Tangible User Interfaces and Programming

Ilja Jetsu

February 1, 2008
University of Joensuu
Department of Computer Science and Statistics
Master's Thesis

Abstract

Tangible user interfaces is a relatively new research field in which computational processes are controlled through physical objects. The process of acquiring information about physical world is relatively difficult and toolkits have been created to allow the developers to concentrate on the creation of the user interface and computational logic by leaving the object recognition to the toolkit.

There has been research on tangible programming interfaces since 1976 when Radia Perlman created the *Slot Machine*. This thesis describes the design and implementation of a tangible programming tool prototype called TangibleProgrammer. TangibleProgrammer is a tangible user interface for programming Java based programs for a LEGO® RCX unit based simple robot. TangibleProgrammer is designed to be an easy to use programming environment for children. The user interface consists of physical code blocks that can be placed on a table top surface to form a structure to a program. The created program can be transferred to a LEGO® RCX unit based robot for execution. An early evaluation of TangibleProgrammer was done with a group of 6 years old children. The evaluation shows that the TangibleProgrammer was able to fill expectations required for this prototype.

ACM-classes (ACM Computing Classification System, 1998 version):
D.2.2, H.5.1, H.5.2, I.4.9

Keywords:
Tangible User Interfaces, Programming Tools, Human Computer Interaction

Acknowledgments

I firmly believe that it is necessary to name some persons and projects that have helped me in accomplishing this task.

The project *Technologies for Children with Individual Needs* in which I was working while beginning this thesis. Through the project I got in touch with the group that used the TangibleProgrammer during the evaluation.

Antony Harfield and Bruce Windram were a real help as I don't speak English as my native language. They proof red this thesis and I also got some suggestions considering the development of thesis from them.

Professor Erkki Sutinen for giving free hands to select research topics that I was interested in and giving guidance during this process.

My wife Tuija, naturally. Thank you for helping me through this long process of writing this thesis.

Contents

1 Introduction

Tangible user interfaces is a relatively new research area. The background of the term ”Tangible User Interface” (TUI) can be traced to the works of Hiroshi Ishii and Brygg Ullmer at the Massachusetts Institute of Technology in 1997, when they proposed an idea of a new type of Human-Computer interaction [IU97]. The main idea behind tangible user interfaces is to augment the physical world by combining the digital information with everyday physical objects. Users can interact with the interface by touching, grasping, moving and modifying real, physical objects. This allows the users to control the computational processes through familiar, easily graspable objects instead of mouse and keyboard. This thesis is based on the idea that Tangible User Interfaces (TUIs) could be used to ease the learning curve of programming environments that are targeted at children by bringing the user interface to the real, physical world; reducing the use of textual representations and simplifying the user interface.

One of the major fields of research at the Department of Computer Science and Statistics at University of Joensuu is educational technology [UoJS07]. This research is done in co-operation with schools and other educational institutions [Tek07]. A project called *Technologies for Children with Individual Needs* was started at the University of Joensuu, Department of Computer Science and Statistics in fall 2005. The goal of this project was to develop methods, models and tools for overcoming learning challenges in special education [Tec07]. One topic of interest for the project was *educational robotics*. The term educational robotics means using robots as a tool for teaching subjects other than robotics [SE04]. According to research done at the University of Joensuu, educational robotics can enhance collaboration, cognitive skills, self-confidence, perception, and spatial understanding when working with children [KLPBSV07].

Research that was done during the Technologies for Children with Individual Needs project brought up shortcomings in the programming environments that are meant for young people. These shortcomings seemed to be related to common problems that children with individual needs may have, like difficulties either in reading or in logical reasoning. There are graphical programming environments that are marketed as designed for children, for example LEGO® Robotics Invention System [LEG07b], LEGO® RoboLab [LEG07a], easyC [Int07] and IPPE - Instructive Portable Programming Environment [JZKS07]. Even though these programming environments are relatively simple to use, there are concerns for using these with children. Most of these

programming environments are only available in English. That unnecessarily limits the user group, because not all children are able to read and understand English well enough and there were children taking part in the project who had problems with reading even with their native language, Finnish. Part of the students taking part in research were in kindergarten and they could not read, which limited the use of textual messages even more.

Tangible user interfaces can be used to help children in the learning process as "TUI is a natural interface which requires little cognitive effort to learn" [Xu05]. This allows the children to concentrate more on the task instead of using the computer or programming environment. Tangible programming interface could therefore be approached more easily than traditional mouse driven graphical programming environment.

1.1 Problem Definition

The main goal of this thesis is to research the possibilities of tangible user interfaces for creating a tangible programming environment that is called TangibleProgrammer. The main subject is divided into four questions whose answers will together form the results of this thesis.

- **Q1**: Definition: What does the term *Tangible User Interface* mean?

  Knowledge of relevant terminology and research background is necessary for understanding the whole concept of tangible user interfaces.

- **Q2**: Design: What possibilities are there to create a tangible user interface for programming?

  There are different technical design aspects that must be considered when creating a tangible user interface. What are the options, and what possibilities and limitations do different selections produce?

- **Q3**: Implementation: How difficult is it to implement a tangible programming interface? What problems can be expected while developing one?

Implementation is as an important part of development process as the design phase. The problems in design can be identified during the process of implementation.
Requirements that the TangibleProgrammer must meet to be a tangible user interface for programming:

1. The user interface must be able to recognize and keep track of the physical objects.
2. By using the interface, the user must be able to create programs that will be compatible with a LEGO® RCX unit based robot.
3. The system must be able to move the created software to a LEGO® RCX unit based robot for execution.

- **Q4**: Evaluation: How well did the tangible programming interface manage to fill out the expectations and how should it be modified for continued usage?

1.2 Methods

The research process of this thesis is divided into the following steps:

1. Research problem definition.
2. Literature survey (**Q1, Q2**)
3. Creating the prototype (**Q3**)
4. Evaluating the prototype (**Q4**)
5. Conclusion and future directions

The research questions defined in Section 1.1 give this thesis the basic structure and define the needs for methodology. Research questions Q1 and Q2 require investigation of literature in order to gather necessary background theory and information about previous research. The first part of this thesis concentrates on compiling information about tangible user interface research and finding out what kind of research has been done before. Chapter 2 is related to question Q1 and Chapter 3 relates to question Q2.

For question Q3 the answer is easiest to be gained through the implementation of a TangibleProgrammer prototype and evaluating it. Answers to this question are collected from the notes that are written during the creation process. This begins in Chapter 3 and continues on Chapter 4.

The created TangibleProgrammer prototype must be evaluated to acquire answers to question Q4. Evaluation is organized so that the results can be gathered through a method called *observation*. As Cochen *et al.* [CMM02], who have authored a book called Research Methods in Education, explain, observation is a versatile research method, which allows the researcher to collect information from multiple settings including: the physical setting (physical environment and its organization); the human setting (organization of people, characteristics of groups and individuals being observed); the interaction based setting (interactions that are taking place) and the program setting (resources and their organization).

As there are no pre hypotheses about what is going to happen during the evaluation session the observation must be done as an unstructured observation [CMM02]. The unstructured observation method is basically unplanned, informal, watching and recording of behaviors as they occur in a natural environment. Because the author is an active participant in the evaluation session, the observation cannot be performed by the author in real time. The observation is therefore recorded with a video camera, which will record the whole situation while not affecting it substantially. Resulting tapes will then be analyzed using methods that are developed for unstructured observation. The main things to observe in this evaluation situation are the relationships between the users and the machine. Evaluation is discussed in Chapter 5.

The last Chapter of this thesis is Conclusions and future directions. In this Chapter the research results are gathered together and the final verdict of feasibility of TangibleProgrammer prototype as a programming tool is given. The Chapter will also introduce some options of developing the TangibleProgrammer further.

2 Tangible User Interface

This Chapter focuses on literature review about tangible user interfaces. The purpose of this Chapter is to answer the first research question: What does the term *Tangible User Interface* mean?

Radia Perlman is considered to be the first to create a tool that fits the description of a tangible user interface [MCK06]. She created a system called Slot Machine [Per76] while working at the Massachusetts Institute of Technology in the 1970s. The Slot Machine allowed users to create programs for the LOGO language without prior knowledge about computers. Although Perlman never used the term tangible user interface to describe her research, Slot Machine is also considered to be the first published project that has used tangible user interface principles in the programming context. There is a more detailed explanation of Radia Perlman and the Slot Machine in Section 2.2.

The term tangible user interface (TUI) has rooted itself relatively well to the research community. Other similar terms have surfaced, but none of them has gathered as widespread consensus as TUI. As O'Malley and Fraser [OF04] state, the history of the term tangible user interface is often mentioned to start from the works of Hiroshi Ishii and Brygg Ullmer in the MIT Media Lab. They published a paper: "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms" in 1997. Another term that is also used is "Graspable User Interface" by George Fitzmaurice [Fit96] from the University of Toronto. The term *tangible* explains the main difference between tangible interface and graphical user interface. The Merriam-Webster Dictionary [Mer07] explains the etymology of the word tangible back to the Latin word *"tangere"* which means "to touch".

In a tangible user interface the interface is developed to give physical form to digital information [Ull02]. The user can grasp the tangible objects and take control of the system. In traditional interfaces the user gives the computer information through the input device and waits for the computer to process the task and display the results on the output device. The user must then try to understand the output in the context of the given input. In a tangible user interface the input device is usually designed to work also as the output device. As it has been mentioned in [CSR03b, FIB95], the input in a tangible user interfaces is constructed from real physical objects that act as physical handles to the running software and allow direct control over the desired computational

processes. The user can physically select and modify desired controls and by that input the data through for example moving, replacing and rotating the objects.

Tangible User interfaces and *Augmented Reality*(AR) are both subclasses of *Mixed Reality* research. Mixed Reality defines the merge of digital and physical worlds, where the physical reality is extended and augmented with information that originates in virtual reality. Augmented reality could also have been a good description for tangible user interface technologies, but nowadays AR means basically head mounted 3D glasses that augment reality by embedding virtual objects to real environment [SV00]. Tangible User Interfaces on the other hand are expanding the reality in opposite direction, towards the computer. The physical reality becomes part of the computers user interface.

The term Object in the tangible context is a difficult subject. The term object can be used to describe the physical things as well as the objects that exist inside the software. This can create even more problems in the area of tangible user interfaces as the same object can have both the physical and software based meaning at the same time. Because of this difficulty Ullmer [Ull02] has criticized the usage of the term object in this context. As most physical objects have no connection with tangible interfaces it is an ambiguous term. The term ”physical/digital objects” has been used to clarify this ambiguity, highlighting the dual physical/digital aspect of TUI elements. Another term ”Physical icon” or ”phicon” is used to describe the tangible user interface objects. The term was introduced by Ishii *et al.* [IU97] who noticed that the controlling device of a tangible user interface could be referenced with the “icon” concept borrowed from the graphical user interfaces. The use of this term has been discussed and later it has been noticed [UI01, Ull02] that this term also has shortcomings: “strictly speaking, many so-called “icons” (and “phicons”) are not “iconic,” but rather “symbolic” in form.”

Figure 1 shows an example of a tangible user interface. Reactable is a tangible user interface that can be used to create music. The system recognizes special objects that are placed on the tabletop surface. The object that is placed on the center of the interface acts as a microphone, or an audio output device. Other objects on the table are designed to represent creation or modification of sounds. The user can move the objects around and rotate them to change the playing sounds. The further away from the microphone object the sound objects are, the quieter the sound they create. Object recognition method used by Reactable is visual recognition. The computer generates graphics that are projected onto the same surface as the objects; the visual output is

calibrated so that the graphical output of the objects matches the physical ones. The user sees the visualization of the sound waves traveling objects and can reorganize the objects to change the order in which the sound is processed.

Figure 1: Reactable is a tangible musical instrument [Rea07]

2.1 Why Tangible?

Machinery that people used before digital technology tended to be relatively easily graspable. The user interface of a machine usually had a direct one-to-one mapping between the switches or levers and the functions that the machine could do. Then electronics developed and embedded microprocessors were invented which resulted in changes to the design of user interfaces of everyday objects [McN00]. Control panels became simpler, there were less buttons to press, but more functions. A good example of this development is a multi-function digital watch which has many functions, but only three buttons. This multiple uses for single controller - approach often raises the learning curve of a user interface as the user must remember the correct usage patterns; but just giving each available action a dedicated button does not always help. For example remote controls of home entertainment systems usually have a button

assigned for each available function, but the user interface is not necessarily any easier to learn due to the vast amount of different functions [McN00].

The user interface must therefore be simple, but allow necessary tools for required functions. A word that is often used in the context of tangible user interfaces to classify different tools is *specialized* [FB97]. One control in a tangible user interface cannot usually be used for more than one function. This specialization helps the user in the process of grasping the interface. For example a computer mouse cannot be classified as a specialized tool [KBNR03]. The mouse is a multipurpose tool and can be used for many different tasks. It has different meanings in the different parts of the user interface and the user must learn to use the mouse in a different manner in different programs [Fit96]. Physical objects that people are used to handling rarely have more than one meaning or function, but they are designed to be the best possible tools for the task in hand. Accountants, animators, sound engineers and graphic designers, even though their use of computers differs a lot, all use a similar set up of input devices, consisting of a mouse and a keyboard [FB97]. This keyboard and mouse combination seems inefficient for specialized jobs like graphical design or animation.

Another criterion by which the input devices can be classified into different groups is their input model. There are two main groups of input models. Input devices are space-multiplexed, time-multiplexed or both space- and time-multiplexed [FIB95]. With space-multiplexed input each function that needs to be controlled is attached to its private transducer which occupies physical space of the working environment. Time-multiplexed controllers are used so that different functions are controlled at different time points. For example (for devices used for expressing text) time-multiplexed controller could be a button that is used for encoding alpha numeric characters in Morse code and the space-multiplexed input option could be a QWERTY keyboard which has one key per character [FB97].

Audio mixing software is a good example of the differences between a specialized controller and a mouse [Fit96]. The time multiplexed control scheme of a mouse requires that in order to control multiple volume setting sliders the user must perform multiple tasks. For each slider that the user wants to control, the user must first attach the mouse to a slider by pressing the mouse button down. The selection of the value is done by moving the mouse and simultaneously observing the values on the image of the slider. When the correct value is set the mouse must be detached by freeing the mouse button. The specialized controller allows the user to control multiple faders at the same

time, just by moving them manually with fingers. A mouse gives the user only one active pointer but humans tend to have 10 individually controllable fingers which can be used when working with the specialized tool. An example of such audio mixing program, Digidesign Pro Tools [Dig07], is presented in Figure 2a. Power users can use shortcuts or fader groups to shorten the time required to make these adjustments, but compared to the specialized tool, this non-specialized tool is a lot more difficult and time consuming to use [Fit96].

