A virtual sandtray on a tabletop computer

Thomas ten Cate

July 9, 2009

Master's thesis
Computing Science
University of Groningen

Supervisors:
Dr. Tobias Isenberg (University of Groningen)
Dr. Sheelagh Carpendale (University of Calgary)

Abstract

We explore the applications of shallow-depth three-dimensional interaction on a touch-sensitive tabletop display. To guide this exploration, we chose the particular application of a virtual, simulated sandtray, such as those used in certain kinds of play therapy. This sandtray program is developed and evaluated in cooperation with several experts in sandtray therapy.

The usefulness of a virtual sandtray for play therapy is discussed, and some possibilities of the virtual world are explored to see what advantages a digital sandtray can have over its physical counterpart.

Interaction with 3D objects on a tabletop is extended from previous work, allowing for full control over all six degrees of freedom. For further manipulations, the concept of virtual tools is introduced. Such tools avoid modality in the interface, thereby allowing for their use in a collaborative, multi-user setting.

Contents

Acknowledgements

First and foremost, I would like to thank my supervisors, Sheelagh Carpendale at the Interactions Lab of the University of Calgary, where this work was done, and Tobias Isenberg at my home university of Groningen. Without their excellent guidance and advice, this thesis would not have been possible.

I also want to thank Mark Hancock, whose software framework I used, and with whom I worked together in a fruitful and very pleasant collaboration. The little tidbits of wisdom I picked up here are too many to count.

Furthermore, I am grateful to everyone else in the Interactions Lab. Many iLabbers contributed valuable suggestions to my prototype, and in general provided a very stimulating environment in which I immediately felt at home. “A great place to work and learn” indeed.

But not only computer scientists made a contribution to this work. I am indebted to Steve ‘Toumbi’ Heynen, who provided much information about sandtray therapy, and was very much with me and often even ahead of me when thinking about its digital implementation. I am also very grateful to Monica Carpendale and Nicole LeBihan, who travelled a long way to come and see the prototype, and provided a huge amount of extremely valuable feedback during our exhausting but very rewarding evaluation day.

I would also like to thank Annemieke Beereboom, the international exchange coordinator at the University of Groningen, without whom I would never have made it to Canada in the first place.

Last but not least, many thanks to my friends and family at home, especially my parents, for putting up with my absence. I missed you all.

This work was largely done using free open source software. Many thanks to the countless developers who willingly sacrificed their spare time to bring us TeX Live, Eclipse, TeXlipse, Sumatra PDF, Inkscape, Paint.NET, GIMP and Blender.

No rubber ducks were harmed in the making of this thesis.

Chapter 1

Introduction

Much work in the human-computer interaction field nowadays investigates the interaction with large displays and touch screens, and tabletop computers in particular. However, the vast majority of this research only uses images that are as two-dimensional as the display itself, whereas little is said about interaction with a virtual three-dimensional world. Yet humans are familiar from birth with three-dimensionality, making interaction with the virtual 3D realm a worthwhile object of study.

As a case study into 3D interaction on tabletop computers, we construct a virtual sandtray, inspired by the (physical) sandtrays used in sandtray therapy. In this form of therapy, the patient is allowed to play freely with a range of small figures or toys in a box, or tray, filled with sand. The form factor and affordances of a physical sandtray have many similarities with those of a tabletop computer, making the virtual sandtray a suitable stepping stone into the field of 3D on tabletops.

Moreover, there are many potential advantages to the use of a virtual sandtray instead of a physical one. The absence of physical limitations means we can provide therapy patients with a greater range of options to express themselves. A digital implementation might also appeal more to patients of a certain age group.

The rest of this chapter is organized as follows. The project description ('what') is given in Section 1.1, its motivation ('why') in Section 1.2 and its method ('how') in Section 1.3. The project is scoped in Section 1.4, and its objectives are presented in Section 1.5. Finally, Section 1.6 will introduce the structure of the rest of this thesis.

1.1 Project description

In this project, we construct a computer program that simulates a sandtray on a tabletop computer. Of course, representing the full richness of real-world interaction is not feasible; instead, we aim for a program that is similar in concept and purpose to a physical sandtray, while reducing the possibilities in some areas and extending them in others.

Figure 1.1 shows the prototype application resulting from this project; the 'demo' video referred to in appendix A demonstrates the prototype in action.

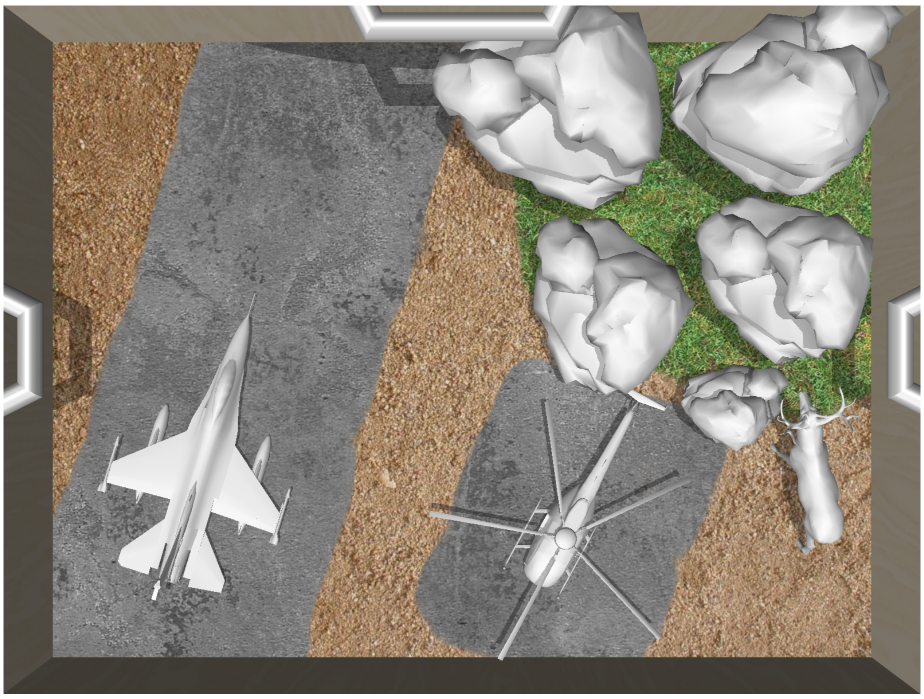


Figure 1.1: A screenshot of the sandtray application constructed in this project. It shows a typical scene that could be constructed during an actual therapy session, with an airfield, two aircraft, a forest and a deer.

