, cloth and canvases for embroidery and sewing, materials for quilting and stuffing, yarn of various kinds, pearls, feathers, buttons, and so on, there are many different kinds of materials that could be prepared for children to make their own computer programs.

Third, central to the making textile patchworks is, like with most other crafts, its character of physical manipulation. This not only means that there is a direct bodily connection to the material one is working with, but also that it affords two-handed interaction and that a group of people are able to work collectively on a shared project. In programming, this involves a shift from viewing the interactions as mostly cognitively based, to instead emphasise aspects of social and physical engagement with technology.

These three properties – the media-sensitive design, the ‘making of a whole out of pieces’, the social and physical manipulation – form the basis of the approach to children’s programming that I have chosen to refer to as a ‘digital patchwork’.

1.3 Contributions

Based on the perspective of children’s programming as outlined above, my work could be summarised through three interrelated research contributions:

1. Conceptualisations of children’s programming through the three themes of a digital patchwork, emphasising aspects of media-sensitive expression, concrete construction with higher-level programming pieces, and tangibles for social interaction. This also includes studies of how different resources alter, scaffold and support children in the activity of programming. A specific outcome concerns the importance of also considering ‘offline’, and socially oriented action when designing for children’s programming.

2. Reflections on the research practice. Through extensively mixing technical development with staged activities with users, we attempted to keep the interactive properties of programming constructs in focus. Core to our process was to find a balance between the specific qualities of physical artefacts and what can be displayed, and what is expected, in a digital format. Important in this process was for instance
how giving programming elements tangible form is not just an alternative form of representation, its actual function must be explored in a context of use.

3. The design of a new programming system, Patcher, with which groups of children can program pictures displayed on a large screen surface. The hardware of the system takes the shape of a digitally enhanced physical mat upon which resources in the form of plastic cards and blocks are used. The technical platform is based on a software system in combination with RFID and Bluetooth technology. Analyses of children’s interaction with this system show how the tangibles turn programming into a more physical, social and collaborative activity. More specifically, the physicality of the resources seems to work to join the activities of discussing, designing and programming.

By exploring the possibilities of blending properties of physical and digital crafting, my hope is to contribute to further developments of tools and materials that can more readily be used for creative work with computing technology, such as programming and interaction design at large.

1.4 Division of Work

The design process leading up to the Patcher system was a truly collaborative one, initially based on workshops, conducted by Jakob Tholander and me, using various low-fidelity materials and prototypes, as parts of our PC-based programming activities with children. The first set of example systems and libraries of programming behaviours were designed and programmed by me and Jakob using ToonTalk, based on the programming model of ‘Anima Gadgets’, already developed in the Playground project (Hoyles et al., 2002).

The first experiment at building a working tangible prototype was performed by Jesper Holmberg and me during a course in tangible sensor technology in 2003. I also presented this prototype to Ken Kahn and Augusto Chiocciariello, with whom there were extensive discussions in the months that followed, regarding various alternative technical solutions. The first working tangible prototype that was tested with children was built in Java by master student Christopher Balanikas, supervised collectively by Martin Jonsson, Johan Mattson, Jakob Tholander and me. The latest version of Patcher was reprogrammed from scratch by me, and the final making of the loose physical resources (printing, cutting, laminating cards, measuring, gluing, and sewing the mat) was also done by me.

All the video analyses were done in a highly collaborative manner by me and Jakob. We often had a quite advanced setup in our office with two large screen projections in opposite directions, with one displaying the video and
the other the transcription, controlled from our separate laptops. We then took turns controlling the video and taking notes, during constant discussions of what we observed in the data. Collaborative work towards a large screen projection was also a common setup when writing papers and reports. Some sessions were however conducted in a more distributed fashion.

The last studies of the Patcher platform were performed at the Museum of Natural History in Stockholm, by the staff at the museum. I made a few visits to the museum during the time, but in order to leave the activity in the hands of the users, we instead let master students observe these activities.

1.5 Structure of thesis

Chapter 2 aims at providing an overview of my understanding of how children’s programming could be approached as a real and relevant activity, based on the perspective of a digital patchwork. Chapter 3 provides an overview of my research process, interwoven with the methods that I have used. Chapter 4 summarises the design of Patcher and how the specificities of its design refer back to the ideas discussed in chapter 2. Chapter 5 provides some meta-reflections on my results, and Chapter 6 includes summaries of the appended papers.

Figure 3. Technical setup of the system.
2 A Patchwork Approach to Children’s Programming

Programming tools particularly targeted at children have been developed since the late sixties, early work starting with Logo (Papert, 1980) and Smalltalk (Kay & Goldberg, 1977) and more recently in research in visual, and tangible programming environments (Eisenberg et al., 2002; Kelleher & Pausch, 2005; Montemayor et al., 2002; Smith & Cypher, 1999). A substantial amount of research has also investigated different aspects of how activities can be designed for engaging children in productive use of such tools, for instance as creators of science simulations (Ioannidou et al., 2003), as learners of mathematics (diSessa, 1997; Papert, 1980), as game programmers (Kafai, 1995; Robertson & Good, 2004; Tholander et al., 2002), as software designers (Kafai & Ching, 2001), and as design partners when developing new technology for children (Druin & Fast, 2002). Programming activities have also been introduced to limited extents in computer clubs and other environments where children and young teens use computers for creative and artistic purposes. Other activities that are popular among children and that have similarities to programming include ‘God games’, and other software that allow users to design and specify properties of computational characters and play worlds.

Based on the perspective of children’s programming as a children’s craft, this work takes inspiration from conventional patchwork activities. This is summarised through three interrelated themes. The first theme concerns the personal and dynamic property of programming materials, letting children create content that is sensitive to the media of the computer screen. The second theme is to explore how programming could be made available to children by instead of forcing them to ‘weave the fabric’ of new programs, let them create by combining and altering existing high-level programming constructs. A third theme concerns explorations of how children could be supported in making shared digital artefacts in a fashion that harmonises with everyday social, physical, and creative play practice. These three themes will be referred to as media-sensitive design, making a whole out of pieces, and tangibles for social interaction.
2.1 Media-Sensitive Design

A common way of defining computer programming is as the creation of storable instructions that may be interpreted and executed by a digital device (see e.g. Blackwell, 2002; Hoc & Nguyen-Xuan, 1990; Kelleher & Pausch, 2005). Such instructions may be represented as series of 1s and 0s, or as some higher level form of representation. Early programs did for instance take the form of holes in physical punch cards, while more recent formats include textual codes, graphical rewrite rules (Canfield Smith et al., 2001) or even animated (Ken Kahn, 1996) and tangible (Horn & Jacob, 2007) program representations. Other definitions focus more specifically on algorithmic structures, it has for instance been shown how formal structures of textual code are central to how people understand and define the activity of programming. This is also well reflected in handbooks and course material on programming basics, as well as in how children have described what they think computer programming may be like (Sheehan, 2003).

Note that with these kinds of definitions, computer programming could be interpreted as an activity where the main purpose is to produce ‘code’. Even if this may sometimes be the case, it is important to keep in mind that as computer programs have developed from text based, through graphical, towards systems based on physical and even multimodal interaction, the core qualities of new computer programs depend to large extents on the user’s experience of interaction, which can not be easily captured only through specifications in code. The ultimate goal of computer programming, may then be understood not as the actual instructions, but as a working computational system.