(a) ProTools Audio Mixing Software (b) ProControl Control Surface

Figure 2: Programs (a) may benefit from the dedicated controller (b). [Dig07]

Due to the advantages of specialized tools, people working in audio production environments and studios have bought different kinds of controllers that allow them to handle multiple aspects of the recording process simultaneously. Figure 2b presents one option for such a device. These specialized devices are not necessarily suitable for different computing tasks. For example drawing pictures by using an audio mixing controller would be difficult. This is one side-effect of specialization, and therefore specialized devices are rarely usable outside their original context [Fit96].

The space multiplexed nature of tangible user interfaces gives it an advantage over the normal graphical interfaces when working in groups. Graphical user interfaces are not really well suited for collaborative work when all the participants are located in the same physical space. A mouse is a tool that can be only held by one person at the time. The process of doing collaborative designing is not as easy as plugging more mouse-type pointing devices into the computer, as most graphical user interfaces are not designed to work with multiple active pointing devices. There has been development for the use of multiple mice in one environment [PPGT07], but these technolo-

gies are still in the development stage. Technologies that allow people to collaborate through a network do exist, but that requires that all collaborating users are using their own computers. That, on the other hand limits the possibilities for communication between participants in collaboration. Collaboration that is achieved through computers and networking is also limited compared to methods of collaboration which can be applied to tasks that are not computer related. In collaboration, the space multiplexed nature of tangible user interfaces works well. In a tangible user interface all the users collaborating in the task can get their hands on the objects and use them to modify things [McN00].

Users of tangible user interfaces can take advantage of the learned motor behaviors [Fit96]. When people use tangible tools they do not need to focus on the task of moving the objects around as much as they must with GUI. People know their motor responses, and have learned to manipulate real life objects in their everyday life using only muscle memory. Users can often recognize the thing they are holding by identifying the weight, texture and shape, all without the need of visual help.

2.2 Programming with Tangible Interfaces

When the first computers arrived in classrooms and teachers allowed students to use them, the term "computer literacy" often meant learning to use the computer by making programs for it. At that time LOGO was one of the most widely used programming languages for beginners. LOGO allowed students to use the computer to control the robotic "turtle" to follow their instructions [Pap80].

Nowadays almost all students are taught the basic skills of computer usage in elementary school; when moving to work after school computer literacy tends to even be a requirement. That literacy however often just means that one has the basic skills that include the use of applications such as text processor and spreadsheet. Programming computers is not anymore considered to be part of that literacy and is left to highly trained professionals [WC00].

This does not mean that programming, broadly speaking, would be solely a professional task [McN00]. Musicians use similar practices in their work while they program their synthesizers to accompany them in performances. In a similar way accountants use spreadsheets and create new functions by using the programming language that is

available to them inside the application. Also consumer technology nowadays requires rudimentary programming skills from the general public. Video recorders, microwave ovens, bread makers and other household devices are at least partially programmable.

LOGO's spirit of allowing young people to experiment with computers has not faded during all these years since 1967 when Seymour Papert created the LOGO language [Log00]. There are schools and other institutions which are still using and teaching LOGO or other similar, simple programming languages for beginners. The Department of Computer Science and Statistics at the University of Joensuu has also developed a programming language for beginners called IPPE [JKS02]. Programming tools are more easily available and while the robotic turtles used with the LOGO were expensive and fragile [McN00], nowadays children can experiment with programming by using robotic tools like LEGO® Mindstorms [LEG07b] or VEX [Inn07, Int07] that they have constructed themselves.

The programs that children create are usually relatively simple and easy to make. This motivates the children and teaches them structured thinking [WC00]. Programming environments however are usually not usable enough for children to use alone. The programming language and the user interface of the program can be simplified by using a tangible user interface and therefore be made accessible to the hands and minds of younger children [McN00].

There have been programming related projects that use the idea of tangibility. These tools try to bring the abstract concepts of programming closer to the real world, to be more easily graspable [McN04]. The tools introduced in the following Chapter have been selected to widely represent possibilities and technologies for the creation of a programming environment in a tangible way.

Perlman's Slot Machine

One of the first programming environments that can be considered tangible was Radia Perlman's *Slot Machine* [Per76] which was an interface for programming in the LOGO-language and controlling a turtle. (The turtle was either a real mechanical robot or a screen based simulation of such a robot). The Slot Machine employed plastic cards that could be inserted into one of three colored slots (red, yellow, and blue). These cards (namely different "action," "number," "variable," and "conditional" cards) placed into slots presented the structure of the program. It was also possible to stack

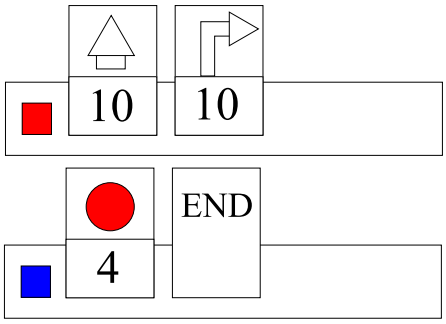

Figure 3: An example program for Slot Machine

action and number cards together to form different commands. Stacking number 2 with Turn Right created program code that made the turtle turn right twice. On the left side of each slot was a “Do it” button. When that button was pressed, the turtle performed the actions pictured on each card in the rack, in a sequence from left to right.

Maybe the most unique thing about the Slot Machine was the possibility of coding subprograms. There were 3 different slots, which were color coded. The system also had color coded jump-cards, which directed execution to the beginning of a slot which was the same color as the used card. Using the blue jump card on the end of blue slot would jump the execution back to the beginning of the blue slot, hence creating never ending loop. This in conjunction with variable-cards made things like using the turtle to draw a spiral or rectangle much easier.

An example setup of using the turtle to draw a rectangle with a Slot Machine is pictured in Figure 3. The red slot has the basic parts needed for the one side of a square. The turtle is instructed to move forward a determined length, which is the desired length of the side of the square. The second command on the red slot turns the turtle 90 degrees to the right. The blue slot also has two cards. First card causes the execution to jump to the subprogram and execute it. As the variable card four attached to the jump card commands, the contents of the red slot will be executed four times. When the execution returns from the fourth execution cycle of the red slot, the last card in blue slot commends the execution to terminate.

Slot machine is an abandoned project as nothing has been done with it since 70’s. The system was hand built, expensive, bulky and fragile. Perlman mentioned, that it would be nice to have a smaller cheaper version of the Slot machine, but because of technical limitations of the era, it was never constructed.

AlgoBlock

The AlgoBlock system is a programming environment that is somewhat similar to the Slot Machine. The AlgoBlock system consists of a set of physical blocks made of aluminum that connect to one another to form a program. Each block corresponds to a single command in the LOGO programming language. It is possible to adjust variables in blocks by turning or touching embedded knobs and levers. The term "tangible programming language" was actually invented by Suzuki and Kato in order to describe the AlgoBlock collaborative programming environment [SK93]. The system also facilitates collaboration based aspects by providing simultaneous access and mutual monitoring of each block [FIB95].

Figure 4: AlgoBlocks [SK93]

Figure 5: Programming by using tangible cubes [Smi06]

GameBlocks

The Gameblocks system consists of blocks that are placed on trays to form a control sequence [Smi06, Smi07]. The recognition of an inserted block is done by magnets that are installed at the bottom of each cube. The rack has reed switches that react to the magnets in the blocks. The current implementation allows 31 different identities for each cube. Each identity can be used more than once. The created program is transmitted through infra red signals to a humanoid robot that will execute the commands. The system is still a prototype and some problems have been noticed. The cubes are too big or too slippery for small children to handle and the icons on the blocks are not easily identifiable. The structure of the blocks also requires some work, as the corners of the objects are sharp and the magnets that identify the objects do not always stay in the correct position.

MouseHaus Table

MouseHaus Table [HDG03] is not a programming environment in the same sense as the previously introduced environments. It allows participants, who have no previous experience with computers, to interact and program a pedestrian simulation program. That is basically programming, but not in the same sense as on the other programming environments as the language used in the MouseHaus Table is much more limited. Interaction with the simulation program is done by creating a map of a certain area through physically cutting out different shapes from sheets of paper and placing them on the table as Figure 6 shows. The shapes then become buildings, parks and other types of surroundings that can be simulated depending on the selected color of the paper. For example green shapes could be parks and grey shapes could be buildings. Preliminary tests showed much more interaction from children when they were using the paper-scissors interface compared to the normal mouse driven interface [HDG03].

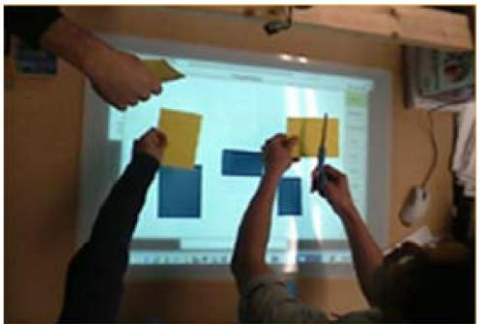

Figure 6: MouseHaus Table [HDG03]

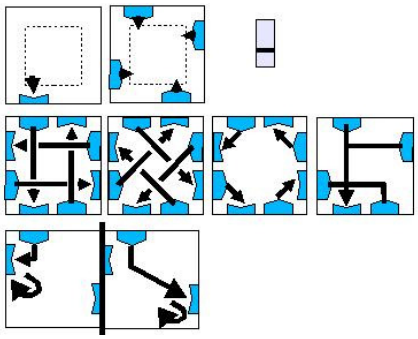

Figure 7: The basic set of C-cards [Val03]

C-Cards

C-Cards are computational cards for learning programming concepts [Val03]. C-Cards are tangible in the sense that the system is based on physical cards which are used to represent the software logic. The cards can be used as a tool for teaching programming logic. The internet based computer software exists, although there is no direct connection between the computer software and the structure built with the cards. The user must transfer the created structure to the computer by hand. Figure 7 shows the basic set of cards available in the environment.

Tangible Programming Bricks

Tangible Programming Bricks [McN00] are *computational objects* that are based on the *programmable structure* type of system. The input of the system is the locations of the objects within the structure. This location information is gathered through the connectors on the top and bottom of the blocks. The output is organized through integrated displays or other output devices that can be connected to the system like any other block. The system does not require an external computer. Figure 8 shows an example set of Tangible Programming Bricks that could be used to program a microwave oven.

There are also problems with the bricks. They are uniform in shape, which is problematic as it lowers the chance of differentiating the blocks without visible cues. The structure of the blocks is organized in a fashion such that it limits the building, the blocks can be stacked only in one dimension. This disallows, for example, the possibility of creating an if-else program structure as the code path cannot be split. They are also expensive to manufacture. The cost of a single object, including microprocessor and connector system is far too expensive for the creation of programming kits for consumer and educational use [McN00].

Figure 8: Tangible Programming Bricks [McN00]

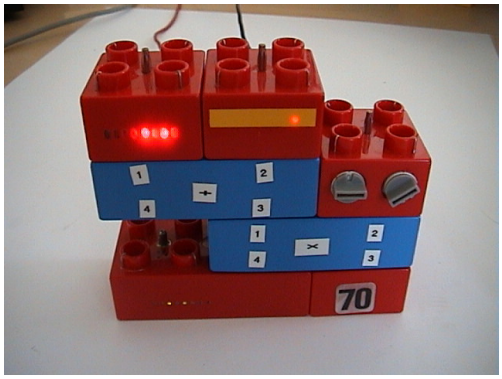

Figure 9: I-Artefacts [Lun07b].

I-Artefacts

I-Artefacts (known also as I-Blocks) are a prototype of a computational objects based programmable structure. Active parts in I-Artefacts are built inside LEGO® DUPLO® blocks. Each block can connect to other blocks as Figure 9 shows. Inside all the blocks there are electronic components, so the building of such objects is expensive. The shape of the blocks gives the user guidelines about the direction of building

and the method of connecting blocks together [Nie02]. The structure that the bricks now have is not firm as the connectors are based on the original LEGO® DUPLO® connectors which do not lock together. Some guidance is needed in the building phase as the objects are only identified by a number tag, and the number must be checked from paper to gain knowledge about the identity of a block.

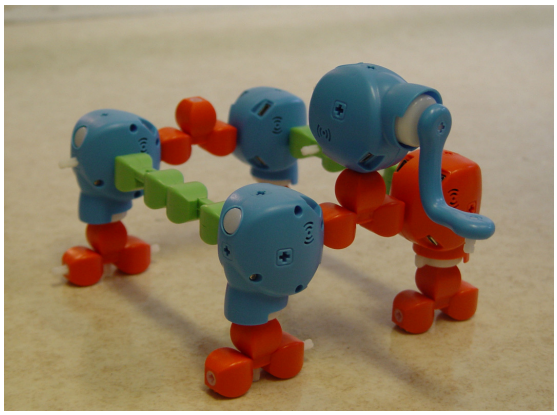

Figure 10: Topobo blocks creature

Topobo

Topobo [Raf04] is a three dimensional constructive assembly system embedded with kinetic memory. This allows recording and playback of physical motion of that constructed assembly. Topobo system consists of active and passive blocks which can be combined together to quickly assemble larger structures, which can then be animated by moving parts that are connected to active blocks. After the animation has been created it can be played back by pressing a button. As the conducted tests show, students are quickly able to develop various types of walking robots [Raf04]. This suggests that the interface can assist the students in understanding balance, leverage and gravity.

Summary

Table 1 lists all the programming tools mentioned in this Section. The tools differ on the aspect of how they implement the programming activity.

Programming interfaces are created for making programming easier. There have been many different designs which have attempted to create working tangible programming interfaces. The Slot Machine was the first tool that could actually be called a tangible programming environment but the creators never did call it such. AlgoBlock creators did refer to their system with the term tangible programming language. GameBlocks

Table 1: Summary of Tangible Tools for Programming

System	Type	Year
Slot Machine	Programming interface	1976
AlgoBlock	Programming interface	1993
GameBlocks	Programming interface	2006
MouseHaus Table	Programming interface	2003
C-Cards	Computational card game	2003
Tangible Programming Bricks	Computational objects	2000
I-Artefacts	Computational objects	2002
Topobo	Programmable structure	2004

can be used to program small robot and the MouseHaus Table can be used to program pedestrian simulation software, thus they are also programming interfaces. The C-Cards system is not a programming environment in the same sense as other environments as it is not able to compile or execute the created program. The program can be recreated on a computer for execution.