The figure and video serve only to show the general idea; the details will be discussed in depth throughout this thesis. It is good to keep in mind that this application is merely a prototype, a proof-of-concept, rather than a polished, finished application that is ready for use in actual therapy sessions.

1.2 Motivation

The reasons for undertaking this particular project fall into two categories. On the one hand, there are questions from the field of human-computer interaction relating to 3D interaction, tabletops and direct touch, that can be explored using the virtual sandtray. On the other hand, there are various reasons why a digitally supported form of sandtray therapy could be advantageous over the current, purely physical setup.

Tabletop computers are used mainly for collaboration. As such, researchers try to find ways to perform real-world collaboration tasks on tabletops, such as passing documents, making sketches and taking notes. Because the tabletop is a two-dimensional surface, it most naturally affords interaction with two-dimensional virtual objects, which is what most tabletop research focuses on. However, in real-world collaboration, people have the ability to easily flip, stack, sort and store artifacts using the third dimension. Such abilities could be offered on tabletop computers by adding a third dimension to the virtual world as well. Since the touch input to a tabletop is two-dimensional in nature, this raises the problem of how interaction with the third dimension should take place. The virtual sandtray serves as a case study to investigate this problem.

A sandtray is a suitable case to study for several reasons. First, like a tabletop, it is a horizontal surface, with all the affordances that come with it. Second, unlike for example the 3D world of first-person games, a sandtray stays in the same place relative to the viewer, eliminating the need for viewpoint changing. Third, sandtrays in therapy have a suitable size to be represented on tabletop computers at 1:1 scale. All these factors make the sandtray a relatively good match for the affordances of a tabletop computer.

From a therapeutic point of view, there are also several reasons to develop a virtual sandtray. One advantage of a tabletop computer over a tray of sand with toys is that it may appeal more to patients of a certain age group. Especially teenagers may find a sandtray childish, but might be more motivated to play with a modern, "cool" piece of technology. A virtual sandtray can also be more suitable for patients with certain forms of autism, who dislike the feel of sand and the disorder of a sandtray.

Another therapeutic advantage is that a virtual sandtray can offer functionality that is simply not possible in a physical one, because the latter is constrained by the laws of physics. In the digital world we, the designers, have nearly unlimited freedom. We can leverage this freedom to provide the patient with more ways to express themselves in the sandtray, thereby giving the therapist more insight into their psyche.

1.3 Method

During the investigation, a prototype of a virtual sandtray program was developed for a tabletop computer. This was done using readily available tabletop hardware. The prototype is based on the Java framework that was developed by Hancock for his work on 3D display and interaction on tabletop computers [Han07a; Han07b].

The design and evaluation of this program was done in cooperation with three sandtray therapists. During initial development, we have been in contact with them via e-mail to discuss the aspects of the project relevant to therapy. Decisions about which features to include in the prototype were largely based on this correspondence. When the prototype was in a presentable state, the therapists were physically present to evaluate the application first-hand and provide feedback, by which further research can be guided.

1.4 Scope

The topic of tabletop interaction is vast, and so are the potential possibilities of a virtual sandtray. It is therefore necessary to clearly define the scope of this project in the various areas that it touches upon.

Our main field of research is human-computer interaction, and in particular tabletop interaction. Interaction techniques using direct touch obviously play an important role throughout this work. Although we do design for multi-user interaction, research into the nature of collaboration is not an important part of this work.

The virtual sandtray is, essentially, a computer simulation of a physical system, and therefore uses results from the field of computational physics. However, we will not attempt to create a perfect simulation of the physical sandtray in the digital world. In particular, the use of simulated sand in our implementation would be a large project in itself, and we will not attempt it. Apart from resizing, we will not perform any deformations on the objects in the sandtray, but restrict ourselves to the simulation of rigid bodies only.

The topic of modality in interfaces is too large and varied to attempt to provide a solution to all problems. Therefore, the notion of virtual tools must be seen as exploratory, and reveals only the tip of the iceberg.

In terms of hardware, we use only the possibilities offered by a tabletop computer without any extra devices; no haptic devices, physical proxies or other possible input devices will be used. We do, however, assume that a large number of touch points can be detected simultaneously. The use of sound effects might be a feasible addition, but will not be explored; the only output will thus be in the form of a two-dimensional image on a tabletop screen.

Some well-established techniques from the field of three-dimensional computer graphics are used in the implementation. We also use some results from computational geometry.

1.5 Objectives

The aim of this project is threefold. Firstly, it explores what the advantages and drawbacks of a virtual sandtray are in comparison to a physical one. Secondly, the project develops some new and augmented techniques in the more general field of 3D interaction on touch-sensitive displays, and raises questions to be answered by future work. Thirdly, it explores the concept of ‘virtual tools’, which could prove to be a useful paradigm on touch screens, and on tabletops in particular.

1.5.1 Possibilities of a virtual sandtray

A virtual sandtray offers many options that a physical one does not, because it is not constrained by the laws of physics. Many of these potential features are considered, but not all of them will be explored in this project; instead, only a subset will be chosen for implementation and investigation. Features to be implemented will be selected based on their feasibility, their usefulness for therapy, and their relevance to interactions research.

1.5.2 3D interaction on touch displays

The interaction techniques referred to in the previous section will, for the most part, not be specific to virtual sandtrays only. Rather, the results found in this investigation will allow themselves to be used for interaction with 3D objects on touch displays in general, for example, other creative applications, or virtual desktops. Hence, the second purpose of this thesis is to provide more general results in the field of 3D interaction on touch-sensitive displays.

1.5.3 Virtual tools

It is generally recognised in the human-computer interaction community that modality is a bad thing for interfaces. On tabletops, where multiple people can interact simultaneously, modality presents an even larger problem. In this thesis, we introduce the notion of 'virtual tools', which can be used to avoid system-maintained modes and thereby help to avoid mode errors and reduce cognitive load.

1.6 Organization

The rest of this thesis is organized as follows. First, in Chapter 2, we investigate related literature on tabletop and 3D interaction, and on storytelling software. For readers unfamiliar with sandtray therapy, Chapter 3 gives the necessary background information. In Chapter 4, we then construct the design of our virtual sandtray based on this information. Interaction with figurines is a sufficiently large topic to warrant a chapter by itself, and is presented in Chapter 5. The concept of virtual tools, which act on the sandtray and its contents, is introduced and explored in Chapter 6. Chapter 7 gives an overview of the implementation of the program, without getting into too much detail. The results of an evaluation session with sandtray therapists are presented in Chapter 8, and finally, Chapter 9 presents our conclusions and indicates directions for further research.