In the design of programming tools for children, the relationship between usability and expressivity is a topic that is often discussed (Ioannidou et al., 2003; K. Kahn, 1999; Kelleher et al., 2007; Papert, 1980). To increase the usability of the tools, most programming environments for children are defined as ‘domain-specific’, meaning that only certain kinds of programs can be created. This naturally limits the expressivity of the tools, however, and contrary to the often mentioned notion of ‘general purpose programming languages’, most existing programming tools have been designed for more or less specific purposes, always following the shifting requirements and abilities of people and technology. An important purpose of developing new tools for programming has always been to find more efficient ways of bridging what people want to express in the computational media with instructions that can be interpreted in the hardware of a digital device.

An early milestone in the field of children’s programming and an important source of inspiration in my work is a report published more than thirty years ago, entitled Personal Dynamic Media, by Alan Kay and Adele Goldberg (Kay & Goldberg, 1977). The core of the paper concerns how Smalltalk, an early object-oriented programming system, was taken into use by
middle school children to create their own pictures, animations, and pieces of music. By referring to the programming environment as a “communication system” (p 31), and more generally to the technology as *Media*, the potentials of computers for expressing and sharing ideas were brought into focus, as opposed to the then dominating view of computation as useful primarily for automating calculations, solving complex mathematical problems, or for assisting in other work- and efficiency-oriented tasks. As part of their argumentation, the authors stated that for a technology to become really successful, it has to become attractive and usable even by children, which naturally places high demands on robustness, bandwidth and ease of use. In that sense this ambition requires the technological development to be pushed forward into new directions. Grounded on this very basic idea, the work of Kay and Goldberg has been quite influential also to the broader field of computer science, including for instance an early description of overlapping windows, dynamic use of different fonts, a pointing device, as well as a proposal for what would later become the laptop computer.

Today, many of the visions discussed by Kay and Goldberg in the seventies have become commonplace, children do to large extents use computers for creative purposes and there are much powerful and easy-to-use software for creating and communicating pictures, music, interactive homepages and animations. One could say that time have taken us from a world where almost all computer users were programmers, to a society where everyone, even the programmers, are end users of software. However, development of new programming environments specifically targeted at children is still an active area of research and many research projects are concerned with children as developers of software.

What I find especially exciting in the work of Kay and Goldberg is that by treating the work of the children as *media*, the main purpose of the activity is to express and communicate ideas, from one person to another or back to oneself. With this, the programs are also considered cultural objects, emerged from and meaningful to share in a social context. The term *Personal Dynamic Media* also captures two more qualities that have guided much of my thinking about children’s programming. However, instead of ‘personal’ as pointing to the physical artefact of a personal computer, I prefer to understand this in the sense that the activity is *personally engaging* for the participants and that the outcome of the activity holds some sense of uniqueness. An important feature of the screen-based media is for instance that you easily can display any visuals that you like, adding a highly personal ‘touch’ to the expressions. With *dynamic*, I refer to properties that are specific to computer programs, i.e. that they may be active, interactive, and behave differently each time they run, characterized by continuous change, activity, or progress. This include animated simulations, computer games, interactive fantasy worlds and certain digital art, but exclude modelling and scripting activities that may only result in static pictures, sequential anima-
tions and hypermedia. My interest is therefore on qualities that are specific to programming and interaction design, and how sensitivity to such qualities may be supported when children make their own computer systems.

2.2 Making a whole out of ‘pieces’

In educational research the term scaffolding is often used to refer to the kind of support that help learners engage in activities that otherwise would be out of their reach. This support may come in many forms ranging from specifically designed software that help structure the work (Guzdial & Kehoe, 1998), to letting students take on different roles such as presenter, critic, and manager throughout the learning processs (Ching, 2002; Kolodner et al., 1998). Another approach is to design microworlds or sets of readily prepared program constructs for children to explore and play with.

A main concern in the area of children’s programming has been to develop representations that children may easily understand, and with which they may express themselves and their ideas through. Because children with no prior experience of programming often have difficulties connecting lower level constructs to their higher level game ideas, programming is often introduced by using pre-built systems that could be explored and expanded in various ways, instead of starting by defining ones own algorithms from scratch (see e.g. Tholander, 2002). Another theme is how programming can be performed on different ‘conceptual levels’, for instance to make programming representations that are closer to the perceived properties of ‘the program’ than are more traditional programming models. Examples are programming using comic strip metaphors (Kindborg, 2003), visual before-after rules (Canfield Smith et al., 2001), and environments with simple sets of ready-made programming components (Kay, 2002). Other approaches are to use concrete virtual objects (Tholander et al., 2002) and real world tangible program constructs (Eisenberg et al., 2002; McNerny, 2004; Zuckerman & Resnick, 2003), attempting to make programming constructs more concrete and easier to manipulate.

Much of my work was set in ToonTalk, an animated programming environment targeted at children. This was partly because ToonTalk holds some interesting properties of interaction, and also since one of the projects that we worked in, WebLabs, was relying on this environment in the studies. In ToonTalk the ‘normal’ mode of programming consist of robots, which are trained to perform simple manipulations, for instance changing the numeric value of the position and size of a picture. A robot may control various aspects of a picture’s behaviour on the screen, for instance how it moves and what will happen if it collides with another picture. To make a robot control a picture, it is first trained to do what it should, and is then placed on the picture’s back, or ‘flipside’, where also comments and visual descriptions of
the robot’s behaviour can be added. There is also a higher level mode of programming with robots already trained, and by combining such robots more complex programming actions can be defined. A robot can be transferred between pictures by copying or moving a robot from the back of one picture to the back of another. There is also a form of inheritance where the entire behaviour of one picture may be transferred to another by placing that whole picture at the back of another. Libraries of such pictures with behaviours are packaged as sets of running examples called Anima Gadgets (for game programming) and Animal Gadgets (for making behaviours related to living organisms, e.g. eating, starving, growing, and reproducing). Even though the expressiveness of these two libraries is limited, this style of programming has proven successful even for children as young as seven years old allowing for creation of games and similar systems (Tholander et al., 2002).

Similarly, in many popular methods for software development, including object-oriented approaches, extreme programming, and open source methodologies, reuse of program code is a central aspect. This means that the task of the programmer is not only to define and formulate new algorithms, but also to locate and identify readily functioning object classes, code snippets or even full programs that can be altered and reused. Other ways of allowing for higher level styles of programming include the use of class libraries, interface widgets, readily prepared media elements, and so on. This theme is sometimes referred to as ‘postmodern programming’ (Noble & Biddle, 2002).

This relates to a central aspect of conventional patchwork techniques, i.e., the principle of reusing pieces from other designs. An important difference from textile patchworks is however that in programming, any piece from one design does not necessarily fit with any piece from any other design. This means that an important part in allowing for a patchwork model of programming is to provide a set of higher-level constructs, or ‘pieces’, that can be combined in a variety of ways. (This is not to be mixed up with ‘software patches’, which are small programs used for correcting bugs in existing software).