Tangible Programming Bricks and I-Artefacts are both Computational objects. The object in use is the user interface, it forms the program and also executes that created program. This is a compact system which can be used also in places where there are no computers available. The drawback of this sort of systems is the cost per object, which limits the commercial development of such environments. Topobo is in a way similar to computational objects; all the necessary processing logic is implemented inside the objects. However it has limited computational possibilities as the programming is limited to the movements of the integrated motors.

3 Interaction Technologies

In order to create a working tangible user interface, different possibilities for technological implementation must be researched. This Chapter aims to answer the second research question: What possibilities are there of creating a tangible user interface for programming? In Section 2.2 there were some examples of different kinds of tangible user interfaces that could be used for programming. As the examples show, there are many physical technologies that can be used for implementing a tangible user interface. The technical characteristics of the needed implementation technology are largely dependent on the actions that the developed interface must be able to do.

Tangible user interfaces are defined by their input methods, namely physical, tangible objects. In addition to input, also output is naturally an important part of computing experience. Therefore it is important to select the correct technology for input and output. The mind map on Figure 11 sums up the contents of this Chapter. The technologies mentioned in the mind map are discussed in detail in the following sections.

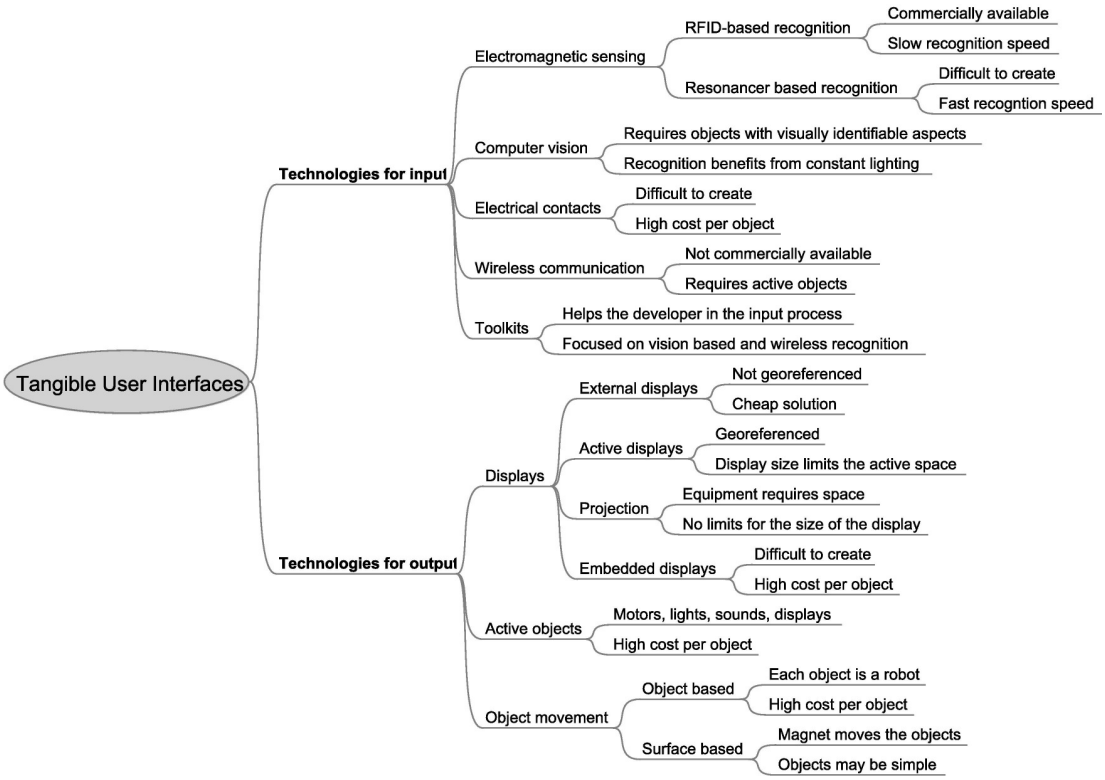

Figure 11: Mind map about technologies for Tangible User Interfaces

3.1 Input

Input, in the context of tangible user interfaces, relates mostly to the acquiring of information about the physical environment. The *reliability* of the input is the most important aspect that affects the usability of the tangible user interface [BKJ05]. The reliability can be divided into the subcategories *precision, speed and latency*. Precision means that the identity of an object must be recognized correctly. The misinterpretation of an object as some other object may disturb both the user and the developer. The need for precision depends a lot of the type of the system. It is also usually important that the location of each object is recognized with the necessary precision.

If it takes a long time for the interface to recognize a new object the user may begin to question the functionality of the system. Therefore latency and speed of recognition are both aspects that affect the usability of the system. Latency is the time that the system needs to react to changes in the objects. If the user moves an object and the system doesn't respond to it in a reasonable amount of time, the system cannot be said to be user friendly [BKJ05]. The latency is a measurement whose limits depend a lot of the desired interactivity of the interface. For situations where the output is not modified in real time the latency is not problem, but, for example, programs used in musical composition or graphical work need almost instant responses from tangible objects in the system, or the usability suffers. The number of simultaneously active objects has a large effect on the latency in certain technical implementations, like recognition based on radio frequency identification (RFID) [HP99]. This technology will be covered in more detail in the next Section.

A decision that affects the technological choices regarding input is whether the objects of the tangible user interface should be *active or passive*. Selecting passive blocks limits the gathered information to the location of the object and possibly the direction the object is facing. Active blocks on the other hand allow the designer to add advanced actions to the objects. Objects can have buttons, dials or other moving parts that can be interpreted which gives the user the ability to modify aspects of the software. If the technical implementation supports bidirectional communication, the object may even have output devices like lamps, displays, motors and speakers inside [Nie02].

Input, in tangible user interfaces, can be achieved through various technologies. The following sections represent a collection of the most widely used technologies in tangible user interface input. Section 3.1.5 introduces a group of toolkits that can be used

to help the development of a tangible user interface by leaving the object recognition to the toolkit.

3.1.1 Electromagnetic Sensing

Electromagnetic sensing is a group of technologies that use different forms of electromagnetic recognition to sense the place of the object. These technologies have been used for various tasks. There are companies like Wacom [Wac07] and Ascension [Asc07] that have created products that use electromagnetic sensing for object location. Most touch screen variations also use sensing based on electromagnetic effects. The problem with these readily available technologies is that they are not very well suited to tangible user interface creation. The commercial systems are usually limited by the number of input devices at use it the same time [PIHP01]. Because of the need of specialized electronics and complex designs these systems are usually also expensive [HP99].

Electromagnetic sensing as a technology for object recognition is generally considered to be better than visual recognition in terms of reliability, precision and speed [HP99]. All of these are important aspects to consider while creating a tangible interface. With electromagnetic sensing the user does not have to deal with changing lighting conditions, which can cause problems with image recognition based systems. Electromagnetic technology also allows for the creation of active blocks or blocks that can be modified (dials, switches etc.).

Electromagnetic sensing can be divided in two different groups; the location of the sensors being the deciding factor. In the first group the surface recognizes the object placed on it. In the second group the objects calculate their own location on the surface. If the object provides the sensing function, it may need external power and a way to output the gathered data. This usually requires wires to be attached to each of the sensing objects [HP99]. These wires limit the usage of the sensors; because the wires limit the movement of the objects; if there is more than one object in use, the wires may clutter the programming environment. Most electromagnetic systems that use three dimensional tracking suffer from this aspect, for example the Flock of Birds system from Ascension [Asc07].

Radio frequency identification(RFID) -based recognition of objects is another implementation of electromagnetic sensing that could be used to create a tangible user interface. A RFID tag is an object that can be identified by using radio waves. RFID tags usually contain a small silicon chips and an antenna. RFID tags are read with a RFID reader, which can sense presence and identity of an object. RFID tags are small, which allows "tagging" arbitrary sized objects by attaching a RFID tag to it. RFID based systems are relatively easy to develop, because the readers are commercially available and the cost of a single chip is often just a few cents. RFID systems are also developed to have a large address space, for example a 128 bit wide address space would mean approximately $3{,}4*10^{38}$ different objects. Unfortunately most of available RFID readers are only able to identify the presence of objects, not their locations. Another difficulty that RFID based systems suffer from is the latency in recognition. The RFID chips have long response times which means that the detection of each chip can take anything between 1/100 seconds and 1/10 seconds [HP99]. This delay gets even worse when the reader must identify multiple objects in the same physical space. There are devices which are not even capable of working if multiple RFID chips are present. Other devices try to use different collision protection schemes, which delays already slow recognition by a large degree [HP99].

Simple magnetically coupled resonancers as presented in Figure 12 are probably the most efficient solution for implementing tangible user interfaces with electromagnetic sensing. The basic technology is similar to the theft prevention systems in shops. The objects resonate in the magnetic field and that resonance can be measured. These objects are relatively small, relatively cheap and do not need external electricity. This basically means that the resonator can be placed on any object unless the object is made of a material that is magnetic, blocks magnetism or is electrically conductive, as the reader might encounter interference from these materials [HP99]. All tags used must be tuned to resonating at different frequencies. This allows the system to determine the presence and location of all objects independently. If the whole system is well built, this technology can be used to get stable position data which is accurate to within 4mm [PRI02].

If the object is coupled with two different resonators, the software can use the locations of the two resonators to calculate, in addition to the location of the object, also the orientation [PRI02]. Figure 13 shows a cross-section of such an object. The second resonator (B) is coupled with a button so that the resonator can also be used to transmit

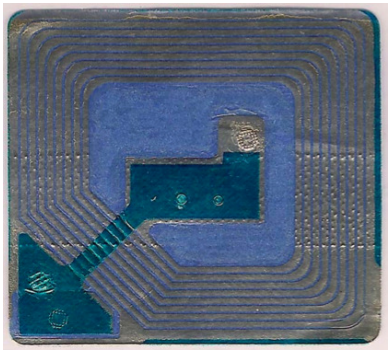

Figure 12: Simple resonance object [Fle02]

data about the state of the button by switching the second resonator on and off. The other resonator in the object (A) is always active, so the system does not lose the location of the object even when the button is pressed.

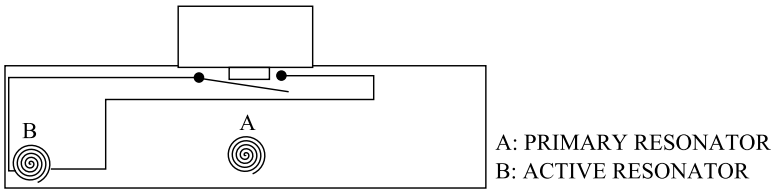

Figure 13: Resonator based active object

Systems created with simple resonators can have at least 30 uniquely identifiable objects [HP99]. Using RFID tags the system could easily have a countless number of different objects, but as mentioned before the speed of RFID technology is not enough for tangible user interfaces with small latency requirements. The resonance based system developed by a team at the MIT Media Lab [HP99] was able to establish refresh rates of over 30 reads per second, while recognizing all the objects on the surface on every refresh.

The problem with the resonator based systems is that such interfaces are not commercially available. Without knowledge about electronics, proper blueprints and facilities suitable for construction, the process of constructing an interface based on simple resonators is difficult if not impossible.

3.1.2 Computer Vision

Computer vision is an area which has been widely researched. The first uses of computer vision date all the way back to the work of L. Roberts in 1965 [Ull02]. The first use of computer vision in the context of tangible user interfaces was the Digital Desk in 1993 [Wel93]. New technologies for shape and object recognition are being developed actively all around the world.

A lot of the visual recognition research focuses on Mixed Reality research. Mixed reality requires the locations of objects to be recognized in three dimensions, which adds additional difficulty in the recognition process. In a tangible user interface environment the visual recognition is usually done in only two dimensions and because of the reduced complexity, the object recognition algorithms have been getting faster and more reliable [BKJ05].

The most common technology for giving objects a visually identifiable identity is to embed a sort of visual machine readable graphical marker into them. The term that is often used when talking about these visual cues is *fiducial markers* or just *fiducials*. An example of one sort of fiducial from the ReacTIVision toolkit [KBBC05] can be found on Figure 14. ReacTIVision based Reactable is shown on figure 1 on page 7. EAN bar codes found on products in shops can also be classified as fiducial markers. Fiducials are designed to be identified purely by observing their internal topological structure. If the fiducial is designed to be usable as a tangible interface marker, it usually has graphical structures that allow the computer to recognize, in addition to the location information, also the orientation of that fiducial.

Figure 14: Example of ReacTIVision markers [BK05]

When an object is tagged with a fiducial, the location of that fiducial must be selected such that the camera capturing the image for the visual recognition will be able to see

it. If the fiducial is added to a side that is visible to the user, it may distract the user by making it more difficult to differentiate the objects from one another. One way to limit the visual disturbance and give more freedom to the design of the tangible object is to place the visual cues at the bottom of the object. This requires placing the object on a transparent surface, so that the camera used in object tracking can see from the below. This also prevents the user from blocking the visual recognition by moving their hands between the camera and the visual cues.

Another approach in visual recognition is to use the object itself as an identifier. Software can observe the color and shape of an object and classify and identify it by those details [DGR04]. That approach works well and is algorithmically pretty fast, but this technology has limits concerning the number of different objects that can be reliably recognized. Because of the limits in the visual scope of the recognition camera the system cannot have very large objects. The differences in color must also be distinct enough for the system to recognize the changes even in different lighting conditions. Size and color both limit the number of different objects the system can reliably recognize.

Benefits of computer vision exceed those of electromagnetic technologies in terms of the cost of created objects. Fiducials, for example, can basically be printed with a normal printer and multiple fiducials can be fitted to one sheet of paper, so the real cost of a single fiducial is nearly zero. If a fiducial gets broken, dirty or otherwise unreadable it is an easy and cheap process to create a new fiducial just by reprinting it [CSR03a]. Creation of a new copy of any electronic tag would be much harder, since, for example the id of an RFID tag cannot, in most cases, be changed; the change of an object probably demands a change in the software, or at least in the configuration. Another positive aspect of computer vision systems is that they do not need any special hardware for doing the recognition work. A basic, standard, off the shelf web camera is enough for the system to work [CSR03a].

The size of the objects, the resolution of the camera, and the size of the surface all have an effect on each other. Large objects are easier for the visual recognition software to recognize, even with a camera that has poor resolution. A large object size limits the number of object that can fit on an area visible to the camera. If the size of the objects is smaller, then either the camera must be closer to objects, which limits the active area, or the camera must have better resolution which requires more computing power.