Each of the following chapters starts with an introduction of its contents and structure. The main matter of the chapter follows, and then each is rounded off with a short summary.

Chapter 2

Related work

This chapter embeds our work in the existing literature, giving references to related publications and indicating the relevance of each publication to our work. This work is based on previous work in several different fields within computer science, and interactions research in particular. The rest of this chapter is divided into sections according to the primary field of study of the publications.

Section 2.1 describes some general work related to tabletops. In Section 2.2 we discuss some applications of tabletop computers that are in some way related to our work. In Section 2.3 we give an overview of the current state of direct touch hardware. The most directly relevant literature, however, is that on interaction methods on tabletop computers, as discussed in Section 2.4. Finally, other forms of storytelling software and hardware are discussed in Section 2.5.

2.1 Tabletops and interaction

Grossman and Wigdor provide a taxonomy [Gro07] of 3D display and interaction on tabletops (and several other device categories), giving such systems a place along various dimensions. In their terminology, the *perceived* and *actual display spaces* of our 3D sandtray are both *2D table constrained*, since the projection space is constrained to the 2D plane of the tabletop, and the viewer perceives it as a 2D surface even though 3D objects are projected onto it. In our current work, no *viewpoint correlation* is taking place, although an interesting direction to explore would be the use of head tracking to provide motion parallax. The tabletop is touch-sensitive, hence, we use a *direct 2D input space*. The physical form of the display is, of course, a *table*, of either *personal* or *collaborative* size. Interesting future work would be to allow the use of physical *proxy* objects on the table. Grossman et al. mention the problem of mapping 2D input to 3D space, a problem that we will address in Section 5.3.

Terrenghi et al. [Ter07] performed an exploratory study to investigate the fundamental differences between interaction with the actual physical world and interaction with the digital simulation thereof on a tabletop surface. They conclude that designers should not strive for the most faithful representation of the real world, but rather think about what properties of the real world help people accomplish a certain task, and then try to use possibly different styles of interaction to accomplish that same task in the digital world. We use this approach when designing our sandtray by not slavishly trying to emulate real-world behaviour, but using only those real-world concepts that are useful in the virtual realm as well.

A study by Jacob et al. [Jac94] suggests that the input device that is employed in a particular application should match the task at hand and the interaction method used to accomplish that task. This suggests that the two-dimensional input to a tabletop screen is a poor input device to manipulate a three-dimensional scene. However, true three-dimensional input devices are not yet widely available, so several methods have been developed to manipulate 3D scenes using 2D touch input; see further Section 2.4.2.

Ryall et al. [Rya04] explore the effect of table size on collaboration. Although they focus on the use of tabletops in a group setting, some factors, such as physical reach and visibility, are equally applicable to a single-user application. They find no significant effect of table size on the completion time of an assembly task, but this may be due to the fact that the two tables used in the study do not differ much in size (80 cm and 107 cm diagonals). A tentative conclusion we can draw is that table size is not a very significant factor in our work.

2.2 Applications of tabletop computers

Most work related to tabletop computers focuses on collaboration in a multi-user setting. Yet, much of this work can directly be applied to the simpler, single-user case. We mention here some of the applications of tabletops that are closely related to our work.

Streitz et al. [Str99] use a tabletop computer as a part of their i-LAND 'roomware' project. The i-LAND project focuses on the use of several different roomware components, such as wall displays and chairs with built-in tablet computers, to foster creativity. A tabletop computer is used for collaboration in concert with the other components. The 'creativity' aspect is of much interest to us, but unfortunately the paper says little about this aspect in general.

Several applications of tabletops in the medical world were investigated by Piper. Piper and Hollan [Pip08] experimented with a tabletop computer to facilitate the conversation between a deaf person and their physician. This conversation uses text entry through both keyboards and speech recognition. A direct-touch interface can be used to move and organize speech bubbles, and is also used to display medical images relevant to the conversation. Although this work focuses more on collaboration, and the form of the communication is very different, the setup is very reminiscent of ours: a doctor and a patient using a tabletop computer to communicate.

Piper et al. [Pip06] also developed a cooperative game on a tabletop computer to help autistic children develop their social skills. The objective of the game is to construct a path from tiles, in such a way to maximize the score. Enforcing of the rules was done either by a human moderator or by the software itself. Although mostly focused on the group cooperation aspect, this work does demonstrate that tabletop computers can be a valuable tool when working with children in general and in a therapeutic setting in particular.

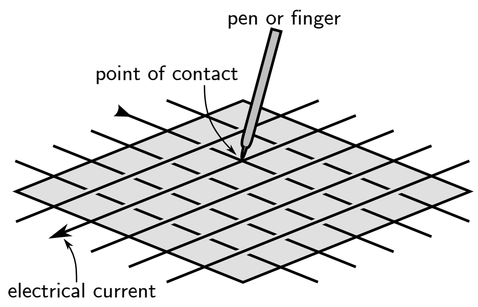


Figure 2.1: The working of a resistive touch screen. Pressure exerted by a pen, finger or other object presses two strips of conducive material together, allowing a current to run.

2.3 Direct touch technologies

Several different technologies exist that provide touch sensitivity on tabletop screens and displays in general. To give the reader an overview of the options without wasting too many words, we will not discuss all of the possibilities, instead mentioning only the most common and useful methods.

One of the most commonly used methods is the resistive touch screen, used in devices like drawing tablets and the Nintendo® DS. Two layers inside the screen are separated by a narrow space, one layer with horizontal strips of conducive material, one with vertical strips. Pressure on the screen closes the gap and allows an electric current to run. By measuring the resistance between each pair of strips, the electronics can determine which intersection was closed; see Figure 2.1. Another possibility is to use resistive films instead of strips and applying an alternatingly horizontal and vertical voltage gradient. In this case, the location of the contact point can be determined by measuring the amount of current that flows through the layer. A major limitation of this technique is that it is single-touch only; when multiple touch points are present, there is no way to detect each one individually.