The idea of programming by taking what is found and doing something out of that has many similarities to the ideas of ‘Bricolage’ for children’s programming, as initially conceptualised by Turkle and Papert (1990). However, bricolage in French essentially means ‘tinkering’, and has much more to do with manly hobbyist activities, such as tinkering with a motorbike, than with creative and collaborative activities such as soft textile crafting (see discussion in Beckwith et al., 2006). The choice of metaphor thereby implies a focus on what the programming pieces might be, as well as what for and for whom they could be useful. Rather than a piece of hobbyist engineering, a patchwork has a connotation of a decorative artefact made out of pieces in a specific form of media.
2.3 Tangibles for Social Interaction

Much of the current discourse in human-computer interaction is based on studies of people’s actual activities in more or less naturalistic settings, for instance in the form of ethnomethodological accounts of meaning making practices with and around new technology. Examples of such work include Lucy Suchman’s analyses of interactions among office workers (Suchman, 1987), the investigations performed by Heath and Luff on control room interaction (Heath & Luff, 2000), and Jonas Ivarsson’s studies of children’s use of talk and gesture when interacting in computer settings (Ivarsson, 2001). Results from such studies have repeatedly indicated that knowing and doing are closely intertwined with properties in the physical and social circumstances of a situation. In human-computer interaction, these views are conceptualised in terms such as ‘situated action’ (Suchman, 1987), ‘distributed cognition’ (Hollan et al., 2000), and ‘embodied interaction’ (Dourish, 2001). Central to all these conceptions are that they seek to avoid simplified, dualistic perspectives on human action, such as distinguishing bodily actions from cognitive ones, or regarding interaction as a simple matter of ‘input’ and ‘output’. A common theme is for instance to emphasise how knowledge and decision-making may be supported by, and even off-loaded, to resources in the environment. An example of this is the graphic user interface, where users to a large extent can rely on recognition and visual search for possible action, rather than relying only on personal memory (Norman, 1993). More recent examples include conceptualisations around systems that can be interacted with through talk, gesture or through physical manipulation (Hornecker & Buur, 2006; Klemmer et al., 2006). This theme has been especially prominent in interactive technologies targeted at children, reflected for instance in systems that allow for several children to actively participate simultaneously (Africano et al., 2004; Druin et al., 1997), physical toys as interaction devices (Paiva et al., 2003), and systems that are controlled by gesture and voice (Cassell & Ryokai, 2001; Höysniemi et al., 2005). All these systems are based on the same principle that physical and social activity is important and desirable in children’s interaction with technology.

The qualities of these more recent developments of computing technology are sometimes referred to as embodied aspects of interaction (Dourish, 2001). All human action is of course ‘embodied’ in some sense, however, in explicitly pointing at bodily action this works as a contrast to action and activity that is described as less physically engaging and visible to others, such as performing a calculation individually in one’s mind. This also provides an alternative way of looking at interactive processes, emphasising aspects such as social and physical performance with and around technology (Jacucci, 2004).

Naturally, physical aspects are central in most argumentations for tangible approaches to programming, however the designs have in those cases pri-
marily addressed the needs of individual users, and the cognitive benefits from working hands on with physical programming objects (see e.g. McNerney, 2000; Patten et al., 2000). Even though those aspects are of much relevance, especially since programming is known to be a cognitively complex activity, further emphasis could be placed onto how users may account for ones actions in a group, and to act socially around the resources.

This is relevant to point out since tangibles are sometimes conceptualised as working as ‘input’ and ‘output’ devices at the same time (see e.g. Ullmer & Ishii, 2001). However, such conceptions can be difficult to follow as notions from an information-processing paradigm get replaced by descriptions of collaborative and bodily forms of interaction. The idea of tangibles as resources for input and output also breaks down since the physical parts of a tangible system often come to replace parts that would otherwise be displayed on a screen, and also that arrangements in the physical space are sometimes the only way that the system shows its actual ‘state’. In my work, the values of tangible interaction are instead understood through notions of human action, rather than on representation and transformation of information. This shift in perspective has been a general theme of my whole research process, and is as I see it central to understanding programming as a craft. A seamstress would for instance never describe her practice as information being sent back and forth.
As in most HCI research, an important part of my work has been to engage in the process of developing new computer systems and novel interaction possibilities. Core parts in such processes are practice-based activities, such as sketching, prototyping, programming, testing, and making revisions to earlier designs. Moreover, since the ‘interactive’ properties of a computational system only are present in the context of use, research in this area usually needs to combine a range of activities including not only the design and evaluation of systems, but also investigations into how the actual interaction takes form.

As an effect of this, my research process has been a quite explorative one, combining design, programming, user studies and analysis, often in parallel. By necessity, the description of this process will be a simplified one. In this chapter I will start by giving an overview of the process as a whole, followed by descriptions of what I see as core activities in the more practical work:

1. Design of new behaviour structures
2. Workshops where children design with low-fidelity materials
3. Role plays with groups of children
4. Hands-on explorations of tangible technologies in our lab

3.1 Overview of the Process

This work is based on qualitative analyses of design prototypes and of staged activities with children using these. The process of analysis has thus been a continuous activity occurring alongside the empirical studies and the design of resources. This is in many ways similar to participatory design approaches to research, which in recent years have become dominant in studies within the field of interaction design and children (Druin & Fast, 2002). However, instead of involving the children explicitly as informers to the design process, our approach to user involvement was to engage children in productive use of technology. From critical reflection of patterns of use and of the artefacts developed, we have further developed and refined our designs.

The process did thereby not follow a linear structure of problem, data collection and analyses. Instead it has more similarities to how design work is
commonly conceptualised, involving issues such as 'conversation with materials', 'reflection-in/on-action', and a shifting relationship between 'problem' and 'solution' (Schön, 1983). In such work the process is brought forward through sketches and prototypes, and there is a constant reflection on mishaps, possibilities, limitations, and unforeseen problems that arise along the way, requiring repeated revisions of original ideas. For instance, focus in our user studies was not so much to perform formal evaluations of the designs, instead, we used them more as a resources for developing ideas for interaction.

During the first three years of my thesis work we (Jakob Tholander and I) worked in close collaboration with two school classes in a local school (4-6 grade), and towards the end we also performed activities with children in an art gallery setting and at the Museum of Natural History in Stockholm. When introducing the activities to the children we presented ourselves as researchers who wanted to know how to make better computer programs for children, and that we wanted their help. The children were usually very enthusiastic about this task, and for all of the studies more children wanted to participate than we had space for. To select who to participate in each study we drew lots, and to reduce the risk of gender-related bias we strived for an equal number of boys and girls.

One could sort these activities into several study periods, where different sets of resources were taken into use. Apart from taking place at different periods in time, with different children, different tools, and at different stages in our research process, the studies were conceptualised around different themes (Vikings, Games, Endangered Animals, Randomness, Fantasy worlds, and Inventions). The study periods also had slightly different purposes. The main objective with the first activities was to introduce ourselves and the programming tools to the children and their teachers, and to prepare for following activities. In the second period, we wanted all the activities to evolve around what the children themselves wanted to produce with the tools. The goal of the third, forth and fifth periods was to provide a more structured framework for building dynamic systems and for the Swedish children that we worked with to exchange and communicate their projects with peers from other European countries. The final studies concerned children’s interaction with tangible prototypes and resources. We also made numerous informal visits to the school and held meetings with parents, teachers, and museum staff.