3.1.3 Electrical Contacts

The *electrical contacts* - type of input works by connecting interface objects to each other. These connections form a structure, which is the interface for that system. Electrical contacts give the processing logic the necessary data about the presence, locations and attributes of connected objects. The processing logic may be implemented inside the objects or it may be located on a computer or another device connected to one of the objects. Some object designs use connectors that have combined data and power channel, which keeps the connectors relatively simple as the number of necessary concurrent connections is small. This gives the connectors more robustness compared to the designs in which the power and data are delivered by different connectors.

Designing the contacts and boundaries for objects is said to be the most difficult part of the design process of an object set that uses electric contacts [GOI98]. The objects must be designed so that they only connect at the correct places and at correct orientation in relation to other blocks. Figure 15 shows contacts of an ActiveCube -block [KIMK00]. ActiveCube blocks can be connected to each other by any of their sides. The orientation of an object does not matter as the objects only stick together at right-angle. Another example of a connection method is triangles [GOI98], as shown in Figure 16. Triangles are triangular objects that can be connected by any of the three faces. The connection is fixed and held together by strong magnets. Tangible Programming Bricks and iArtefacts shown in Chapter 2.2 (Figure 8 and 9 on page 15) both stick together by standard LEGO® -connectors. Tangible programming bricks have dedicated connectors for data transfers between the structural connectors, so the data and electricity are kept on different areas of connectors which limits the possibility of misconnections.

The Tobobo system from MIT gives the builder the opportunity to decide which blocks are connected and in which order. The electric contacts are not connected at the time of building the actual structure. Builders must attach a cable between every block they want to connect as Figure 17 shows. This solution is two-fold. On the one hand, it allows the builder to create more diverse structures as the connectors are not limiting the size and the form of blocks. On the other hand, the need for external connecting wires makes the end product less attractive to children and unnecessarily complicates the process of programming the structure [Raf04].

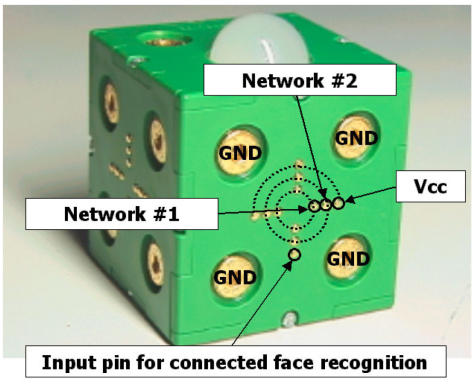

Figure 15: ActiveCube contacts [KIMK00].

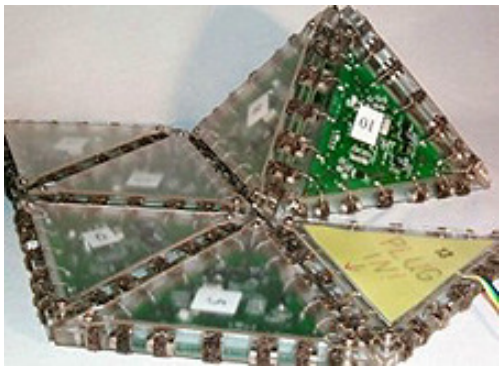

Figure 16: Triangles can connect by any of their sides [GOI98].

Figure 17: Topobo creature with wires connected

Most of the active blocks that work by using electrical connections need a considerable amount of electronics inside them, which in turn raises the production costs of these objects [McN00]. In the prototyping phase the cost of the electronics may rise even higher, since all objects must be hand built. The price of the constructed blocks may be the main reason why there are not yet commercial products that would use electronic contacts in the context of building things.

3.1.4 Wireless Communication

Even though electromagnetic sensing and computer vision both are wireless techniques they are not included as part of the wireless communication class. This group relates to direct wireless communication from one TUI object to another. Wireless technologies are not really widely used in TUI development as the technology basically requires the objects to be active, and thus to have loads of electronics inside. This raises the cost of a single object and makes them unfeasible to fabricate.

A group of researchers from American University of Sharjah created a wireless system that uses IEEE 802.15.14 standard compliant wireless radio communication to send data between the objects and the controller [ZAAM+07]. The system is not able to tell the locations of objects and thus is not ideal for most of the tangible user interface designs. On the other hand Audiocubes [Sch04] are designed to use infrared to communicate between the user interface objects. Audiocube objects recognize the distance and orientation of objects in use. The system is in the prototype phase so the processing is done in an external box, which is connected to the objects with wires. The Massachusetts Institute of Technology also created a project in which infrared communication was used to recognize and locate objects on a table [BBMM02].

Altogether the use of wireless communication is not yet a feasible technology selection for tangible user interface creation. If the cost of wireless network communication equipment comes down enough it may change the situation.

3.1.5 Toolkits for Tangible Interfaces

As previous sections have shown, there are different methods such as electromagnetic sensing, computer vision, electrical contacts and wireless communication for getting

data from the physical user interface to the computer. The problem with using these technologies is that they usually require the tangible user interface developers to "get down and dirty" with the technology [KLLL04]. That requires knowledge about specific areas of electronic engineering (for electromagnetic sensing), pattern recognition (for computer vision) etc. making these UI's difficult and time consuming to program. It is also important to remember that the developed input method must be robust and the latency cannot be too high [CSR03a, BKJ05].

Using ready toolkits for the object recognition in tangible user interface development enables people who are not experts in, for example, hardware development or visual image recognition technologies to develop tangible user interfaces, as the GUI-toolkits have helped programmers who are not graphics hardware experts to write working GUIs. Similar reductions in development time can be achieved in tangible user interfaces as in graphical user interfaces by using toolkits [KLLL04]. Toolkits help in reducing the amount of code that programmers need to produce in order to create a working user interface and enables more rapid prototyping and more iterations in the design process.

In the previous studies about tangible user interface toolkits there have been discussion about proper evaluation criteria for selection of tangible user interface toolkits [KLLL04, BKJ05]. The following list shows the main evaluation points that should be considered while selecting a toolkit:

- How readable are the programs, developed by using the toolkit, for other programmers?
- How comprehensible are the input mechanisms that the toolkit allows for the end users?
- How easy is it to learn to use the toolkit?
- How scalable is the developed system in the terms of the number of uniquely identifiable objects?
- How robust is the toolkit in terms of recognizing and tracking the location, orientation and identity of objects?

Table 2 lists the toolkits that were tested for this thesis. The toolkits will be introduced in detail in following paragraphs.

Table 2: Properties of Tangible User Interface toolkits

Name	World	Type of input	Language	OS
Papier-Mâché	2D	Obj. Classifier	Java	Any
ARToolKit	3D	Fiducial Rec.	C	Linux, Mac, Windows
D-Touch	2D	Fiducial Rec.	C	Any
ReacTIVision	2D	Fiducial Rec.	C	Any
Wireless	-	Wireless	-	Windows
iStuff	-	N/A	Java	Any

Papier-Mâché [KLLL04] is an open-source- licensed, Java based toolkit, which can be used for visually recognizing objects by classifying them by their visual properties, or by using RFID tags for object recognition. The possibility of using RFID tags in addition to visual classification gives programmers the flexibility to easily change the input method. The programmer then has the option to first quickly prototype the system with visual object recognition and then build a more robust system with RFID technologies. The Java programming language makes the generated applications relatively easy to comprehend even without previous knowledge about the toolkit. The toolkit documentation has good examples of different types of tools it provides for object recognition, which makes it relatively easy to learn. Visual recognition based classification limits the use of this toolkit as a basis for a tangible user interface as the number of reliably recognizable different objects is relatively low. The information gathered from the recognized objects also contains only the identity and location of them.

ARToolkit [KBP+00] has been developed, as its name suggests, as a toolkit for Augmented Reality. It tracks fiducials in 3D-space, and gives users locations of those fiducials. The toolkit is developed for overlaying computer generated information to places where the fiducials have been calculated to be. That use is not general enough for use with a wide variety of different tangible user interfaces [KLLL04]. The toolkit is relatively easy to use with projects and the generated software structure is easily understandable for people that have programmed before. The system can keep track of multiple objects at the same time. The toolkit can recognize the identity, location and orientation of visible objects.

D-Touch [CSR03b] toolkit is also a toolkit for visual fiducial recognition. The difference to ARToolkit is that the recognition is done in only two dimensions. This gives more speed and reliability to the recognition and the toolkit can recognize smaller objects. From the developers point of view the connection between the D-Touch toolkit and new programs are relatively easy to implement. The toolkit can recognize the identity, location and orientation of visible objects.

ReacTIVision [KBBC05] is a third toolkit that tracks fiducials. Figure 18 shows the toolkit in action. The toolkit is an external program that aids the work of the programmer by communicating with the developed software through TUIO protocol. The TUIO protocol is a result of work in developing a common open protocol for Tangible User Interfaces, that would be versatile enough for all necessary communication tasks [KBBC05] and which would simplify the communication between the physical interface and generated software. The website of the toolkit [Rea07] has, at the time of the writing of this thesis, sample code snippets for TUIO implementation for the following languages: C++, Java, C#, Processing, Pure Data, Max/MSP, Flash and SuperCollider. On the Microsoft Windows platform the software can use almost any video input device that has proper Windows Driver Model compatible drivers. The ReacTIvision framework is open source software and thus free for use and further development. The toolkit can recognize the identity, location and orientation of visible objects.

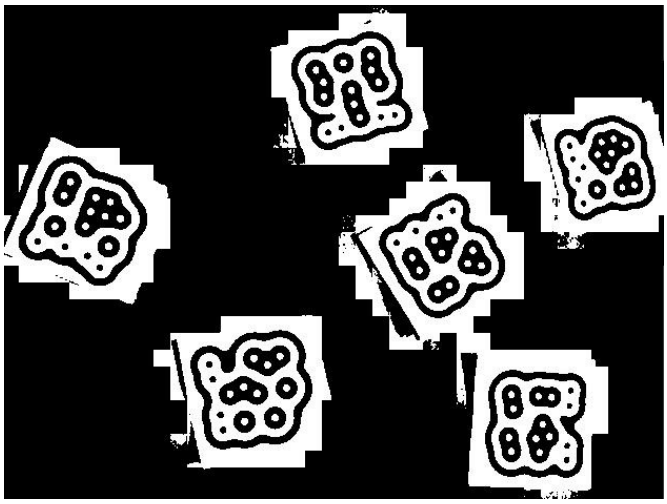

Figure 18: reacTIVision recognizes fiducials from video stream

Table 3 presents a recognition speed comparison of ReacTIVision toolkit and D-Touch toolkit. The table does not include ARToolkit, as it recognizes the objects in three

dimensional space. As the results show the recognition algorithms have advanced. The results are from tracking 12 fiducials at the same time [BKJ05].

Table 3: Speed comparison of algorithms for recognition of fiducials [BKJ05]

Toolkit	Library	CPU Usage	Frame Rate	Year
D-Touch old	D-Touch old	100%	10 fps	2003
D-Touch	libdtouch	82%	30 fps	2005
ReacTIVision	libfidtrack	18%	30 fps	2005

The **wireless Toolkit** [ZAAM+07] is a toolkit that is designed to use the IEEE 802.15.14 standard based wireless networks for tangible interaction. The object network is star shaped, which means that there is one *master node* that controls the communication and a number of *child nodes*. The child nodes are fitted with optical sensors, so they recognize their orientation. The wireless toolkit does not give information about the spatial locations of objects. The toolkit is flexible in terms of the hardware used and does not require any physical connections between the sensors. The objects must be hand built as there is no commercially available equipment for child nodes. The controlling master node is a standard communication device and can be seen in Figure 19 with a couple of objects that can be used for executing searches on the Internet.

Figure 19: Wireless toolkit used for searching Barbie dolls [ZAAM+07]

The **iStuff** [BRSB03] framework is dissimilar to the other frameworks that were introduced. The iStuff framework is meant to act as a hub for connecting multiple digitalized everyday objects as an interface for a computer. Communication between the

objects and software goes through proxy software which translates all actions received from the objects into a common format that the software can understand. Any physical interface can be part of the iStuff hub as the developers of the iStuff framework explain: "All that is necessary for a physical device to become an iStuff component is a proxy that encapsulates data into an event" [BRSB03]. This allows programs developed within the iStuff framework to be independent of the limitations of any particular technology.

3.2 Output

Input in tangible user interfaces is handled through the physical objects, and in most cases the objects are also part of the output. However the objects themselves, if not active, have a hard time representing data or giving feedback to user. A tangible user interface requires therefore a method for output. The following sections will demonstrate the most widely used technologies for implementing the output in a tangible user interface.

3.2.1 Displays

The computer generated graphical output is still a part of the tangible experience. Graphical feedback in a tangible interfaces is often ambient [UI97] and georeferenced [KLLL04], which means that physical input and graphical output happen within the same space. Figure 20 shows an example of georeferenced graphics. This georeferenced output reinforces the perceptual link between the physical and virtual objects. Grasping an object becomes analogous to grasping the corresponding piece of digital information [CSR03b].

Display techniques can be divided by their physical aspects into four different categories: *external displays, active displays, projection* and *embedded displays*.

The use of completely external displays is not well suited for tangible user interfaces. Since the display is not directly related to the user interface, the graphical output is not georeferenced and therefore the mapping between graphics and the physical handles must be done by the user in his/her mind [RUO01].

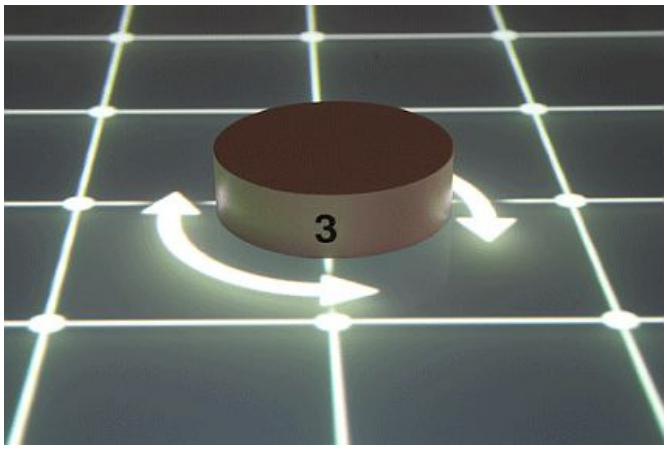

Figure 20: Graphical output may be georeferenced [GH07]

Large LCD/Plasma screens that are mounted into the table can be called active screens. Users place the tangible user interface objects directly onto the display surface. This allows georeferencing, so the graphics can be coupled with the real objects. The surface of the display limits the choices that can be made when selecting the method for sensing the locations of the objects. For example the DataTiles system shown in Figure 21 uses a flat panel display enhanced with sensors to allow the interface to recognize tiles and their locations on the display [RUO01]. Visual recognition can also work with this LCD/Plasma screens, if it is possible to place the fiducials on top of the physical objects.