Capacitive technology, such as that used in the Apple® iPhone™ and iPod® Touch, does allow for multi-touch. It uses a thin, transparent coating of conducive material as a capacitor, which is electrically charged. When the layer is touched, the drop in charge and thereby voltage is measured in each corner. From this data, the touch location can be computed. By using a grid instead of a uniform film, multi-touch can be achieved. Since it uses the natural capacitance of the human body, such a touch screen can only be used with bare skin.

MERL's DiamondTouch [Die01] uses a similar technique called capacitive coupling: people sit or stand on pads through which they become electrically charged, and antennas in the table surface detect the proximity of this charge. By varying the charge between different people, the system can detect who is touching where. Because the antennas in the table surface are arranged in rows and columns, theoretically only one touch per person can be detected, similar to a resistive touch screen. However, using timing information it is possible to provide multi-touch.

Various methods using infrared cameras above the table surface have been

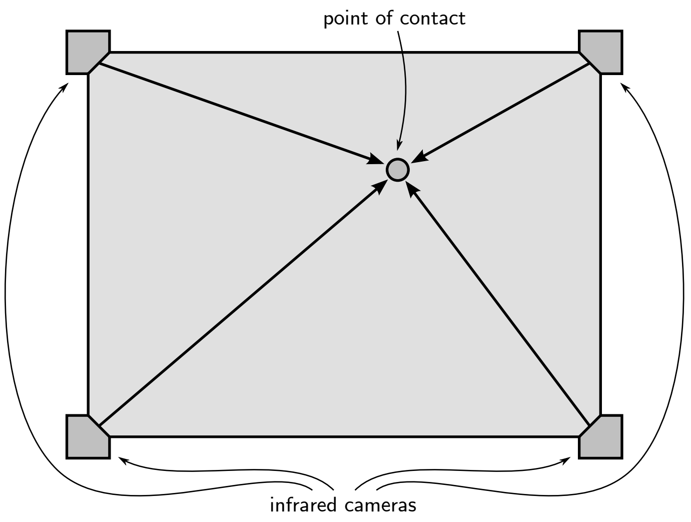


Figure 2.2: The working of a DViT touch sensor [SMA03]. From the image of a finger registered by multiple cameras, the position of the finger can be reconstructed.

developed, such as in the DViT technology [SMA03] shown in Figure 2.2. A fingertip or other object close to the surface is registered by a number of cameras, from which the position can be triangulated. The number of simultaneous touches that can be detected depends on the number of cameras and their positioning; the four cameras in the corners of the SMART Board, which uses DViT, can distinguish at most two touches at a time. Part of our work was performed using such a SMART Board [SMAa].

In the Microsoft Surface [Mic], the table is flooded from below with infrared light. Cameras below the surface register the light reflected from touches and objects close to the surface. Computer vision methods can be used to convert the camera images into discrete touch points, along with size and shape information. The number of touches that can be detected is only limited by processing power. It is also possible to detect objects even if they are not in direct contact with the surface, but it can be difficult to distinguish these from objects that are actually in contact.

A similar technique is ‘frustrated total internal reflection’, better known as FTIR [Han05]. An acrylic layer inside the screen is flooded with infrared light, which is normally totally reflected inside the layer. When the layer surface is touched, the total internal reflection is broken and the light will diffusely leave the layer. Again, a camera beneath the table is used to register the escaped light, as illustrated in Figure 2.3. The number of simultaneous touches that can be detected is only limited by processing power, and is practically unlimited even with commodity hardware. This method cannot detect objects that are not in contact with the surface. On the other hand, it is possible to deduce an indication of pressure from the amount of infrared light that reaches the camera. FTIR is the technique used by the SMART Table [SMAb] on which most of our research was conducted.

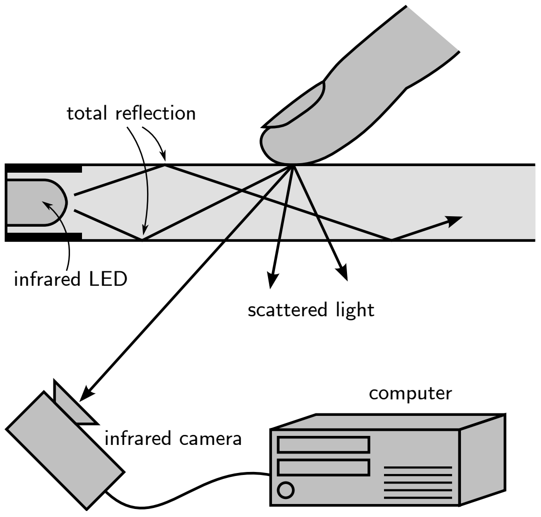


Figure 2.3: The working of FTIR touch detection [Han05].

2.4 Direct touch interaction methods

The most straightforward way to use touch input is to take traditional point-and-click mouse-based interfaces and replace the mouse input by the touch of a fingertip. However, representing the 'hover' action (moving the mouse over an object without clicking it) can generally not be replicated in a direct touch setting. In general, traditional desktop interaction is a poor fit for tabletops and multi-touch screens in general. Much research therefore focuses on novel interaction methods to replace the traditional mouse input by interaction techniques more suited to direct touch input. Other works explore the possibilities of the richer input provided by a touch sensor, consisting of multiple simultaneous touches, shape information or pressure information.

Of those methods that focus on tabletops, most are intended for use by multiple people at the same time. Although, in our project, a single person will be interacting most of the time, we do not assume single-user interaction only, so many of the methods used in multi-user interaction are still applicable. Most interaction techniques focus on two-dimensional interaction, that is, the virtual space and objects interacted with seem two-dimensional; but techniques for interacting with three-dimensional space and objects also exist.

2.4.1 2D interaction methods

Since a screen surface naturally suggests a two-dimensional space, much research focuses on novel methods to interact with two-dimensional objects.

One of the most natural and useful things to do on tabletops, both physical and digital ones, is to move things around, possibly rotating them so they face a particular direction. A study by Forlines et al. [For05] investigated three different methods to automatically orient documents while they are moved. One of their more interesting findings is that the fastest and most precise method is not always the method that is subjectively preferred by the test subjects.

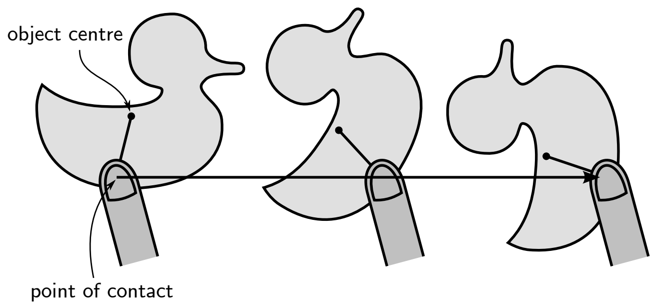


Figure 2.4: Integrated rotation and translation technique using a single finger, as described by Kruger et al. [Kru05]. The method acts as if an opposing force was applied to the object centre. As the finger is dragged to the right, the object's centre will lag behind the touch point.