<table>
<thead>
<tr>
<th>Table 1. Overview of the different settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ages</td>
</tr>
</tbody>
</table>

31
<table>
<thead>
<tr>
<th>Time</th>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002/2003</td>
<td>ToonTalk</td>
<td>In the back of the classroom during arts class with two children at a time, modifying a simple game run on a PC. Two full classes, 40 children.</td>
</tr>
<tr>
<td>2002/2003</td>
<td>ToonTalk Robot Role Play Lego Mindstorms</td>
<td>Weekly sessions in the classroom, 8 children after regular school hours with, making their own games individually using ToonTalk.</td>
</tr>
<tr>
<td>2003/2004</td>
<td>ToonTalk Robot Role Play</td>
<td>During autumn break, 8 children working in pairs making games and simulations using ToonTalk in the lab environment iLounge.</td>
</tr>
<tr>
<td>2003/2004</td>
<td>ToonTalk Tangible Role Play</td>
<td>During winter break in an ordinary ‘computer lab’ at the university, the children worked with ToonTalk, lego mindstorms, and a website collaborating with children from Italy. 8 children.</td>
</tr>
<tr>
<td>2004/2005</td>
<td>Patcher</td>
<td>During autumn break in an art gallery. 5 children working together.</td>
</tr>
<tr>
<td>2004/2005</td>
<td>ToonTalk</td>
<td>Weekly full class sessions in the children’s ordinary classrooms during regular school hours, with computers provided by the university. Two full classes.</td>
</tr>
<tr>
<td>2005/2006</td>
<td>Patcher</td>
<td>At the natural history museum. Groups of different sizes, one hour at a time, working with the tangible ecology game.</td>
</tr>
</tbody>
</table>

Most of the sessions with children were recorded on video and short reflective reports were produced after the sessions. The reflective reports served to develop rich descriptions of aspects focused on in the activities, but also to continuously analyse and discuss what could be improved in following sessions. We also made technical analyses of the systems that the children designed and eventually finalised (see Figure 4). Thus, our empirical data was not merely what people were doing, but also the observable characteristics of artefacts. The main focus in the analyses was however on children’s actual interaction in the different settings with the visual and tangible programming tools.
The video material was collected using one or two digital video cameras during the sessions. Sometimes the children made video reports, taking turns filming what they thought was fun or interesting, but most of the time the cameras were standing on tripods somewhere in the room, normally behind a pair of children or a group working together. The material that we gathered is in no sense perfect – there is often much noise from other activities taking place in the room, parts of the screen is sometimes obstructed, and it is sometimes difficult to get an exact understanding of the children’s gaze, manual action, and so on. However, this basic setup captured most of the children’s speech, gestures, actions with physical artefacts, as well as their interface actions shown on the screen display. A more complex recording setting would have been difficult to manage, given the kind of activities that were conducted.

Much of the analytical work was based on recorded material from the workshops with children. This was done through conversation analysis in an ethnomethodological tradition (Heath & Luff, 2000; Suchman, 1987), making detailed transcripts and interpretive analyses based on these. A central element of such work is to understand the indexical aspects of language and interaction, for instance how people make sense of indexical terms like “this” and “there” and how they are able to understand what one is referring to through pointing and gesture (Hanks, 1992; Koschmann & LeBaron, 2002). This becomes especially important when cognition is conceptualised as an embodied phenomenon, and when focus is on practical activities.

Deeper analyses of specific sequences in the recorded material were made after having looked through some larger amount of video data, identifying episodes that we found especially interesting in some respect. This was usually short sequences (1-2 minutes) where some seemingly important activity was going on, but that at a first glance was difficult for us to interpret. By

![Figure 4. Behaviour structure of two simple systems created by children using ToonTalk.](image)
investigating the data in more detail we could get a better sense of what the children were actually doing in the activity.

Another reason for this approach was that it is well known that in user studies, children are seldom openly negative about anything, even when the technology do not work or when the researchers explicitly ask them about suggestions for improvement. The social setting may not sufficiently allow the children to be critical, and sometimes they do not have the technical experience to come up with realistic and innovative design proposals. This is something that has been commonly noted when conducting user studies with children, discussed for instance by (Read & MacFarlane, 2006).

The variety in settings of the workshops worked as a way of exploring aspects of different contexts and the target group that we worked with. Especially the repeated visits to the school became an important part in getting familiar with the culture in which the technology was to be used. Even though most of the sessions were recorded on video, and we put extensive efforts into making detailed analyses of such material, much of our general understanding of the situations arose from actually being at the school, talking to children, and observing their everyday classroom and play activities. Probably the most valuable part in this was getting to "know" specific children, which was an important resource as we discussed and evaluated various ideas back in our lab.

Figure 5. Visual overview of the process.
Informal observations from being with the children also guided our ideas for designing and setting up new activities. Everyday observations included how the children were always in small groups on the school yard, but in the classroom they seemed to have difficulties collaborating, especially around the computers. This was even though the computers were used primarily during the breaks for different kinds of social activities. And as we walked to and from the classrooms we noted how the pre-school children that we passed by seemed to be spending most of their time sitting, lying, or crawling on the floor – the computers on the desks in the corner were clearly not designed for their everyday practice. Though none of these general impressions from being in the school environment were noted or reported as results from the studies, they added to our sense of how activities and tools in such environments may take form, thereby taking an important role in our conceptual design discussions.

### 3.2 Design of new behaviour structures

A central part of my process concerned explorations into the design of programming constructs at different levels of abstraction. More specifically, this involved the design of libraries of ‘programming behaviours’ that could be combined together, as well as considerations regarding how such behaviours

![Figure 6. Behaviour structure based on a sketch made by two girls participating in one of our workshops, who had no previous experience of programming.](image-url)
could be visually and physically represented. Most of this work was done within the framework of existing programming environments, especially ToonTalk, but also Lego Mindstorms, Game Maker, Director, Java, as well as various tangible resources. Some of these explorations also took the form of prototypes and activities with no direct connection to specific programming tools. Figure 6 does for instance show a sketch of behaviour representations based on the kind of markings used in a storyboard designed by two girls in one of our studies.

At the early stages, much of my work also concerned the design of example systems built with the behaviours in ToonTalk. Due to some technical difficulties at the time with the ToonTalk programming environment, I also programmed a structure of behaviours in Director, and a few example systems using these, primarily as a way of exploring the idea of a set of general ‘programming pieces’ that could be dynamically combined and used in a range of different ways for making games and other animated systems. This could be regarded as a form of sketching, where the ideas were first tested with a (for me) less complicated material before implementing them in ToonTalk.

An important aspect was also to look at specific features of example systems and how these could be designed to actively engage the children, and in that way introduce them to the programming activity. Jakob Tholander describes in his doctoral thesis (Tholander, 2003) several situations where children worked with example projects in order to produce their own computer game design, but where children neither explore the programming structure, nor try to perform as many changes to the pre-built examples as the researchers expect. Sometimes the children liked the games as they were and preferred playing than making any changes, other games made the children exchanging only the graphical objects in the game, and sometimes the children even considered it as ‘cheating’ to make changes to the game. Even though the games had been selected as example projects because of their very simple computational properties, this was obviously not enough for inviting the children to explore these properties. Hence, an important design feature that we saw of games to be used for this purpose was that they should include potentials for improvements, in ways that are both technically feasible in the programming environment and meaningful in terms of game play and use. For the case of media-sensitive design, another important aspect was that the changes that the children made should include some experimentation with the elements that control program functionality.