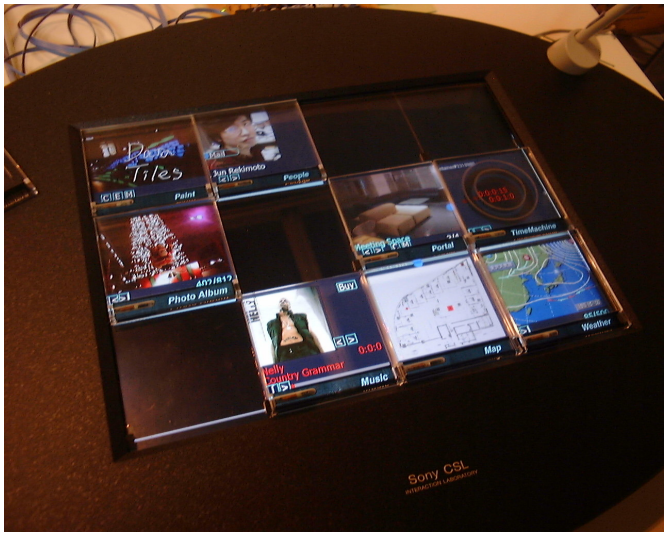

Figure 21: Data Tiles are placed on a large display [RUO01]

Projection based systems are versatile and allow large image sizes, but require more structural construction than using displays that consist of one physical part. The projector always needs a surface on which to project the image. That surface must be firm enough to be stable even while the user interface objects are placed on it. Projection systems can be built in two different ways. The projector can be mounted above the system and then project the image down to the screen, or the projector can be mounted below the system and, with the use of mirrors, the image is projected to a rear-projection screen which works as the user interface surface. Top projection is the only available solution in situations where the bottom of the screen does not allow light to pass through it or the object recognition system is installed below the surface and would interfere with the picture. Bottom-projection on the other hand allows the image to be seen all the time. The user cannot block the projected image and there are no shadows from the objects or the user [Ull02].

An embedded display means that the display is embedded inside the objects of the interface. This method can be used if the interface is based on active objects. Active objects also have other options for output, this is discussed in detail on Section 3.2.2.

3.2.2 Active Objects

If the user interface is constructed with active objects, it is possible to use those objects for information output. The objects can have integrated displays, which can be used to give information about the status of the system and the results of the actions the user has done [Ull02].

The output device in this context does not necessarily have to be only a display. Motor based actuators like meters, wheels and other moving parts can be considered as output. The system can also have embedded lights or speakers for output. For example McNerney's Tangible Programming Bricks [McN00] use embedded displays and control other devices, as Figure 22 shows, to output information and thus do not need external display elements.

Active Blocks [Nie02] have a set of different output options including motor controller bricks that can be used to drive LEGO® Technic-series motors, servo bricks that can be used as actuators, led bricks that can be used to show status information, LCD bricks

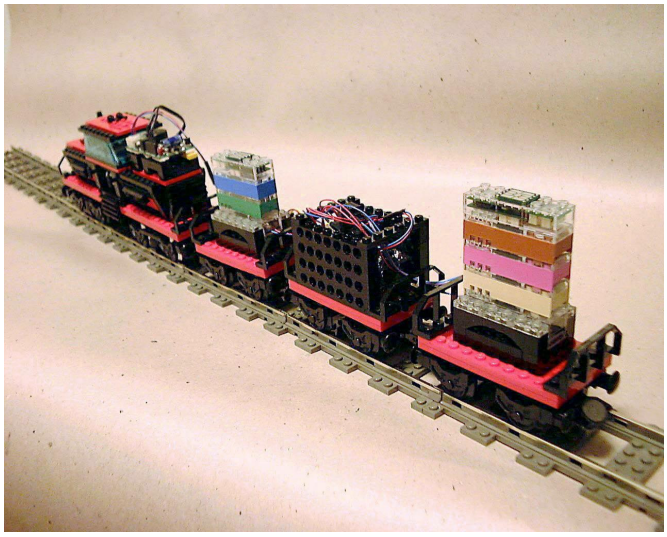

Figure 22: Tangible Programming Bricks do not need displays [McN00]

that can show values on a 2x8 character LCD display and sound bricks that can be used to either play a pre programmed tune or a tone at a given frequency.

3.2.3 Object Movement

Physical manipulation of objects for input has a major drawback in the terms that usually the output is only graphical[PMAI02]. One common form of modifying the input in computer software is the possibility to undo the previous action. On tangible user interface without any means for a computer to control the physical world, problems arise. The locations of the physical objects do not represent the same state as the computers internal representation after undo. There are however possibilities to establish a connection back from the computer realm to the physical world.

The object itself can move if it's an active object and has some energy to spare. This basically means that each of the objects is a small robot like in Planar Manipulation Display [RZSP04]. Figure 23 shows the Planar Manipulation Display in use. The communication between the controlling software and the objects is done by sending infrared commands to the robots.

Another possibility for establishing object movements is to give the surface a method to be able to move the object by applying some force, for example a magnetic field [PMAI02]. The surface of the Actuated Workbench on Figure 24 is designed to have

the ability to move an object on the surface to other location. The control of the object is acquired by creating a variable magnetic field over the surface. This method requires more from the surface, but allows the design of the objects to be simple.

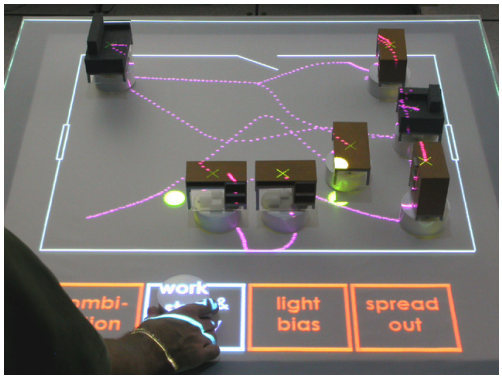

Figure 23: Planar Manipulation Display uses tiny robots to actuate the user interface [RZSP04]

Figure 24: Actuated Workbench can move physical objects on surface [PMAI02]

3.3 Summary of Technologies

Table 4 presents a comparison of traditional technologies which have been used as input mechanisms for tangible user interfaces. The values low-moderate-high used in the table are only meaningful in comparison to the other technologies in each row. Some of the categories do not apply to the electronic contacts class and have thus been marked as non applicable. Wireless communication is left out of the list, as using wireless technologies is still difficult and there are no toolkits that would help the designer in the implementation of wireless communication. The aspects through which the items in the table are compared are those which have a large effect on the creation and operation of a tangible user interface.

Tangible user interfaces need in addition to the input, some method to output data. One option for output is the traditional way: display graphics or text on a display. Table 5 lists different types of displays with the estimated cost of implementing a tangible user interface using that technology in one column and a estimate of the lightlihood of displaying georeferenced graphics in another.

In addition to using displays as output devices if the user interface is designed around active objects, the objects themselves can be used for output. Depending on the design

Table 4: Comparison of Input Technologies

	Input method					
	Visual recognition		Electromagnetic		Electronic	Wire-
	Classify	Fiducial	RFID	Resonator	contact	less
Latency	moderate		high	low	low	high
Precision	moderate		low	high	N/A	low
System cost	low		moderate	moderate	high	high
Object cost	low		moderate		high	high
Commercial	yes		yes	no	no	no
Toolkit avail.	yes		yes	no	no	no
Object count	low	moderate	high	moderate	N/A	low

Table 5: Display types

Display type	Georeferenced	Cost
External	No	Low
Active	Yes	High
Projection	Yes	High
Embedded	N/A	High

of the objects, they can include motors, beepers, displays and other actuators and output devices. Object movement is one of the most interesting output methods as it allows actions like "undo" on the tangible user interface. Unfortunately the technology that would allow creation of such devices is expensive and requires a lot of construction to implement.

4 The Development Process of the TangibleProgrammer Prototype

In this Chapter the creation process of a tangible user interface prototype called TangibleProgrammer is described. The results and comments in this Chapter form the answer to research question Q3: How difficult it is to implement a tangible programming interface? What problems can be expected while developing a tangible programming interface?

I got the first idea about using tangible user interfaces as a programming tool from presentation at ICALT2004. The presentation described the use of ARToolkit in a project that was used to teach molecular physics [FHW+04]. Tangible musical composition devices [PRI02, KOC04] have also given ideas and a framework to this project.

The requirements for the TangibleProgrammer were set on the Section 1.1: Problem Definition as follows:

- The user interface must be able to recognize and keep track of the physical objects.
- By using the interface, the user must be able to create programs that will be compatible with a LEGO® RCX unit based robot.
- The system must be able to move the created software to a LEGO® RCX unit based robot for execution.

Tangible user interface design is mostly defined by the input technology. As Chapter 3 on Interaction Technologies explained, the recognition of physical objects is easiest done through the use of a toolkit. In Section 4.1 I will explain the history behind the toolkit selection for TangibleProgrammer. The next step in the development process is to begin the development of the actual software for TangibleProgrammer, Section 4.2 concentrates on that process.

In addition to the software, the tangible user interface also requires a physical setup. TangibleProgrammer cannot be truly tangible without the physical structures and objects that define the user interface. Details of the development of the physical part of TangibleProgrammer are explained on Chapter 4.3. In Chapter 4.4 the completed

TangibleProgrammer prototype is introduced and it's physical details and functionality explained.

4.1 Toolkit Selection

Acquiring information from a physical context is the most important aspect of a tangible user interface. If the user interface is not physical, it is not a tangible user interface. In Section 3.1.5 the process of recognition of physical objects was found to be a difficult task that would be best left to a toolkit. Therefore the first task of a developer is to select the toolkit that will form the basis of the project. The requirement of using a toolkit limits the options of possible interface technologies, since there are no toolkits available for interface technologies based on electrical contacts or electrical sensing based on resonancers.

For the TangibleProgrammer all the toolkits that were introduced in Section 3.1.5 were considered as possible tools for the user interface creation. iStuff-toolkit and the Wireless toolkit were however left out of further evaluations, as the development of a working user interface in both toolkits requires hardware that is not commercially available.

The first toolkit that was tested for suitability of user interface creation was Papier-Mâché [Kle03]. The process of specifying different objects in the Papier-Mâché environment was relatively easy. However the limitations of the visual recognition system (that recognizes objects by their color and proportions) was too limiting for TangibleProgrammer. Programming environments can require tens of objects in use at the same time, so that each object is individually distinguishable from the others, as each object usually converts to no more than one line of programming code. This means that object recognition system should be able to scale to a large number (at least n>100) of uniquely identifiable objects in order to keep all the code objects unique. Identification by classification limits the number of unique objects and therefore the Papier-Mâché is not the correct toolkit for the creation of the TangibleProgrammer.

The second toolkit that was tested was ARToolkit. It allows a much wider range of objects as it uses fiducial-type graphics in the recognition process so the object count is not as limited as the classification based approach. ARToolkit is designed to work in 3D space and thus the recognition is a bit slower than in D-Touch and reacTIVision

which work in 2D space. Therefore either D-touch or reacTIVision toolkit would be a better option for the TangibleProgrammer.

Both toolkits, D-touch and reacTIVision, are fiducial recognition based, but the reacTIVision is remarkably faster than D-touch, as Table 3 on page 31 shows. Therefore the reacTIvision toolkit was selected as the basis of the TangibleProgrammer. The reacTIvision toolkit is a standalone program that does the image recognition through the use of fiducials, so the number of identifiable objects is high enough for the TangibleProgrammer project (n>100). The toolkit communicates with the rest of the program by using network sockets. This network communication allows installation of the visual recognition part of the user interface to one computer and the rest of the software can be installed onto another in case the TangibleProgrammer is running too slowly.

4.2 Software Creation

One of the requirements for the TangibleProgrammer that were indicated in the research questions was that it must be able to communicate with the LEGO® RCX unit based robot and create software that runs in the robot. Software that LEGO® offers for programming RCX based robots (LEGO® Robotics Invention System [LEG07b] and LEGO® RoboLab [LEG07a]) are both proprietary tools and thus hard to modify. This excludes them from consideration as the base of TangibleProgrammer. There are also open source options for program development and communication with the LEGO® RCX unit. Those tools are *Not Quite C*(NQC) and *leJOS*. Not Quite C is a limited subset of the language C that can be compiled to a format that is binary compatible with LEGO® RCX firmware [NQC07]. leJOS on the other hand is a tiny open source Java Virtual Machine that is designed to run inside a LEGO® RCX unit [leJ07]. It replaces the original firmware of the brick and allows users to run simple Java programs in them. Both languages could be used as the basis of communication with TangibleProgrammer. LeJOS is however a better choice for this project as the Java language that leJOS uses is easier to implement than NQC.

As I am most familiar with the programming language JAVA and as Section 3.1.5: Toolkits for Tangible Interfaces mentioned, it is possible to use reacTIVision toolkit through the pre-made TUIO compatible JAVA classes, the selection of the program-

ming language was easy. To establish a connection between the TangibleProgrammer-software and the reacTIVision toolkit TangibleProgrammer uses TuioClient class by implementing the TuioListener type through the following methods (JAVA):

```
public void addTuioObj(int id)
public void removeTuioObj(int id)
public void updateTuioObj(
        int id, int c, float x, float y, float a,
        float X, float Y, float A, float m, float r)
```

Through these methods the program gets information of all the objects entering (addTuioObject) and leaving (removeTuioObject) the surface as well as information about the changes in the object location (updateTuioObj), including the speed of change in location and angle.

Figure 25 shows the basic internal structure of the TangibleProgrammer. An image stream coming from the web camera is interpreted by the ReacTIVision framework. TUIO protocol transfers the interpreted data to the software logic. The toolkit gives software logic information about the identity and locations of all the objects visible to the camera. This identity is the identification number of the recognized fiducial. Software logic includes a mapping table between the fiducial id number and the user interface object.