Kruger et al. [Kru05] describe a method for integrated rotation and translation of two-dimensional objects using only a single point of contact; see Figure 2.4. To constrain the motion to translation only, a special region on the object can be used.

The work of Liu et al. [Liu06] has similar goals, but uses a specialized input device to measure the rotation of the hand, thereby achieving a very direct match between input and output space.

Hancock et al. [Han06] surveyed five different interaction methods to support rotation and translation of two-dimensional objects. They conclude that no single technique is superior, and that the choice of interaction method should depend on the specific task to be performed. All these studies in 2D form the foundations of the 3D work that we build upon.

In many situations, interaction beyond the moving and rotating of objects can be desirable. Other behaviours can be triggered by traditional buttons and menus, which are well understood. However, on a tabletop computer it is also possible to use gestures. To name just one of many examples, Wu and Balakrishnan [Wu03] use various gestures and hand shape postures to manipulate a furniture layout in their RoomPlanner application. They also employ tool palettes and context-sensitive menus. RoomPlanner is similar to our sandtray in the sense that it allows for the construction of a 3D virtual environment, although RoomPlanner disregards the 3D aspect and focuses on the floor plan exclusively.

Some two-dimensional applications use simulated physics as a means of interaction. Reetz et al. [Ree06] describe a flicking method similar to physical tossing, that can be used to move objects across a large distance to a location that would normally be out of reach. Flicking is a feature that comes 'for free' in our project since we use a full physics simulation engine.

Cao et al. [Cao08] in their ShapeTouch application use the shape of contact point to represent, among other things, the amount of force applied to two-dimensional objects. Although we do not attempt this in our project, it is an interesting path for future research. They also use various gestures inspired by the physical world to trigger actions such as peeling and sliding.

Davidson and Han [Dav08] use pressure information to slightly tilt two-dimensional objects out of the plane, so that they can be moved over and underneath each other. Depth cues are added to indicate the tilt of objects to the viewer, thereby moving this application in the direction of the 3D realm. In fact, this can be seen as a forerunner of the concept of 'shallow-depth' 3D, which we will define shortly.

2.4.2 3D interaction methods

Although tabletops are most naturally suited to interaction with a 2D virtual world, 3D applications are also possible. The mapping from the 2D touch input to actions in the 3D virtual world is a problem to which many solutions have been proposed, but it remains a challenging issue.

3D interaction with 3D input

To achieve 3D input to a tabletop computer, extra hardware is required. This section discusses some publications that use such hardware for various purposes. Although we do not use any such hardware, some of the findings are still of interest to us.

Balakrishnan and Kurtenbach [Bal99] investigate the possibility of bimanual navigation in a 3D world. They conclude that such two-handed control can be superior to one-handed control, if a suitable interaction technique is chosen. This finding is of much interest to us, since two-handed control is a realistic possibility on a multi-touch tabletop computer.

Fröhlich et al. [Frö00] use three-dimensional physics simulation for assembly tasks on the Responsive Workbench. Their method works by connecting one's hands to the object via virtual springs. Although, as Wilson et al. [Wil08] discuss, springs do not work well when the contact points are moved apart, the concept of physics simulation remains viable and is heavily used in our project.

The SandScape project by Ishii et al. [Ish04; Wan02] uses physical sand (actually consisting of glass beads) onto which an image is top-projected. This allows one to deform the table surface directly. The thickness of the layer of sand is measured by observing the amount of infrared light that passes through it. A partly physical, partly digital approach such as this might be more inviting than a purely digital approach, but is unfortunately outside the scope of this project.

3D interaction with 2D input

Hancock et al. [Han07b] extend the 2D method from previous work by Kruger et al. [Kru05] to work with 3D objects in 'shallow-depth' 3D. This means that, although the objects manipulated are three-dimensional, their depth coordinate cannot be changed using the described interaction method. Furthermore, two other methods are described that use two and three touches, respectively, to provide more control. The three-touch method was found to be faster, and it was also the method preferred by test subjects. Since our work is based on this technique, a more detailed description is provided in Section 5.3.

As with two dimensions, three-dimensional interaction methods can also be based on physics simulations. Using the two-dimensional input of a pen on a tablet, Agarawala and Balakrishnan [Aga06] employ 3D graphics and physics simulation in their BumpTop application to bring the desktop metaphor on

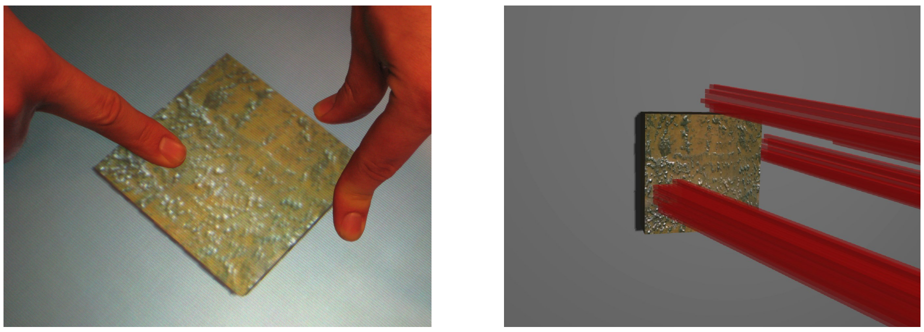


Figure 2.5: The physics-based interaction technique by Wilson et al. [Wil08]. The contours of the contacts are extended into the virtual world as bundles of stick-shaped particles.

desktop computers closer to the real world. BumpTop supports tossing and piling of icons. The interaction method is mainly based on gestures, but also employs pie-shaped menus.

Wilson et al. [Wil08] use a physics simulation in which the outline of the fingers is ‘extended’ by narrow cylinders into the virtual 3D world (Figure 2.5). This allows for surprisingly rich interactions, using not only fingertips but also the side of the hand, or other physical objects. However, moving objects vertically is quite challenging at best. Stacking is demonstrated only with special pillow-shaped objects, which will slide on top of each other when pressed together. Another issue with this method is the lack of precision, which could be a problem in applications that require precise control. Yet, the technique is not without merits, and although we did not implement it, we indicate how future work could incorporate this technique.