Figure 7 shows a simple game that we used in our first study to introduce children to the concepts of programming with behaviours in ToonTalk. The game was loosely designed on the theme of Vikings, which the children worked with in school at the time. When playing the game the player uses the arrow keys to control a character, and the task was to make her pass two bouncing Vikings in order to reach a food table. When reaching the food
table, the computer makes a funny sound, and when colliding with any of the Vikings the player disappears in an explosion. An important feature of this game was that it did not quite work as a game, but that it had some easy to spot potentials for improvement. We had for instance deliberately included some space to add more objects and one of the Vikings moved significantly slower than the other, indicating that there were ways of modifying the speed of moving objects.

*Figure 7. The Viking game*
Figure 8 shows some of the variety of the games that different pairs of children ended up with after experimenting with the Viking game at their first confrontation with ToonTalk. Note that even though they all hold some sense of uniqueness, they are all quite similar, both in functionality, narrative structure, and the chosen graphical elements. The way the interaction took form does however illustrate the idea of combining and altering an existing design, and in that way attempting to make something new and personal.

3.3 Children’s Design with Low-Fidelity Materials

As discussed in section 2.1, a general ambition in my work was that children should be able to create a kind of computational artefacts characterised as being personal. This was reflected in a general attempt to let the entire production processes, from design and selection of media elements to specification of the computational features, to be owned by the children themselves. This relates to several important observations made by researchers over the years. Papert (1980) did for instance emphasise how personal ownership is an important trigger for children’s engagement, and several more recent studies (Ioannidou et al., 2003; Kafai, 1995; Resnick et al., 2000) have
shown how groups of school children often put substantial efforts into personalising their programming projects, e.g. through making their own pictures and other media elements, even though they often do that in groups or in pairs. Experience also shows that children’s engagement with their programming projects become stronger when they work with graphical elements that they produce themselves (Ioannidou et al., 2003; Kafai, 1995).

When working with ToonTalk, the children had access to a library of clip art pictures, and it was also possible to directly drag and drop pictures from any online webpage to use in the games. However, since it is difficult to find images for specific game ideas, we let the children create their own pictures. This allowed them to make use of a range of design materials and tools that they were already familiar with, such as crayons, felt pens, water colours, plastics, paper, and modelling clay. In the later workshops we also provided different kinds of textile materials to be used for brainstorming and for decorating and building background scenery for computer game design. We found textiles useful for this purpose since the other resources were difficult to use for colourising larger areas, such as the floor surface of the tangible system.

In the low-fidelity activities the children not only created the pictures but also developed initial ideas for what to build and how to extend systems that had been partially implemented. By letting the children design the visual elements away from the computer, they also got time to reflect on how things should work without having to be distracted by learning to use new tools. The objects produced in such sessions were then photographed, trimmed, and imported to the programming environment, in which dynamic and interactive properties could be added.

Figure 9 shows two games where the visuals were created using low-fidelity materials. The designers of both these games showed much pride in these games, even though the designs diverge quite drastically from the advanced computer games that they normally play. An important part in this
was probably that the children had designed all the visual expressions themselves, meaning that they already had a special personal relationship to the objects.

3.4 Role Plays with Groups of Children

Making use of low-fidelity materials in interaction design leads to an unavoidable ‘gap’ between the activities of programming ‘on the computer’ and designing with the physical materials. A common way of exploring computer-specific features away from the computer is to perform role plays of different kinds (Jacucci, 2004; Rettig, 1994). Early on we therefore experimented with activities around low-fidelity programming constructs.

The activities were modelled after the style of programming with behaviours in ToonTalk, and were used as a way of discussing, test running, and debugging systems run on the computer. The activities were thereby intended to work as a bridge between prototypes designed with the low-fidelity materials and systems implemented in ToonTalk. The materials used in these activities, such as behaviour cards and pictures, always corresponded to the programming resources in ToonTalk (see Figure 11). The activities were initially designed, in line with research on ‘participatory simulations’ (as described by Colella, 2000; Resnick & Wilensky, 1998), to support children in discussing what they were building on the computer, but after hand they became used more as prototypes for a new programming system.
Within the participatory simulation scheme, groups of individuals use their bodies to enact elements of a computational system by following a set of simple rules. By letting children enact the computational processes themselves, they get new perspectives on the system functionality. However, if each participant only represents objects that are visible in the computational system, several of the complexities remain hidden, such as the triggering of a particular action as an effect of an interaction between two objects. To overcome this we designed role-playing activities where participants enact the roles of the independent sub-behaviours of an object. This was to make computational aspects salient and concrete at levels which are hidden, or difficult to follow, while the game is being run. Staged activities such as these were performed in a range of different settings and with different resources.

What we found especially interesting in these activities was how the tangible resources turned the activity into a highly social one, were everyone participated. The children quickly took on different roles, such as for message passing and execution, and did in this sense also display a first person engagement in the activity. General for these activities was that the social setting seemed to work as a trigger for discussing the technical structure of the system that they explored, as well as providing a stage for physically acting out and displaying those ideas to one another.
Another observation was that tools that were made to represent lower-level functionality (for instance cups with sticks representing the size of an object) led children into discussions concerning the inner workings of computers and computational systems. On the other hand, collaborative and collective construction was more prominent in the activity when the physical objects represented higher-level relations and functionality, allowing for discussions directly connected to observable properties of objects. It was also evident how the large extents of dynamic features, even very simple games, make them hard to investigate through role plays with low-fidelity materials only.

3.5 Explorations of Tangible Technologies

The role play activities inspired us to explore the possibilities of designing a tangible system for programming that would mix the social character of the role playing activities, with programming on the computer. An early vision for the interaction setting was that it should have a similar character as when children build with physical materials such as sand, snow, and Lego bricks, i.e., that a group of children may sit together and through discussion build a shared ‘fantasy world’. The programming model aimed at was the mode of construction employed when children make programs using behaviours in ToonTalk, which is similar to the object-oriented approach to programming, where visual objects can be made to inherit properties of other objects and where readily prepared classes of actions can be dynamically combined. Core to this idea was to allow for object-instance relations between physical and virtual resources, and in doing this moving the programming activity away from the PC setting. This would enable children to in a sense actually reprogram the game in the physical space.

This process resulted in the design of a range of tangible resources. Figure 12 shows two early sketches, as presented in a project report from spring 2003. The idea was then to build a system based on a number of tangible objects to which simple behaviours (like anima gadgets) could be added, removed and moved between objects. The picture objects would be able to identify their own location on the background and also a number of connected ‘behaviour tags’. The leftmost picture shows a grid of bar-code readers, and the sketch to the right shows an idea for how the behaviour tags could be given a more physical form of attachment.
Starting with loose sketches like these, we explored a range of technologies, and performed role plays with children using these. We explored the use of a grid of RFID-readers, a grid of plastic pockets and cards, the possibilities of incorporating the use of wireless mice, and also the Lego Mindstorms toolkit. The leftmost picture in Figure 13 shows the first experiment at actually building a working tangible prototype, based on an RFID-reader assembled with a couple of iButtons, which could sense positions on a simplistic background grid. The construction of this prototype primarily served to explore the technology, and to get a hands-on feeling of where our ideas could lead us. At this stage we were not quite sure how the final system should work, but we envisioned that with a larger grid, with more moveable objects, and if those would be wireless, it would be possible to enhance our role-playing activities with some computational powers, making them more fun and easier to control.