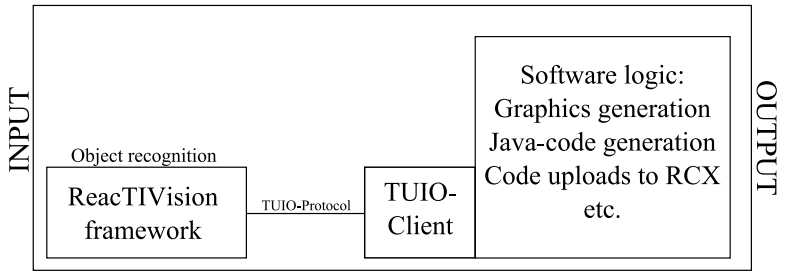

Figure 25: Software structure of TangibleProgrammer

There are 2 main types of objects in the TangibleProgrammer user interface, code and modifier objects. Code objects are used to create new Java code. Modifier objects cannot be used only alone, but are used in addition to any of the code objects that have some parameters which can be modified. A full list of implemented objects is presented on Section 4.4.

As the user interface objects have parameters that are not visible in the physical objects, the user must be able to see the values somehow. The system must therefore be able to output some simple graphics that are georeferenced to the physical objects and can be used to augment them. Drawing simple shapes with Java is relatively easy, but the system must allow rescaling and repositioning of the graphics so that after calibration the graphics will be in sync with the physical objects throughout the whole surface. This use of graphics requires some form of display device, but that will be discussed in the Section 4.3.

Programs created with TangibleProgrammer are saved to a file, run through the Java compiler which compiles the created program to a Java byte code file, which the LEGO® RCX unit can then execute.

4.3 Hardware Preparation

The hardware in TangibleProgrammer was selected so that it would cope with requirements set out by the software. The selected toolkit uses image recognition as the input method. For a basic image recognition using the reacTIVision toolkit any relatively modern computer is fast enough as Table 3 on page 31 shows. Recognition also requires a device which is used to acquire an image of the objects that must be identified, which can be done with basically any web camera.

The following ideas come from the developers of ReacTable [Rea07], the camera to be used with reacTIVision was selected so that it could be modified to capture images in infrared frequencies. The process of modifying a typical web cam for infrared illumination is relatively easy [Ins07, Lun07a]. The technology of the web cam allows it to receive infrared-images, but the manufacturers usually install infrared filters to avoid interference affecting the visible light. That filter must be removed in order to allow the camera to capture infrared image.

The display system was selected to use projection. The projecting image allows visual object recognition and large displays while keeping all the equipment under the surface of the user interface. This removes the possibility that the user of the interface would create shadows in front of the image recognition or the image projection. An infrared filter was installed on a video projector that was chosen to be used in this project. Infrared filter limits the visibility of the projected image to a camera, so that the image

of the projector is clearly visible to users of the interface, but the web camera used with reacTIVision software is not disturbed by the projected image.

The fiducials on the user interface objects must be lighted evenly for the visual recognition to work well. For the first tests with the reacTIVision toolkit the infrared light emitter was built from 24 infrared LEDs. Unfortunately the beams of light emitting from the LEDs were too narrow. The maximum intensity of the light was more than enough to light the surface, but the narrow beam of the LED meant that there was brighter and darker spots on the image, which distracted the image recognition system. The second option for the infrared light was an incandescent lamp, which outputs infrared in addition to visible light. The final lighting was selected to be done with a standard 40W incandescent light bulb, which seemed to work well with the recognition system in the development environment. The selected lamp gave out enough infrared light for image recognition to work. The only drawback with this approach was that the lamp also gave out light in the visible spectrum, which washed out part of the brightness from the projected computer graphics.

The next part of the hardware preparation was the readying of the table surface. As the image display technology was previously selected to be image projection, the system needed a surface onto which the projector could project the image. Another necessity with the surface was that the camera should be able to see the backsides of objects placed on it. So the surface needed to be semi-transparent. An easy and cheap solution for that was to use a surface made of glass and cover the surface with a diffuser. This allowed objects close to the surface to be clearly visible from below, but also allowed the system to project the image onto the same surface. Because the video projector that was used in this prototype was meant for classroom use, its optics were not well suited for distances below two meters. This was fixed by using a mirror so the projector could be placed further away and the image was then projected via the mirror. Figure 27 on page 47 explains the structure of the system.

As the design of the tangible programming interface would not be very tangible if the users would have to use the computer to send the program to the robot, the sending process was started with a large button that was attached to the display surface. The button connects to the computer through a keyboard adapter, so it works like a giant Enter-button on a keyboard.

When the user has added the necessary code objects to the program and adjusted the variables as needed the program is ready to be transferred to the LEGO® robot. When the user wants to compile and send the program to their robot, the system first goes through all the objects in the program and collects corresponding Java code for each object. This code is then sent to the Java compiler which compiles it to Java byte code. The compiled byte code is then sent to the robot through the LEGO® infrared tower via leJOS.

The design of objects and selection of appropriate physical representations is an important aspect in tangible interface design[UI01]. Optimal design of an object should take advantage of humanly graspable concepts like size, balance, texture weight, density and temperature [DGH03, KOC04]. These might help the user to figure out which objects he is allowed to place at which point and what is the meaning of each object. Blocks that seem to fit together easily would be meant to interconnect. In the first plans the blocks were designed to be made out of wood, but due to time constraints they were made out of cardboard. The blocks however had a wooden ”core” so they had some weight and were not so keen on going out of the constraints of the recognizer. Figure 26 shows the structure of an object used in the TangibleProgrammer. Layer A in the figure is the fiducial that is required for object recognition. Layer B is the wooden core and layer C is the icon visible to the user. The borders of the area in which the object can move, are masked, so that the users are not too keen on moving the objects out of the visible field of the camera [KOC04]. This gives the users of the system haptic and visual cues about the area in which the object can move.

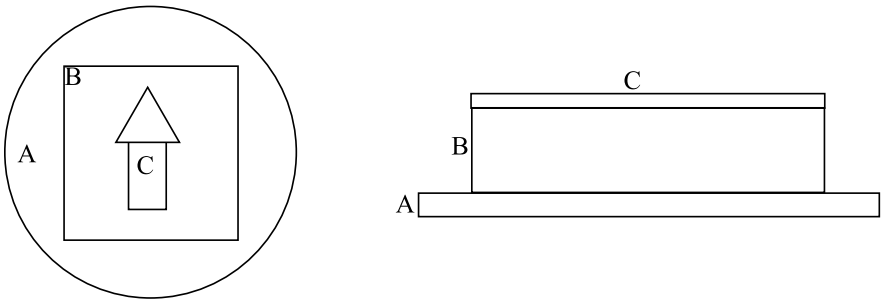

Figure 26: Physical structure of an object. Left: From top, Right: From front

4.4 Completed TangibleProgrammer Prototype

The prototype consists of 2 parts: Computer software, which is driving the system, and the physical interface. Figure 27 shows the cross Section of the setup and Figure 28 shows the completed machine. The surface on which the objects rest is one of diffusing glass, which allows the fiducials attached to the bottom of an object (2) to be visible to a web camera below (7). The same diffusing glass surface works also as a projection screen for the video projector (5). The projector's image is projected through the mirror (6) attached to the structure of the system. The mirror is used because the data projectors have some minimum distance from which they can project sharp image and the distance between the projector and the surface would be too short without a mirror.

Other equipment that is necessary for the TangibleProgrammer to work includes a lamp (8) that is used for giving the fiducials some light which enables the camera (7) to see the fiducials even when the projector is showing blank picture. Button (1) is attached to the surface and is used for starting the compilation and upload process. The upload process requires a communication method between the LEGO® RCX unit and the computer. This communication is done with a LEGO® infrared tower (4). The tower uses infrared to transfer the compiled software to the LEGO® RCX unit (3).

As the TangibleProgrammer is still a prototype, its function set is not computationally complete. The set of available commands are tuned to work well with simple LEGO® RCX unit based robots that have either 2 or 4 wheel drive and in which the turning is done by controlling the throttle on the wheels on the left or right side independently. The system cannot yet be used for complicated programs or calculations, but this set of commands can be expanded in the future.

The command set available in TangibleProgrammer consists of two different classes of objects. Action blocks are user interface objects that create changes in final Java code that is transferred to the robot. The action blocks may have parameters that are changed by using the modification blocks as Figure 29 shows. The modification blocks are not tied to action blocks, but can be used to change the parameters of any of the action blocks in use.

In the following list all the developed objects are shown with a simple usage guide:

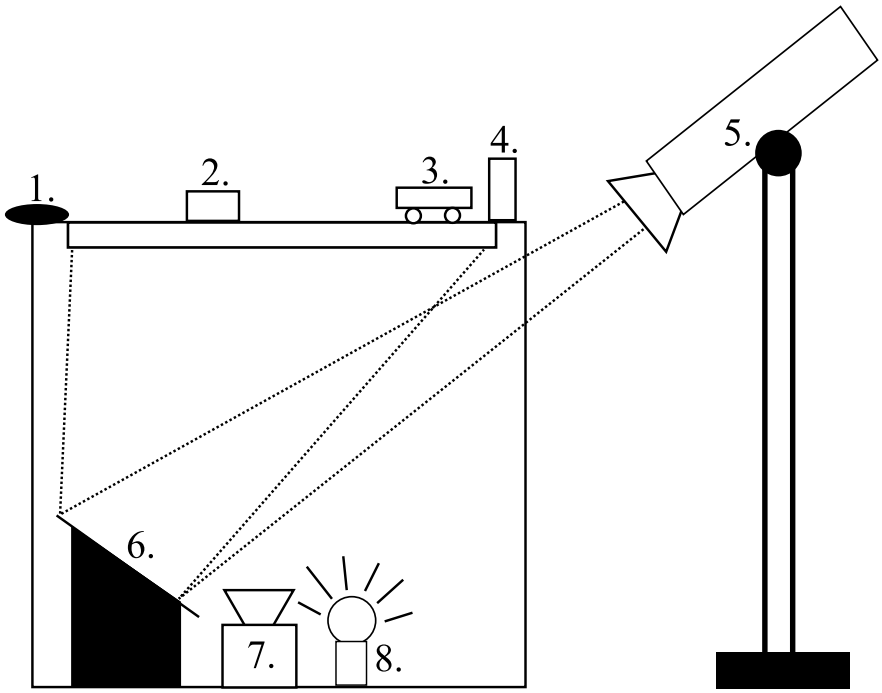

Figure 27: Physical structure of the TangibleProgrammer

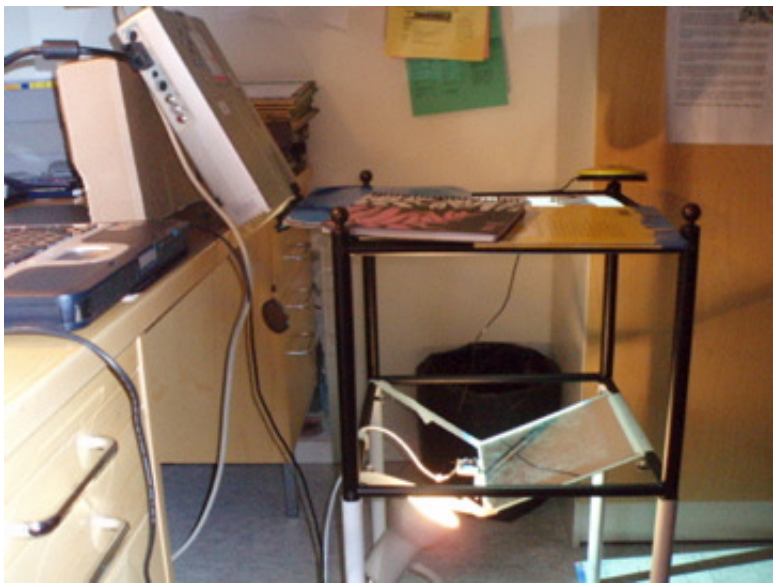

Figure 28: Prototype ready to be evaluated

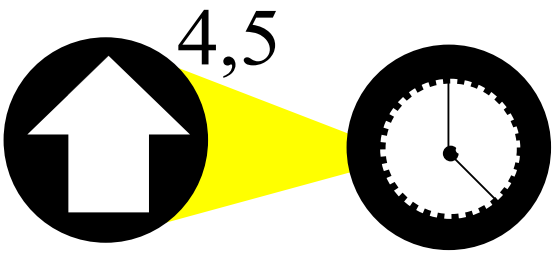

Figure 29: Time modification block changes the attribute of connected action block

Moving: Arrow block.
Can be inserted on the surface either pointing up or pointing down. The application recognizes the direction (up=forward, down=backward) and the program executes as required.
Adjustable parameters: Time of execution.
Adjusted through: Time modification block.

Turning: 2 Different turning blocks.
One for turning right by rotating the wheels on the left side of the robot forward, another for turning left by rotating the wheels on the right side of the robot forward. Turning blocks can be placed on the surface also facing backwards which changes the turning action to run the wheels backwards.
Adjustable parameters: Time of execution.
Adjusted through: Time modification block.

Delay: Block that can be used to delay execution for a given time.
Adjustable parameters: Time of execution.
Adjusted through: Time modification block.

Sound: Block that can be used to produce different sounds.
At the moment the only adjustable attribute is the length of tone.
Adjustable parameters: Time of execution.
Adjusted through: Time modification block.

Repeat: Pair of objects connected together with a string.
Objects are useless working in isolation. First block is used to mark the beginning point of the loop, the second to mark the ending point.
Adjustable parameters: Number of repeats.
Adjusted through: Numeral adjustment block.

Time modification block: Used together with another, compatible action block to change the time allocated to the action.

Numeral modification block: Used together with another, compatible action block to change numeric variable for the block.

Figure 30 shows an example program of the TangibleProgrammer. If the rotation time is adjusted to perform a 90 degree turn this program commands the robot to drive around a square. The time required for a 90 degree turn depends on the properties of the robot in which the program is executed. The background shows the area which the loop commands will execute. The command are read in order from top to bottom. Location of the objects in horizontal direction has no effect on the execution order.

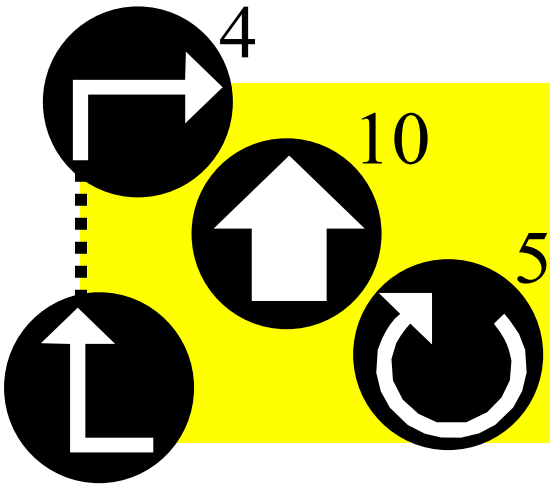

Figure 30: Example Program of TangibleProgrammer

5 Evaluation

This Chapter describes the evaluation of the TangibleProgrammer. This evaluation combined with the next Chapter: Conclusion and future directions, forms the answer to research question Q4: How well did the tangible programming interface manage to fill out the expectations and how should it be modified for continued usage?