2.5 Technologies supporting storytelling

The purpose of our virtual sandtray, like that of its physical counterpart, is to enable and encourage the telling of a story. Several computer-based applications have been developed with a similar purpose.

Much work concerning digitally supported storytelling by children is due to Cassell. Most of her work focuses on collaborative storytelling and the use of artificial partners therein, but some is more directly related to our sandtray. The StoryMat developed by Cassell and Ryokai [Cas99; Cas01] is a physical play mat with physical toys which are tracked by a computer system. This system bears striking resemblances to our sandtray application, but has a much more tangible interface.

Earlier work by Bers et al. [Ber98] uses a system called SAGE to help young cardiac patients cope with their situation. The system employs a robotic stuffed animal to encourage children’s exploration of their inner worlds through storytelling. Like sandtray therapy, this is also a form of therapy-through-storytelling.

Zagal et al. [Zag04; Zag06] used the Alice software [Ali], running on standard desktop computers, to allow children aged 11 and 12 to create 3D animations telling a fable. The storytelling was prepared and thought out in advance, which contrasts with the more spontaneous storytelling that is the focus of our work.

2.6 Summary

The most directly relevant literature for our work is indubitably that on 3D interaction on tabletops by Hancock et al. [Han07b], since our work can be seen a use case for their interaction techniques, and also extends them. The work of Wilson et al. on the use of physics simulations on tabletops [Wil08] is also highly relevant, and to a lesser degree also the work on BumpTop by Agarawala and Balakrishnan [Aga06], which does not use tabletops or direct-touch input, but does show how interactions beyond translating and rotating could be integrated. We make use of the FTIR technology developed by Han [Han05], and it is good to keep the possibilities and limitations of this technology in mind, but we do not in any way extend his results.

There is one body of related work that was intentionally left out of this chapter, which is the work on sandtray therapy. Because this thesis is about computer science, we will not reference any psychology literature, but instead present only the aspects that are relevant to our work in a somewhat less formal manner. This overview of sandtray therapy is given in the next chapter.

Chapter 3

Sandtray therapy

Because we attempt to create a tool that could be used in sandtray therapy, it is necessary to have some understanding of what sandtray therapy actually is. This chapter describes the sandtray, the figurines, and the way in which a therapy session is conducted. Since the focus of this thesis is on computer science and interaction, not on psychology and therapy, we will not go into the details of the working of this type of therapy. Instead, we focus mostly on the mechanics, as gathered from correspondence and interviews with sandtray therapists and various sources on the web.

An overview of the concepts of sandtray therapy is presented in Section 3.1. A description of the physical tray of sand and its varieties are given in Section 3.2. The figurines used in sandtray therapy are described in Section 3.3. Finally, Section 3.4 gives an impression of how a sandtray therapy session is conducted.

3.1 Overview

Sandplay therapy is a form of therapy introduced by Dora Kalff [Kal] in the 1950s. The patient plays in a sandbox using a range of toys and other objects, dry and wet sand, and sometimes water. Though it is possible to create a static scene in a sandtray, patients are encouraged to use the sand and the figures to act out a story. This playing has a healing effect in itself, and may also serve as an expression of the subconscious, to be analysed afterwards. An example of a sandtray is shown in Figure 3.1.

The further psychological background and details of this form of therapy are outside the scope of this thesis, although we will refer to the psychological aspects of our application whenever these are directly relevant to the design and implementation itself.

The terms ‘sandtray therapy’ and ‘sandplay therapy’ are sometimes used interchangeably, but they do refer to different things. Kay Bradway writes about this distinction [Bra06] and suggests that the term ‘sandplay’ be reserved for the specific form of therapy that Kalff developed, while ‘sandtray’ should be used for any form of therapy involving sand, water and miniatures. In line with this suggestion, we will use the term ‘sandtray’ throughout this thesis.

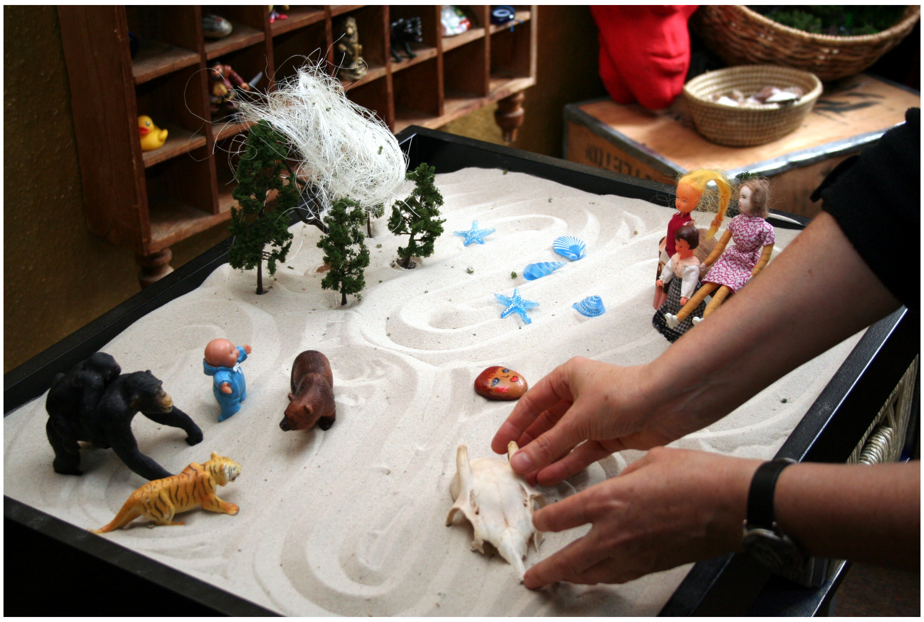


Figure 3.1: A sandtray showing several different objects [Wal08a].

3.2 The sandtray

The original sandtray used by Kalff [Kal] measures approximately 72 by 50 centimetres and is 6 centimetres deep. Its bottom is painted bright blue to suggest the presence of water. The fact that the tray is placed horizontally immediately creates a suggestion of 'ground', inviting the creation of a scene or landscape. The edges of the tray form a clear boundary between the real world and the play world, delimiting the fantasy.