Figure 12. Early sketches of structure of how tangible object could take form.
The first full tangible prototype that was tested with children was built in Java and based on the first prototype shown in the leftmost picture in Figure 13, which we later found had some problems in terms of how the system could be extended. This system was therefore reprogrammed from scratch using the structure that I had previously developed in Director. Other reasons for reprogramming the whole system in another programming language were to get a better sense of control over the development process, and in order to actually understand the interaction it was essential to take it from a state of a buggy prototype to a sharp system. The process also included aspects of the physical design of objects. The floor surface did for instance initially take the form of a white oilcloth surface, which we thought would be appropriate for e.g. making drawings upon. This material did however not afford sitting so much, so in our later design we used a soft fluffy fabric as the floor surface. This also structured the activity to become a ‘softer’ one, for instance that children naturally take off their shoes, and that they interact more carefully with the resources.

This process illustrates the importance of conscious reflection throughout a design process. A less elaborated description can be criticized as trivialising the relationship between a specific solution as a response to a more general design challenge (see e.g. Fällman, 2003). Moreover, if doing this, there is a risk of putting more efforts into making convincing accounts for how well the system or technology works, rather than focusing on why and exactly how it works (or not works). Sharing such reflections are however essential in order to understand the real values of the specificities of a design, e.g. that a particular solution is chosen in favour of another.

In our case, the design that we finally ended up with did diverge quite drastically from the initial ideas, though it was still based on the same initial ideas for interaction. To experiment with the technology, to program, to create different physical artefacts, to fiddle with low-fidelity as well as high-fidelity prototypes, all this was an important part in the ongoing
analytical process. Importantly, alongside these developments we conducted a series of workshop activities with children, where different tools and activities have been taken into practical use. In working towards the goal of supporting children to collaboratively create dynamic and interactive play worlds, we had to focus on an integrated view on the activity as a whole, where the context and the real interaction setting could not be excluded. These activities were based on workshops with children, practical design experiments in our lab, as well as on analytical studies of video recordings.
4 Interacting with Patcher

Figure 14 shows the setup of the latest version of Patcher, with a set of resources consisting of plastic play cards, a digitally enhanced floor surface, a screen display and wireless RFID-readers. The floor surface takes the shape of a soft carpet made of green fleece fabric, upon which a grid has been marked by stitched lines of white thread. Underneath each of the tiles on the carpet is an RFID-tag corresponding to a position on the screen. By placing a wireless RFID-reader on the play surface a position on the screen gets highlighted, onto which pictures may be added and programmed.

There are different kinds of programming cards, used to add behaviours and interactive properties to the pictures on the screen, and for controlling the system. Picture cards are used to place new objects at specific locations on the screen, while the behaviour cards are used to specify the functionality of objects that have already been added. Behaviour cards consist of a set of behaviours for movement, collisions, user interaction, and there are also cards that let you change ‘properties’, such as changing the size, or colour of an object. To add a new picture at a specific location, a picture card is placed on top of the creator block. Behaviours are added to existing objects by first making sure the creator block is at the position of the object one wants to program and then putting a behaviour card on top of it. There are also control cards, which let the user perform global actions to the system, such as

Figure 14. The Patcher system in use.
playing, stopping and saving what has been created.

Simple icons corresponding to the illustrations on the programming cards are used to indicate what behaviours and properties that are attached to each of the objects on the screen. When the program is set to play, all the programming signs are hidden, and the objects start to act according to the behaviours given to them. On the cards are icons as well as textual labels used to illustrate actions in ways that are richer than can be captured in the graphical symbols only. Pictures of physical objects are currently added to the screen manually through digital photography, quicker and easier to use ways of doing this are currently explored.

This chapter will be framed around the ideas of a digital patchwork as presented in chapter 2, and how the design of Patcher, the outcome of the observations and design cycle described in the previous chapter, connects to the current discourse in the area of tangible user interfaces and HCI. The discussion is based on the actual properties of the system as well as our findings from studying children using it together.

![Figure 15. Typical steps taken when adding things to the display.](image-url)
4.1 Media-Sensitive Design

An important design choice was to make the interaction space physically separated from the display surface, i.e., a digital projection of the game is displayed on a vertical screen rather than on the actual interaction surface. This reduces the problem of directionality of differing user viewpoints when interacting around a horizontal display, as discussed e.g., in (Mazalek, 2005). This also allows users to use the interaction space in a very flexible manner, such as spreading physical objects, and even sitting on top of the interaction surface, without obstructing the shared display. Furthermore, using a vertical screen means that the system can make use of affordable and commonly available displays in homes and in schools. A related design choice was to add an extra interaction layer (in the form of the readers) between the physical programming resources and their digital representations. Since this setup diverges for some of the basic assumptions often made when it comes to tangible interaction, this may need some further explanation.

One of the core reasons for letting children program for the screen was that much of what can happen on the screen is impossible to achieve in a physical space. For instance, users may like to add many copies of a specific picture, which would not be possible with a more direct mapping between each physical object and the pictures on the screen. Similarly, on-screen structures of code and the media elements that they control can be copied and reused, which cannot be done if the same programming structure is forced into tangible form. Tangible programming ‘code’ is in that sense difficult to achieve, unless one drastically restrict what such code should be able to express. A goal was also to create systems that would animate on a screen, meaning that independently of how tightly one connected the two, the representation on the screen and with the physical tools would diverge as soon as the system would play.

The primary reason for separating the mat and the screen, as well as the intermediary interaction level of readers, was therefore to provide for media-sensitive design, e.g. to let the same physical object be possible use at several positions on the screen simultaneously. A central function of the readers was also that they work as ‘sights’, allowing users to locate a position on the screen before adding something there. With a more direct coupling between every action on the screen and in the physical space, the system would in a sense be harder for users to control, for instance if all the unused resources will have to be placed off the interaction surface. Moreover, this setup means that the readers may be used for actions beyond adding and modifying pictures and behaviours at specific positions on the screen, e.g., to perform global actions, such as playing, stopping, and saving a simulation or game that has been built, as well as working as devices for interacting with the created system while it is playing. Therefore, a separation between the locus of interaction and feedback may not always be a disadvantage in design,
instead the gap could work to support the activity. This separation may also help making the distinction between material and tools used for programming more salient than is the case in entirely screen-based programming systems.

4.2 Making a whole out of pieces

An important theme of my work concerned making programming constructs more concrete by making representations that are closer to children’s own sketches and talk about how their games should work. In my work, I have explored this first with libraries of pre-built behaviours in ToonTalk, as well as exploring how similar structures could take a physical form.

The open behaviour structure of Patcher means that it is fairly easy to create different applications for specific themes and activities. However since the system does not allow for making ones own behaviours, or even to modify existing ones, the expressivity of the system depends heavily on the selected library of behaviours. Some other restrictions with this approach to programming have been discussed in more detail (Fernaeus et al., 2006), including the challenge of specifying how and where to visually present behaviours, and how more complex behaviours should be represented. If one would need to specify parameters and options for behaviours in a more dynamic way, and for programming at a more detailed level, a complementary lower-level language would always be needed.

Nevertheless, a simplistic programming tool that is limited to predefined behaviours seems sufficient for children at least in the contexts that have been explored in my work, settings where time is a limiting factor and the focus is on creating and exploring dynamic content.