Evaluation of the prototype was performed in a daycare center Pääsky in fall 2006. This daycare center had been a participant in the ”Technologies for children with individual needs” project since spring, 2006. Participants from the daycare center were children who were attending a preschool-class, which had a average age of 6 years. During the project the children had been taught how to build LEGO® Mindstorms robots and how to program very basic programs by using the LEGO® Robotics Invention System 2.0 [LEG07b] programming language. This allowed the evaluation to concentrate on the abilities of the user interface of the prototype. If the subjects used in the evaluation had been new to programming, a large part of the evaluation time would have had to be used for teaching them about programming and the time evaluating the prototype would have been shortened.

The task that was assigned to the children during the evaluation was to recreate the program code they had previously created with Robotics Invention System by using the Tangible Programmer prototype. Copying already created code instead of creating new programs was chosen as the task because of the requirements of the research questions. The question at this point was about the usability aspects of tangible programmer, not its versatility as a learning tool for programming. Copying the code allowed the children to concentrate on the usage of the prototype and less on in the process of developing new code.

The following list defines the planned evaluation process:

- **Preparation phase**: Modification of the groups robot to an RCX unit that already had leJOS installed.

  The children had built the robots themselves and already had programs in the embedded RCX units. The loading of the leJOS firmware to the RCX units would have erased all created programs and that could have created disturbance

in the group. The firmware upload process also takes at least five minutes, so by changing the RCX unit, the children did not have to wait for the upload process to complete. This could be one development idea for the next generation of TangibleProgrammer. TangibleProgrammer could check the existence of correct firmware in the background and if the firmware is missing, the system should ask the user to select whether the computer may overwrite all data on the RCX unit or not. The upload process could then be done in the background.

- **Introduction phase:** A short introduction of the usage of the Tangible Programmer.

  The children had previously programmed their LEGO® RCX unit based robots using the LEGO® Robotics Invention System 2.0, but the process of creating programs using the prototype differs slightly from the usage of that environment. Therefore hands-on introduction was performed at the beginning of each evaluation session. In the introduction, the users were introduced to the usage of the system, the programming methodology (by placing the objects on the surface), and the aspects of the time modifier block, which can be used to change the attributes of the objects.

- **Usage phase:** Helping children when they need assistance

  As the expectation of the system was that it would be easy to use, the children were supposed to be able to use the prototype with as little explanation as possible. Because the system was in the prototype stage there was always the possibility that the system would have bugs and usability problems might come up, in such cases giving help would be allowed.

5.1 Set up of Evaluation

The equipment was transferred to the daycare center in the morning of the day of the evaluation. Because of the prototype nature of the equipment some time was required for putting the equipment together and calibrating it before the evaluation could begin.

During the set up phase an unfortunate problem was noticed in the TangibleProgrammer. Images coming from the projector to the active surface were a bit hard to see.

That was a hardware based problem, which was caused by the light coming from the infrared source for the infrared camera (the 40Watt lamp) combined with the ambient light of a bright day coming through a window. All that excess light was flooding the image projected on the active surface, making the graphics difficult to be read. The software was fortunately running on a laptop computer, which had a working display, which the users could use as a fall back display. If the values on a tangible interface were too light, they could check the values from the laptop display. This helped the process and allowed the evaluation to continue. The subjects used the laptop display to check the details if reading the surface was too difficult. That is not a good example of a tangible user interface experience, as the display is not in the same space as the controls, but that problem is relatively easy to fix in future by changing the light source to one that does not illuminate on the area of spectrum that is visible for humans.

5.2 Observation

Observation was performed on two distinct groups of 2 children. These groups were similar, two boys in each group, and they all were aged six. Comments from the groups are presented in this Chapter in the form of (Group.Person) where groups are A&B and persons in groups are 1&2. The discussion was in finnish and has been translated to english for this thesis.

In the beginning of the evaluation session the children were worried that their programs would vanish from the robot during the evaluation. That was addressed in the research plan, and the prepared RCX unit was changed to the robots that the groups had built. That however proved to be relatively difficult as the robots were not designed to have their RCX units easily changed. The robots had to be partly disassembled and that caused a disturbance on test subjects. The children calmed down when they saw that their robot was complete again. The change of RCX unit was done in the same room in which the TangibleProgrammer was set up. The children were interested in the system and seemed to be eager to begin the evaluation "*Have you done that all by yourself?(B.1)*".

The first task for the groups was to load the software they had previously created by using the LEGO® Robotics Invention System to a computer located near the TangibleProgrammer. That program was to work as a source for the new program. After that

the children got to the programming phase. They were guided with the first blocks. They seemed surprised when they added the first object to the surface, and the surface reacted to that action. Even more exciting seemed to be the connection between code block and modifier block that occurred when they added the time modifier object to the surface for the first time. Figure 31 shows the connection between the time modifier object and a 'go forward' object. "*What?!, How?, What is the idea behind this?!"(A.1)* was the comment when rotating the modifier object changed the value of the other block. Figure 32 shows group B working with the TangibleProgrammer.

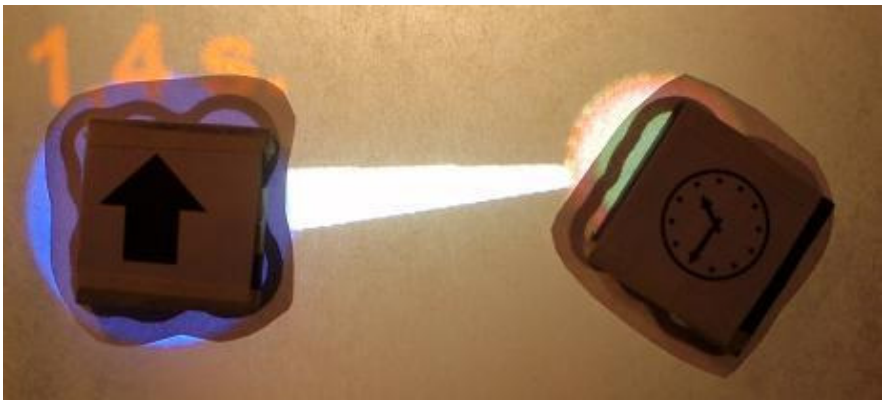

Figure 31: Modifier object connected to another object

Figure 32: Children using the TangibleProgrammer

Both groups were able to distinguish all the icons on the objects and use them in the correct places. The children did not use all the commands available, as the program

they had previously created was a simple program which drove the robot around a maze. Even though teamwork was not a part of this research, the children exhibited some teamwork during this short evaluation. While one child was changing the time on a block another child started to seek for the next block to add to the program.

Evaluation brought out some small problems with TangibleProgrammer prototype. The boundaries in which the modifier object is recognized as being connected to a command block were set too strict. It caused problems when the children wanted to change the values by a large amount. The value is changed by turning the modifier object and large changes require multiple turns on the modifier. The boundaries required the children to keep the modifier object within a two centimeters radius from the point where the connection was made while rotating it to keep the connection. The problem with the reliability of connection was made even worse by the difficulties in reading the surface because of the problem in lighting. The children were not always able to notice when the connection was lost.

When the first group had completed their program, the children were instructed to press the big send button. The way the children used that button brought out a problem that was not noticed during the development. If the button was pressed down for some time it did not send only one request for the compilation and send action, but queued a large number of commands. Since each compilation and upload took about 20 seconds the queue took a while to run and the children got irritated during that: ”*Again?!(A.2)*”.

When the compilation and transfer had ended the children got to test their program. While running their program the children found out that the execution times for run forward block and turn block were not the same as in LEGO® Robotic Invention System. The robot did what it was asked, but turned a little bit less with the program made with the TangibleProgrammer than with the same amount on LEGO® Robotic Invention System. There are many reasons that can cause the difference, but it was probably because the limited processing power of the RCX unit and the underlying Java virtual machine, which may be a little bit slower on executing commands than the original firmware of LEGO® RCX unit. This was probably the reason why the children in the first group said that they thought that the LEGO® Robotics Invention System was better than the TangibleProgrammer. The second group liked TangibleProgrammer better: ”*Cool machine(B.1)*”.

6 Conclusion and future directions

In the beginning of this thesis I introduced the research questions for which this thesis would seek answers for. The questions are reprinted here with summaries of answers.

- **Q1**: Definition: What does the term *Tangible User Interface* mean?

  The term tangible user interface describes a user interface type, which connects the computer and the user through physically tangible objects. The term ”tangible user interface” comes from the research of Hiroshi Ishii and Brygg Ullmer [IU97]. User interface objects in a tangible user interfaces are usually specialized. Specialized object have only one meaning and/or function on the user interface. Specialization and the tangible nature of the user interface is designed to allow the users to grasp the control of the interface more easily.

- **Q2**: Design: What possibilities are there to create a tangible user interface for programming?

  There are multiple technical possibilities which allow creation of a physical interface. The interface technologies can be split into two groups: input and output. Input of information about the physical world can be acquired through electromagnetic sensing, computer vision, electronic connections or wireless communication. This information is however hard to interpret and requires special knowledge, depending on the selected input method, about the field of for example pattern recognition, electrical engineering or some similar subject. There are tangible user interface toolkits which help in the interface creation process by taking care of the information coming from the physical world. By selecting an input technology that has toolkits available the developer can save time in the implementation of the data gathering and use that time for the design of the user interface and the programming of the real processing logic for the system.

  If the technology allows and the objects are built to be "active objects", the output can be arranged through the same objects as used for input. Output in tangible user interfaces is usually graphical. If the selected output technology allows, the computer generated graphics can be displayed in the same physical location

as the user interface objects. This gives more meaning to the physical interface objects as the graphics can give information about the status and possible actions for those objects.

- **Q3**: Implementation: How difficult it is to implement a tangible programming interface? What problems can one expect while developing one?

  Problems that the developer can face depend on the chosen implementation technology. The implementation phase of this thesis concentrated on the usage of visual recognition and the reacTIVision toolkit for tangible user interface creation. The toolkit worked well and the problems faced on the software side were related to the developed code, not to the toolkit.

  The physical part of preparing a tangible user interface is, even with the help of toolkits, hard. Tangible user interfaces always require some building. Physical objects for the user interface must be crafted and technology gathered or built to sense the locations of the objects. In the TangibleProgrammer prototype the image quality was a problem that affected both, the input and the output of the system. The web camera selected for the TangibleProgrammer took images at the resolution of 320 x 240, which required the camera to be very close to the objects to achieve robust object recognition. That limited the available width and height for the space inside which the objects were recognized. The image quality of the projected graphics was also poor because of the excess light coming from the infrared illumination.

- **Q4**: Evaluation: How well did the tangible programming interface manage to fill out the expectations and how should it be modified for continued usage.

  Requirements for the implemented prototype from research question Q3:

  - The user interface must be able to recognize and keep track of the physical objects.
  - By using the interface, the user must be able to create programs that will be compatible with a LEGO® RCX unit based robot.
  - System must be able to move the created software to a LEGO® RCX unit based robot for execution.

  TangibleProgrammer was able to fill the set expectations. ReacTIVision toolkit handled the connection between the physical objects that were used as the user

interface and the TangibleProgrammer. The user interface created LEGO® RCX unit compatible Java code that was transferred to a robot with a press of a button.

As the TangibleProgrammer worked relatively well it is easy to think changes which, when established, would make the TangibleProgrammer usable for even a wider audience.

Difficulties related to physical aspects of the TangibleProgrammer are all probably easy to solve. The main problem that was faced during the evaluation was the instability and limited active area in the recognition of the objects. There were certain places on the active surface where the objects were not recognized well. The camera that was used in the TangibleProgrammer was a cheap web camera. It responds to infrared light with small modifications, but much better results could be achieved through using a bit more expensive camera, which is using a CMOS-type cell. A CMOS-cell gives sharper image and is even more responsive to infrared light [Bla01]. Another option could be a commercial infrared camera that would give even sharper images thus giving more space for the objects.

The camera was not the only part in the TangibleProgrammer that had problems. The light source for the infrared camera flooded the image projected from the video projector, which in turn made the computer generated aspects of the environment difficult to read. The light source could be changed to a real infrared light source that would evenly illuminate the surface with infrared light, without any light on the visible spectrum.

Another possibility that could be used to enhance the precision of the object recognition would be the change of the input technology from visual recognition to electromagnetic. The image based system are still cheaper to develop, since there are no commercially available devices that would allow precise and fast location tracking by using electromagnetic objects. Thus the objects and the recognizing equipment would have to be hand crafted, which would require a lot of work if, in the future there were a need for more than one TangibleProgrammer.

These changes to the equipment would probably also give a larger active area, as the size of objects in use is mostly limited by the quality of the picture that the camera receives. This enhanced space would also give users the possibility to create more complex programs with more blocks. Another option to allow creation of longer programs would be giving users the possibility to create ”macros”, or custom blocks.

As it is easy to reprogram the code “in“ the objects it could be possible to use one type of blocks as a placeholders for larger group of objects as De Guzman *et al.* suggest [DGH03]. This would also help with the amount of usable objects, as the objects used in the first batch of code could be reused in the second. This type of use limits the good aspects of tangible user interfaces. These code blocks are not so easily graspable anymore. And since the users are mainly beginners, the programs they create are not really long. For coding long programs with difficult structures this is not the ideal tool.

The amount of different commands coulda be extended. The TangibleProgrammer did not have any of the basic computational objects such as variables, conditional statements, arithmetical operators or logical operators which would all create useful extension to the available toolset. These changes would extend the possibile user group to older and more experienced programmers.

TangibleProgrammer, while being a prototype, showed in the small conducted evaluation possibilities for future development. Hopefully the price of technology continues to come down so that someday we will see tangible user interfaces outside development facilities. Tangible interaction is still the natural way for humans to deal with physical things.

References

[Asc07] Ascension Technology Corporation. Corporate website. nov. 2007. `http://www.ascension-tech.com/products/flockofbirds.php`, 2007.