Dry and wet sand can be used to fill the tray. Sometimes water is also provided. The sand can be used to draw in, to form a landscape with mountains, valleys and rivers, or to create shapes. In this sense, the sand is a very open and expressive medium. It can also be used to partly or completely bury objects, which has several profound psychological interpretations.

Not just the visual aspect of the sandtray is important; the other senses also play a role. The feel of the sand and the texture of the figurines contribute to the experience, but sounds and even smell can also be relevant. Since the therapist will generally object to the patient eating the sand, the sense of taste does not play any significant role.

3.3 The figurines

The objects that the patient plays with are often called 'miniatures', 'figures', 'figurines', 'toys' or simply 'objects'. We feel that the word 'object' is too general (especially in a programming context) and the others are too specific, and have arbitrarily adopted the term 'figurine' to refer to the objects that the patient plays with. Figure 3.2 shows an example of shelves with various figurines.

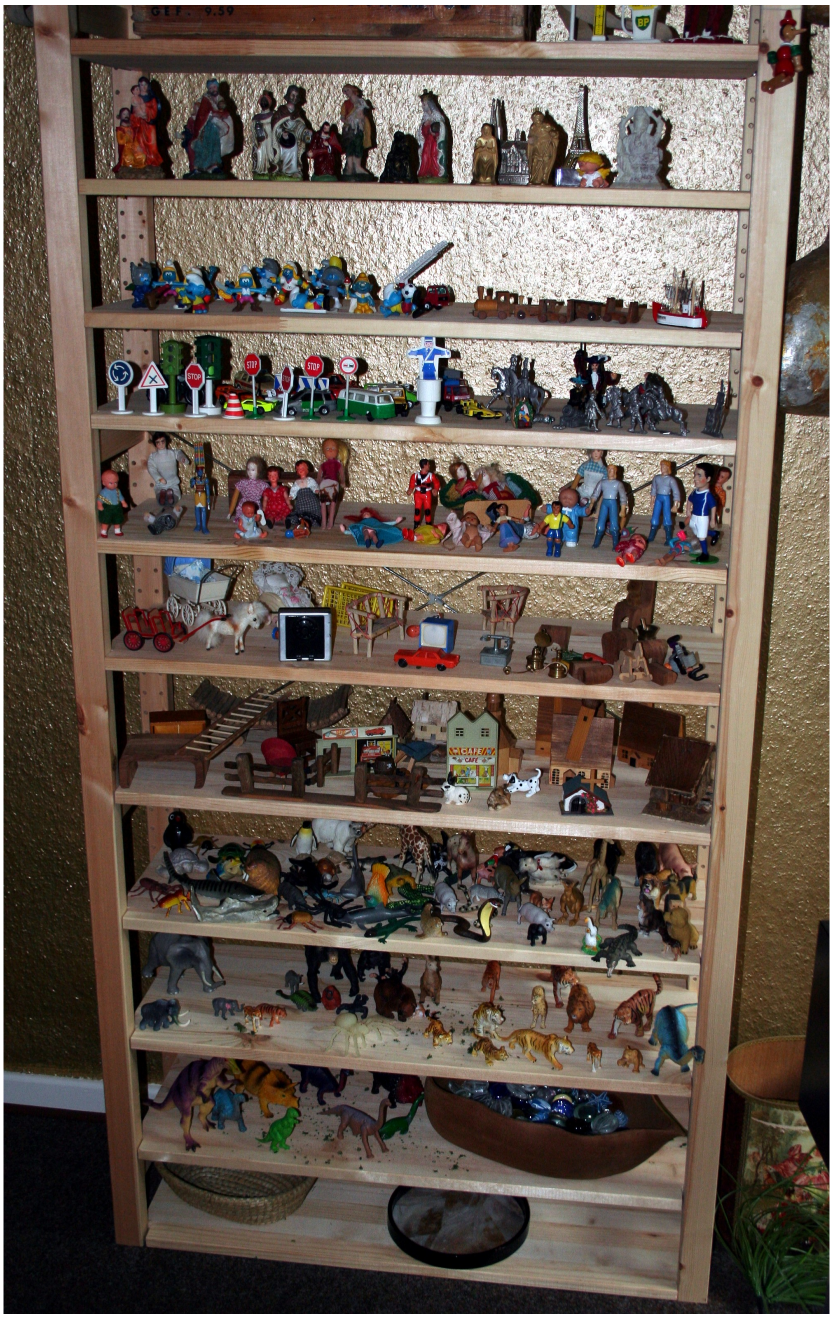


Figure 3.2: Shelves filled with figurines [Wal08b].

The main purpose of the figurines is to provide a medium to play with. However, another purpose is suggestion: the various figurines may evoke certain reactions in the patient, suggesting a certain scene or story.

Such suggestions evoked by a figurine come in two flavours. Firstly, there is the archetypical association that most people have when seeing a certain figurine. Items with symbolic value, such as a key or an hourglass, can therefore play an important role. Another symbolic aspect of the figurines are dualities, such as large/small, heavy/light, fast/slow and good/evil. Secondly, the association that a certain person has with a figurine will be unique for each person. A spider will evoke a different reaction in a biologist than it will in someone with arachnophobia. Many figurines will also have a symbolic purpose, representing entities from the patient's world, or even the patient themselves.

The set of provided figurines should be as broad as possible, with representations of all kinds of objects and creatures from both the real and imaginary worlds: people, animals, fairytale creatures, vegetation, stones, houses, vehicles, et cetera.

Figurines from the outside environment can also be brought in. For example, a stick could be broken into pieces which would serve as swords for the characters, a handful of pine cones of different sizes could represent a family, an empty cigarette box could serve as a cradle, or a piece of cardboard could be folded to form a house.

The size of the figurines is associated with their perceived power or relevance, and therefore a range of sizes of figurines is provided. Having more than one of the same or a similar figurine can also be important if the patient wants to depict a family or other kind of group. Furthermore, figurines can be modified in many ways: they can be stacked, taped together, or painted, to name just a few of the possibilities.

Some therapists offer the figurines in boxes, with no particular organization at all. Others present them on shelves or group them into categories, such as fantasy, western, household and sports. This is largely a matter of the personal style of the therapist.

3.4 The sandtray session

Most often, the patient is a single person. However, sandtray therapy can also be used for problems within couples or entire families. In these cases, multiple people will be playing at the same time, (hopefully) interacting with each other. Although we will use the singular 'patient' throughout this work, it is good to keep in mind that this word may actually refer to multiple people.