Through various activities in different programming settings we found that many children had difficulties with conditional statements and numeric properties, especially when represented in tangible form. To handle object interactions, and to simplify conditional statements in Patcher we made use of readily defined classes that objects could be assigned to. For specifying relations between objects different programming behaviours were made with relation to these classes, for instance, any object could be given the property of a logical ‘colour’, and there were corresponding behaviour cards like ‘eat blue’, ‘chase green’, and ‘avoid red’.

To give an example of how children relate to program behaviours at this level, the following excerpt is taken from a short sequence where a group of children are discussing what behaviours to add to one of the characters in a fantasy world that they are building. When the transcript begins, one of the children (Sebbe) is just about to place the ‘green’-card on the reader (turn 15). In parallel to this, Niki turns around to locate a behaviour card lying behind her on the floor.
At turn 17, Niki holds forward a card (the card reading ‘eat blue’) and says ‘Yeah and it eats blue’. As a direct response to this proposal, Ivan takes the card from Niki’s hand saying ‘No, it shouldn’t eat anything’ (turn 18 and picture). At this very moment, Sebbe has turned his focus back from the screen display, and immediately agrees with Ivan ‘No it doesn’t eat anything’. Ivan holds the card towards Niki, and says once again ‘the baby shouldn’t eat anything’. As a response to this, Niki suggests ‘it bites’, upon which she and the other children laughs.
What could be seen as the most prominent aspect in this sequence is how the programming behaviours of objects, in relation to their narrative properties, became a very significant part of the negotiation. Bodily actions conducted with the programming cards shapes the relevant meaning of words like ‘eating’ and ‘blue’ to refer to the program action of eating and having a colour, as properties one can ascribe to objects being programmed. The programming construct of ‘eating blue’, is in fact a quite abstract construct, however here it seems to blend into the children’s discussion regarding the general story of the world they are building.

Moreover, that it was Niki who suggested the baby to have the behaviour ‘eat blue’, was possibly an effect of that card being located right beside her. It had been less likely that someone sitting at the other side of the mat and much further away from that specific card to come up with the same suggestion. In some sense the cards therefore works as structuring resources for the negotiation, not only for making existing ideas more explicit, but also for providing suggestions for new ideas. ‘Building with pieces’ may then be understood in a quite literal sense here, with the available resources taking a central part in shaping the product from the start, rather than the children first coming up with ideas that may later be found impossible to realise.

4.3 Tangibles for Social Interaction

Compared to a conventional patchwork activity, as well as to most other approaches to tangible programming, one must say that interacting with Patcher has a quite limited affordance and gestalt – in fact every command is based on the same physical action (placing a plastic card on a reader). Instead, many of the actions that children perform are ‘offline’ and directed to the social and physical setting, rather than to the software on the computer. When looking at how the children interact with the system, it seems that these forms of interaction were essential to using the system, both for collaborative purposes, as well as for planning and thinking about design ideas before actually implementing them on the computer.

Note for instance how the programming card that Niki initially puts forward (in Excerpt 1) is moved between the children, thereby becoming a means of representing arguments in the negotiation and also ‘transporting’ these between one another. This is especially since the card is not just any physical object, it is also the actual resource that the children use to complete the result of the negotiation. This means that the cards may function as external ‘memories’ of the current status of the discussion, while also being the means used for actually completing what gets agreed upon. In this sense the tangible shape of the behaviour cards work as resources in a quite different way than they would if displayed only on a screen in e.g. ToonTalk.
Most importantly, the physical setup with cards, blocks and interactive play surfaces on the screen and on the floor seems allow for creation of shared digital artefacts in a way that is different to most other computer user interfaces. This shows that even though the sensory experience of using the system is limited, the card based model of interaction still affords a larger set of manipulations than do conventional graphical user interfaces, primarily in terms of collaborative interaction. This means that it can be integrated into settings that require group work, which are otherwise difficult to combine with computer use. This relates to a possible conflict between rich sensory experience of interaction and what may actually be accomplished through the activity (see e.g., Benford et al., 2003). For instance, individual experiences of touch cannot so easily be shared (unless users actually touch one another or the same artefact), and do in that sense not on its own address collaborative aspects of the interaction.

The relationship between conventional crafting and design with digital materials has been extensively discussed for instance by Malcolm McCullough (1997), highlighting how physical properties of interaction are central to media production even in conventional computing settings. Just as with any other design material, designing with mouse and keyboard has to do with joint achievements of multiple senses, such as vision and touch, of spatial relationships explored through bodily action, and of visual affordances combined with manual skill. Instead of focusing on the experience of touch, my work has been more concerned with the features of tangible interfaces that explicitly support collaboration, i.e. the ability to incorporate gesture and a broadened range of ways for accounting for ones actions in a group. This also illustrates a potential clash between ethnomethodological theories, emphasising what is observable in the social space (e.g. Maynard & Clayman, 1991), and more pragmatic perspectives emphasising personal and sensory experience (e.g. Dewey, 1934; McCarthy & Wright, 2004).
In my analytical work, as well as in practical design activities, a central theme has been to develop a notion of programming that acknowledges its fundamental properties as a physical craft. In my published work, this has been conceptualised as embodied programming, referring to a perspective of programming where action and activity is studied as it takes place in the social and physical space of people. By using the notion of a ‘programming patchwork’, I have attempted to further conceptualise the relationship between computer programming and conventional crafting, and to examine how physical activity can support creative work with digital and otherwise intangible materials.

Based on this perspective, I will here elaborate on three aspects that I see as central to children’s craft activities. First, a relevant aspect of any craft is the role of tools, in this case the programming resources that I have created and explored. A second aspect concerns how activities are constantly getting reshaped by the people who conduct them and the cultures within which they take place. This means that throughout its short history, computer programming has never been a stable concept, but must be re-explored and re-defined continuously based on actual practice. In this case this has concerned what programming becomes in the context of children’s creative play. Lastly, activities get largely defined through their intended outcome, and in the case of programming as craft, this concerns the properties that are special for computation, and what it means to create with the digital material.

5.1 The use of tools

The general goal in my work was to allow for children to create dynamic screen-based systems. An effect of the target user group was that the setting should allow for collaborative activity, so we needed to move away from the conventional PC-setup and design a new set of resources, allowing for several children to interact at the same time. Important in this work was to explore different resources used for adding and programming objects displayed to a screen, including libraries of programming behaviours, example systems made with these, and also tangible computational artefacts. However, new resources affect not only the surface appearance of an activity; instead they must be understood to turn the activity into something that is also essentially
different to what it would otherwise be. In programming this is especially relevant since the actual appearance of the tools may take on so many different forms. Exploring how new forms of representation and interaction may change the activity of programming has therefore been an important part of my work.

Through analyses of video recordings in different settings (Fernaeus et al., 2004; Fernaeus & Tholander, 2003, 2006; Tholander & Fernaeus, 2006), we have illustrated how physical action can be an important element in children's programming activities. A basic example shows how external resources, such as gestures directed towards a paper, are used by children to negotiate ideas around the dynamic properties of a game that they are building together. Another example is taken from a collaborative role-play activity, showing how bodily enactment with physical resources can add a social dimension to discussing the workings of a game. A third example shows how computationally enhanced tangible objects work to bridge children's social interaction with actual implementation on the computer. (Fernaeus & Tholander, 2006)

By conceptualising programming in all these settings through notions of social performance and bodily action, we are able to point out aspects that are otherwise often neglected in the design of programming tools and environments. The primary observation is that programming in all these situations is largely becoming a physical activity. The children extensively rely upon and make use of bodily action and manipulations of interface elements, and they often do this for communicative and collaborative purposes. The examples also show how different representational forms structure the activity and how actions that the children take are intrinsically intertwined with these. Thus they are not only new forms of representations, but primarily new resources for action.