[BBMM02] Aggelos Bletsas, Vimal Bhalodia, Marios Mihalakis, and Ilia Mirkin. Network beatles: A distributed wireless network platform for tangible user interfaces. `http://web.media.mit.edu/~aggelos/aggelos_tui2002.pdf`, 2002.

[BK05] Ross. Bencina and Martin Kaltenbrunner. The design and evolution of fiducials for the reactivision system. In *Proceedings of the 3rd International Conference on Generative Systems in the Electronic Arts*, 2005.

[BKJ05] Ross Bencina, Martin Kaltenbrunner, and Sergi Jordá. Improved topological fiducial tracking in the reactivision system. In *2nd IEEE International Workshop on Projector-Camera Systems*, page 99, 2005.

[Bla01] Nicolas Blanc. Ccd vesrsus cmos - has ccd imaging come to an end? In *Photogrammetric Week 01*, 2001.

[BRSB03] Rafael Ballagas, Meredith Ringel, Maureen Stone, and Jan Borchers. istuff: A physical user interface toolkit for ubiquitous computing environments. In *Conference on Human Factors in Computing Systems (CHI 2003)*, pages 537–544. ACM, 2003.

[CMM02] Louis Cochen, Lawrence Manion, and Keith Morrison. *Research Methods in Education*. Routledge Falmer, London, 2002.

[CSR03a] Enrico Costanza, S. B. Shelley, and J Robinson. D-touch: a consumer-grade tangible interface module and musical applications. In *Proceedings of Human-Computer Interaction (HCI03)*, pages 175 – 178, 2003.

[CSR03b] Enrico Costanza, S. B. Shelley, and J Robinson. Introducing audio d-touch: A tangible user interface for music composition and performance. In *Digital Audio Effects (DAFX-03)*, pages 1–5, 2003.

[DGH03] Edward De Guzman and Gary Hsieh. Function composition in physical chaining applications. Technical report, Department of Electrical

Engineering and Computer Science. University of California, Berkeley, 2003.

[DGR04] Edward S. De Guzman and Ana Ramírez. Objectclassifierviews: Support for visual programming in papier-mâché. Technical report, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 2004.

[Dig07] Digidesign. Corporate website. dec. 2007. `http://www.digidesign.com/`, 2007.

[FB97] George W. Fitzmaurice and William Buxton. An empirical evaluation of graspable user interfaces: towards specialized, space-multiplexed input. In *Human Factors in Computing Systems (CHI 1997)*, pages 43–50, 1997.

[FHW+04] Morten Fjeld, Daniel Hobi, Lukas Winterthaler, Benedict Voegtli, and PAtrick Juchli. Teaching electronegativity and dipole movement in tui. In *4th IEEE International Conference on Advanced Learning Technologies (ICALT 04)*, pages 792–794, 2004.

[FIB95] George W. Fitzmaurice, Hiroshi Ishii, and William Buxton. Bricks: Laying the foundations for graspable user interfaces. In *Human Factors in Computing Systems (CHI 1995)*, pages 442–449, 1995.

[Fit96] George W. Fitzmaurice. *Graspable User Interfaces*. PhD thesis, Department of Computer Science, University of Toronto, 1996.

[Fle02] Richard Ribon Fletcher. *Low-Cost Electromagnetic Tagging:Design and Implementation*. PhD thesis, Massachusetts Institute of Technology, 2002. `http://www.media.mit.edu/physics/publications/theses/02.09.fletcher.pdf`.

[GH07] Daniel Guse and Manuel Hollert. Tangible table - userinterface-research and -studies. nov. 2007. `http://www.tangibletable.de/`, 2007.

[GOI98] Matthew G. Gorbet, Maggie Orth, and Hiroshi Ishii. Triangles: Tangible interface for manipulation and exploration of digital information to-

pography. In *Human Factors in Computing Systems (CHI 1998)*, pages 49–56, 1998.

[HDG03] Chen-Je Huang, Ellen Yi-Luen Do, and Mark D Gross. Mousehaus table, a physical interface for urban design. In *16th Annual ACM Symposium on User Interface Software and Technology*, 2003.

[HP99] Kai-Yuh Hsiao and Joseph Paradiso. A new continuous multimodal musical controller using wireless magnetic tags. In *International Computer Music Conference*, 1999.

[Inn07] Innovation First. Vexlabs, corporate website, jul. 2007. `http://www.vexlabs.com/`, 2007.

[Ins07] Instructables. Making a night-vision webcam. website. nov. 2007. `http://www.instructables.com/id/EF7RFPCE2YEP287GZV/?ALLSTEPS`, 2007.

[Int07] Intelitek. Easyc™ programming environment. corporate website. oct. 2007. `http://www.digidesign.com/`, 2007.

[IU97] Hiroshi Ishii and Brygg Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Human Factors in Computing Systems (CHI 1997)*, pages 234–241, 1997.

[JKS02] Ilkka Jormanainen, Osku Kannusmäki, and Erkki Sutinen. Ippe - how to visualize programming with robots. In Moti Ben-Ari, editor, *Second Program Visualization Workshop*, pages 69–73. University of Aarhus, Department of Computer Science, 2002.

[JZKS07] Ilkka Jormanainen, Yuejun Zhang, Kinshuk, and Erkki Sutinen. Pedagogical Agents for Teacher Intervention in Educational Robotics Classes: Implementation Issues. In *the First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL 2007)*, pages 49–56, Los Alamitos, CA, March 2007. IEEE Computer Society.

[KBBC05] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza. Tuio: A protocol for table-top tangible user interfaces. In

6th International Workshop on Gesture in Human-Computer Interaction and Simulation, Vannes, 2005.

[KBNR03] Boriana Koleva, Steve Benford, Kher Hui Ng, and Tom Rodden. A framework for tangible user interfaces. In *Physical Interaction (PI 2003)*, 2003. Ei sivuja.

[KBP+00] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana. Virtual object manipulation on a table-top ar environment. In *International Symposium on Augmented Reality (ISAR 2000)*, pages 111–119, 2000.

[KIMK00] Yoshifumi Kitamura, Yuichi Itoh, Toshihiro Masaki, and Fumio Kishino. Activecube: A bi-directional user interface using cubes. In *Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*, pages 99–102, 2000.

[Kle03] Scott R Klemmer. Papier-mâché: Toolkit support for tangible interaction. In *Symposium on User Interface Software and Technology (UIST 2003)*, 2003. Ei sivuja.

[KLLL04] Scott R. Klemmer, Jack Li, James Lin, and James A. Landay. Papier-mâché: Toolkit support for tangible input. In *Human Factors in Computing Systems (CHI 2004)*, pages 399–406, 2004.

[KLPBSV07] Eija Kärnä-Lin, Kaisa Pihlainen-Bednarik, Erkki Sutinen, and Marjo Virnes. Can Robots Teach? Preliminary Results on Educational Robotics in Special Education, July 2007.

[KOC04] Martin Kaltenbrunner, Sile O'Modhrain, and Enrico Costanza. Object design considerations for tangible musical interfaces. In *ConGAS Symposium on Gesture Interfaces for Multimedia Systems (COST287)*, 2004.

[LEG07a] LEGO. Lego robolab. website. nov. 2007. `http://www.lego.com/eng/education/mindstorms/home.asp?pagename=robolab`, 2007.

[LEG07b] LEGO. Lego robotics invention system 2.0. website. nov. 2007. http://mindstorms.lego.com/eng/products/ris/index.asp, 2007.

[leJ07] leJOS. Java for lego mindstorms. website. nov. 2007. http://lejos.sourceforge.net/, 2007.

[Log00] Logo Foundation. Website. nov. 2007. http://el.media.mit.edu/Logo-foundation/logo/index.html, 2000.

[Lun07a] Lunar and Planetary Institution. Life at the limits: Earth, mars and beyond - an educators workshop and fieldtrip. website. nov. 2007. http://www.lpi.usra.edu/education/fieldtrips/2005/activities/ir_spectrum/ir_webcam.html, 2007.

[Lun07b] Henrik Hautop Lund. Adaptronics group. website. dec. 2007. http://www.adaptronics.dk/, 2007.

[MCK06] Leonel Morgado, Maria Cruz, and Ken Kahn. Radia perlman - a pioneer of young children computer programming. In J.A. Mesa González Méndez-Vilas, A. Solano Martín and J. Mesa González, editors, *Current Developments in Technology-Assisted Education*, pages 1903–1908. FORMATEX, Badajoz, Spain, 2006.

[McN00] Timothy Scott McNerney. Tangible programming bricks: An approach to making programming accessible to everyone. Master's thesis, Massachusetts Institute of Technology, 2000. http://xenia.media.mit.edu/~mcnerney/mcnerney-sm-thesis.pdf.

[McN04] Timothy Scott McNerney. From turtles to tangible programming bricks: explorations in physical language design. *Personal and Ubiquitous Computing*, 5:326–337, 2004.

[Mer07] Merriam-Webster. Merriam-webster dictionary. website. nov. 2007. http://www.m-w.com/dictionary/tangible, 2007.

[Nie02] Jacob Nielsen. Intelligent bricks. Master's thesis, University of Southern Denmark, Odense, 2002.

[NQC07] NQC. Not quite c programming language. website. nov. 2007. http://bricxcc.sourceforge.net/nqc/, 2007.

[OF04] Claire O'Malley and Danae Stanton Fraser. *REPORT 12: Literature Review in Learning with Tangible Technologies*. Futurelab, Bristol, United Kingdom, 2004.

[Pap80] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York, 1980.

[Per76] Radia Perlman. Using computer technology to provide a creative learning environment for preschool children. Technical report, MIT Logo Memo #24, 1976.

[PIHP01] James Patten, Hiroshi Ishii, Jim Hines, and Gian Pangaro. Sensetable: a wireless object tracking platform for tangible user interfaces. In *Human Factors in Computing Systems (CHI 2001)*, pages 253–260, 2001.

[PMAI02] Gian Pangaro, Dan Maynes-Aminzade, and Hiroshi Ishii. The actuated workbench: Computer-controlled actuation in tabletop tangible interfaces. In *The Proceedings of Symposium on User Interface Software and Technology (UIST 2002)*, volume 2003, pages 699–699, 2002.

[PPGT07] Udai Singh Pawar, Joyojeet Pal, Rahul Gupta, and Kentaro Toyama. Multiple mice for retention tasks in disadvantaged schools. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1581–1590, 2007.

[PRI02] James Patten, Ben Recht, and Hiroshi Ishii. Audiopad: A tag-based interface for musical performance. In *New Interface for Musical Expression (NIME '02)*, pages 1–6, 2002.

[Raf04] Hayes Solos Raffle. Topobo: A 3-d constructive assembly system with kinetic memory. Master's thesis, Massachusetts Institute of Technology, 2004. `http://web.media.mit.edu/~hayes/topobo/Raffle_MS_Thesis_small.pdf`.

[Rea07] ReacTable. Website. nov. 2007. `http://www.iua.upf.es/mtg/reacTable`, 2007.

[RUO01] Jun Rekimoto, Brygg Ullmer, and Haruo Oba. Datatiles: A modular platform for mixed physical and graphical interactions. In *Human Factors in Computing Systems (CHI 2001)*, pages 269–276, 2001.

[RZSP04] Dan Rosenfeld, Michael Zawadzki, Jeremi Sudol, and Ken Perlin. Physical objects as bidirectional user interface elements. *Computer Graphics and Applications, IEEE*, 24(1):44–49, January/February 2004.

[Sch04] Bert Schiettecatte. Interaction design for electronic musical interfaces. In *Human Factors in Computing Systems*, page Poster, 2004.

[SE04] Elizabeth Sklar and Amy Eguchi. Learning while teaching robotics. In *AAAI Spring Symposium 2004 on Accessible Hands-on Artificial Intelligence and Robotics Education*, 2004.

[SK93] H. Suzuki and H Kato. Algoblock: a tangible programming language, a tool for collaborative learning. In *The Proceedings of 4th European Logo Conference*, pages 297–303, 1993.

[Smi06] A.C. Smith. Tangible cubes as programming objects. In *16th International Conference on Artificial Reality and Telexistence–Workshops, 2006. (ICAT '06)*, pages 157–161, 2006.

[Smi07] Andrew C Smith. Using magnets in physical blocks that behave as programming objects. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 147 – 150. ACM. New York, NY, USA, 2007.

[SV00] Dag Svanæs and William Verplank. In search of metaphors for tangible user intefaces. In *Designing augmented reality environments (DARE 2000)*, pages 121–129, 2000.

[Tec07] Technologies for Children with Individual Needs. Technologies for children with individual needs. website. aug. 2007. `http://cs.joensuu.fi/etp/cms/`, 2007.

[Tek07] Teknologiakasvatuksen kehittämisprojekti. Project website (finnish). nov. 2007. `http://cs.joensuu.fi/tkp/`, 2007.

[UI97] Brygg Ullmer and Hiroshi Ishii. The metadesk: Models and prototypes for tangible user interfaces. In *Symposium on User Interface Software and Technology (UIST 1997)*, pages 223–232, 1997.

[UI01] Brygg Ullmer and Hiroshi Ishii. *Emerging Frameworks for Tangible User Interfaces*, chapter Human-Computer Interaction in the New Millenium, pages 579–601. Addison-Wesley, 2001.

[Ull02] Brygg Anders Ullmer. *Tangible Interfaces for Manipulating Aggregates of Digital Information*. PhD thesis, Program in Media Arts and Sciences, Massachusetts Institute of Technology, 2002.

[UoJS07] Department of Computer Science University of Joensuu and Statistics. Organization website. nov. 2007. `http://www.joensuu.fi/tkt/english/`, 2007.

[Val03] Andrea Valente. C-cards: using paper and scissors to understand computer science. In *The Proceedings of the Kolin Kolistelut-Koli Calling, 2003*, pages 84–92, 2003.

[Wac07] Wacom. Corporate website. nov. 2007. `http://www.wacom.com/`, 2007.

[WC00] Tim Wright and Andy Cockburn. Writing, reading, watching: A task-based analysis and review of learners' programming environments, 2000.

[Wel93] Pierre Wellner. Interacting with paper on the digitaldesk. *Communications of the ACM*, 36(7):87–96, July 1993.

[Xu05] Diana Xu. Tangible user interface for children – an overview. Technical report, UCLAN Department of Computing Conference, 2005.

[ZAAM+07] Imran A Zualkernan, Abdul-Rahman Al-Ali, Hassan A. M. Muhsen, Mohammadhossein Afrasiabi, and Serop; Babikian. A wireless sensor-based toolkit for building tangible learning systems. In *International Conference on Advanced Learning Technologies 2007*, pages 152 – 156, 2007.