5.2 Programming as creative play

Many researchers and educators seem to agree that children are highly creative by nature and that artistic expression is as an important and enjoyable part in education (Barnes, 1987; Papert, 1993; Prentice, 1994). However, despite this, a common conception has been that the most natural place for computer programming in the curricula should be within the domains of other school subjects. One way of framing this is to define programming, or certain kinds of programming, as an area of science. For instance, in work that explores students' participation in computer programming projects, the activities are sometimes described as involving the “scientific method” (Smith & Cypher, 1999, p. 203) or even as “a wonderful example of the scientific method” (Resnick et al., 2000, p. 22). The essence of computer programming is then conceived of as the development of hypotheses (in the
form of programming rules) which can be evaluated (by running the program), and later reformulated (by making changes to the program). A general impression from our user studies is that to get children to focus on such aspects, all the expressive powers of computer programming may be a distracting factor.

Studies of how children interact with the Patcher system and in other settings, illustrate a view of programming that goes beyond the approach grounded in a tradition of educational technology, computer science, math, and science learning. In the hands of children, using these tools becomes an activity of mixing fantasies with computational properties, making rules for the activity as much as programming objects on the screen, and the process as being almost as important as is the final product of the activity. Apart from educational values, many children may want to create games, interactive play worlds, and dynamic stories, and to do that together with their friends.

5.3 Understanding the Digital Material

The making of textile patchworks is in many ways representative of a physical craft, characterized by the media providing specific sensory and multimodal qualities, and that the end result is in some sense persistent, static, and constrained by the laws of physics. For programming the end result is instead highly dynamic, intangible, allowing for substantial and quick alterations, it is easy to multiply and share, and is instead constrained by mediating devices (e.g. 2D screen surfaces).

An important issue in this work has therefore concerned what it means to create with mixed media of computational and physical materials and resources. Our analytical outcomes from the process, exemplified in the development of Patcher, illustrate the values of separating physical resources from the digital media on the screen, the importance of not letting digital manipulations get restricted by physical constraints, and in the case of children’s programming to allow for concrete modification of dynamic content through higher level programming constructs.

Attempting to combine programming with physical patchworking thereby holds some quite interesting challenges, which could be seen as referring directly back to the notion of ‘atoms and bits’, which is a common way of conceptualising the relationship between physical and digital artefacts and materials (see e.g. Ullmer & Ishii, 1997). Central to this notion is that it points out that computational ‘artefacts’ are essentially not made of physical matters (atoms), but of the stream of signals that controls the hardware of a digital device (bits). So while the physical world has some persistency in its physical manifestation, the digital ‘material’ is a dynamic one, made up of actions and instructions, without any real physical substance. This makes
explicit dichotomies that interaction designers are in constant struggle with, such as connecting properties of static-dynamic, physical-intangible, persistent-fluid, and so on.
6 Summary of Papers

Below is a short summary of all the appended papers, with a quick note on how they each connect to the themes of this thesis.

Paper I: Computational literacy at work

This paper describes how a group of 11-year olds interact throughout a programming project. The task that the children engaged in was to describe – with the computational media – the life conditions of an endangered animal that they knew of. The tools that the children had at hand were modelling clay, a library of tools for building simulations in ToonTalk, and a website for presenting their work. Our analyses were based upon detailed investigations of video recordings from the entire activity sequence of a pair of children working together.

An important observation in this study was that we found the results conflicting with arguments often put forward in other investigations made in similar settings, and also with the specific educational ambitions set up for the project. In our recorded material, very few episodes could be identified as related to either of the subject contents of science, mathematics, or even endangered species, even though this was the focus of all the teacher-led discussions. Instead, a number of other issues were much more prominent, such as negotiating and discussing aspects of playability, aesthetics, and functionality, as well as collaborative handling of complex technology. Additionally, interviews with the children, as well as with their teachers, revealed that they foremost suggested educational usages of the technology in areas such as language, arts, and history, hence areas which are not primarily formally oriented, but instead directed towards communication, culture, and media. A reflection that this gave rise to was that although the programming activity may have had some educational value, the ‘knowledge’ gained by the children could not be claimed to be primarily articulated in the format of ‘science content’. Moreover, the strong focus on physical handling of specific technology meant that the character of the activity became more of a practical kind than a theoretical investigation.
Paper II: “Looking At the Computer but Doing It On Land”

In this paper, we present the conceptual ideas behind Patcher based on three goals for interaction: supporting co-located collaborative activity, screen-based execution, and behaviour based programming.

We emphasise several aspects that we found especially interesting in how a group of 10 year olds used the different resources while building an animated fantasy world together. The results show how the tangible resources shaped the activity of programming so that bodily actions and positioning became prominent. This is conceptualized through the notion of embodied programming, which highlights how the activity must be understood through its interlinking to external resources and context.

Paper III: Finding Design Qualities in a Tangible Programming Space

In this paper, we reflect upon the process of developing Patcher, defined as a ‘tangible space for children’s collaborative construction of screen-based systems’. We elaborate on four qualities often put forward with tangible interfaces, i.e. the concepts of coupling, rich physical manipulation, the notions of input-output, and tangibles as representing digital information. We discuss how our design process changed our understanding of these concepts. In particular our design involved a shift from a focus on persistent representation and readability of tangible code structures, to instead focus on achieving reusability of programming resources. On a general level, our results illustrate a view on tangibles as resources for action instead of only as alternative forms of data representation. Importantly, this view includes action directed towards the computer as well as off-line socially oriented action conducted with the tangible artefacts.

Paper IV: Designing for programming as joint Performances among groups of children

This paper gives an overview of some of the activities and tools used throughout our research process, and that explicitly address aspects of social performance in programming, with a specific focus on children’s making of own computer games and simulations. We exemplify this work through three different situations where tools and activities are used by children as resources for building of interactive systems, while at the same time allowing for bodily action in negotiation of design ideas. We discuss how situations
like these may provide directions for new technologies for programming as well as methodological developments in the area of interaction design.

Paper V: Rethinking Children's Programming with Contextual Signs

This paper elaborates on the idea of programming at a level that reduces the use of conditional statements and traditional algorithmic control structures. We discuss how the structure of visual expressions could be used for representing programs at a similar level as how we have observed children to sketch out game ideas on paper. The approach is framed around the semiotics of comics, aiming at allowing for a form of programming that let children build computer programs in a direct and concrete way by using a class of signs that we call contextual signs. There are two aspects that distinguish contextual signs in comics from other sign systems used for programming. The first is that the signs (e.g. speech balloons, speed lines, arrows) are displayed in the immediate visual context of the object that they refer to. This is different to most other programming tools, for instance ToonTalk where the description of the behaviour can only be accessed through actively ‘opening up’ the object that it controls. The second aspect is that the signs are used to illustrate actions and properties in a way that is directly perceivable by the user. Instead of conditional statements, these take the form of a basic label with a word, picture, an icon, or a more comic-book like sign. It is important to note that our ambition was not to visually express the signs as would a trained comic artist, but to explore the expressivity of programming constructs at a level that remove the need of conditional statements.