The role of the therapist during play is usually a passive one. The therapist will never touch the sandtray or even offer suggestions, unless explicitly asked to do so by the patient. However, therapists' styles differ, and some therapists will in some cases ask the patient to depict the situation that is troubling them, or take turns with the patient in putting things into the sandtray.

After the patient is done playing, a picture can be taken of the final situation in the sandtray. The therapist will sometimes discuss the story or scene with the patient, asking questions like "Why did you put this witch there?" or "What do you think about the snake?" In some cases, the patient will also clean up and take the scene apart under observation of the therapist. The order and manner in which the patient disassembles the scene can lead to insights.

3.5 Summary

We have seen that a sandtray is a shallow box filled with sand, discussed why it looks like that, and what the possibilities of a sandtray are. We showed that all kinds of objects are used to play with in the sandtray, and we know that this playing is framed within a therapy session with the therapist playing a mostly passive role. We now have sufficient information about physical sandtray therapy to begin designing its virtual counterpart.

Chapter 4

The virtual sandtray

This chapter discusses the global design of a virtual sandtray. First, we limit our design space by the available hardware, described in Section 4.1. From the information presented in this section and in the previous chapter on sandtray therapy, we derive in Section 4.2 general guidelines and considerations to be kept in mind during the design. In Section 4.3, we list the possible features that a virtual sandtray could have, and select a subset of these for implementation and investigation. We then proceed to describe the design itself. Section 4.4 discusses how to project the three-dimensional sandtray onto the two-dimensional tabletop screen. Section 4.5 discusses globally what the virtual sandtray will look like. Finally, Section 4.6 will motivate the need for a physics simulation engine and describe how it is used.

A note on language: though the people interacting with the sandtray can be either male or female, we use the female pronoun exclusively to avoid awkward constructions.

4.1 Hardware

As discussed previously, we design our sandtray application for readily available tabletop hardware, and do not use any other devices or extensions. It is therefore necessary to have a good understanding of the affordances and limitations of the pieces of hardware used in this project.

Two pieces of hardware are used for the development of this system: a table based on the SMART Board, and the SMART Table. Since the design of our application is partly governed by the possibilities and limitations of the hardware used, we describe both devices in detail in the following sections.

Development began on the larger, two-touch SMART Board table, but moved to the smaller SMART Table as soon as it became available for this project. Most of the design was done for the newer SMART Table, which is the reason that three-touch techniques (as discussed in the upcoming Section 5.3.3) are used without restraint.

4.1.1 SMART Board table

The larger of the two touch sensitive tables used in this project is a table based on the SMART Board 'for Flat-Panel Displays' [SMAa]; see Figure 4.1. The viewable area of this table is approximately 146 by 110 centimetres in size. Four projectors underneath the table each project a quarter of the image on the table surface, leading to visible seams. The projectors each have a resolution of 1400 × 1025, leading to a total resolution of 2800 × 2100 at approximately 50 ppi (pixels per inch).

Touches are detected above the surface by SMART's DViT technology, which uses four infrared cameras in the corners of the table. This method allows only two touches to be detected at any given time; the presence of more than two touches will lead to unpredictable results. Moreover, objects do not need to touch the surface in order to be detected, which sometimes leads to false positives, for example when some fingers are bent underneath the hand and the knuckles come too close to the table surface.

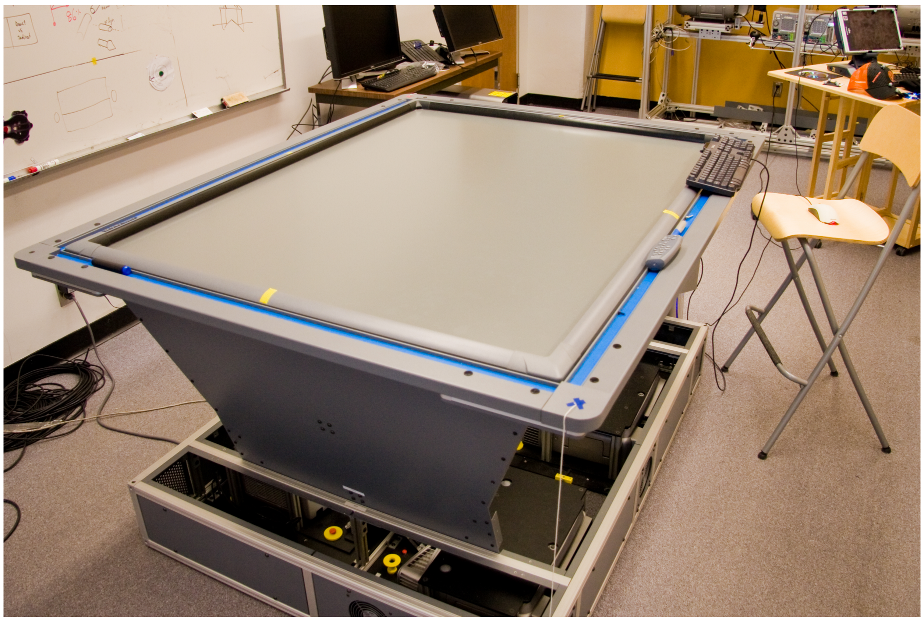


Figure 4.1: The table based on the SMART Board.

4.1.2 SMART Table

The other piece of hardware used in this project is the new SMART Table[1] [SMAb] shown in figure Figure 4.2. This table has a viewable area of 59 by 44 centimetres. A projector underneath the table projects an image of 1024 × 768 pixels via two mirrors, leading to an approximate resolution of 44 ppi.

[1]The naming of these SMART products may be confusing; in particular, the large 'table based on the SMART Board' described in Section 4.1.1 is a custom-built device, and should not be confused with the smaller 'SMART Table' from Section 4.1.2, which is a consumer product.

Touches on this table are detected using frustrated total internal reflection (FTIR) [Han05] using a 60 Hertz, 640 × 480 pixels infrared camera placed underneath the table. FTIR theoretically allows for an infinite number of touches to be detected simultaneously. The practical limit is dictated by processing power and is, according to the manufacturer, 40 simultaneous touches. Assuming that the entire camera image is spanned by the tabletop screen, the resolution of the camera is 27 ppi, or slightly less than 1 mm per pixel. Subpixel processing can theoretically improve this precision; however, we found that the points detected by the software, even after calibration, can show a consistent error of several millimetres in certain regions of the screen. However, this error is negligible compared to the size of a fingertip, and will not be noticed in most cases.

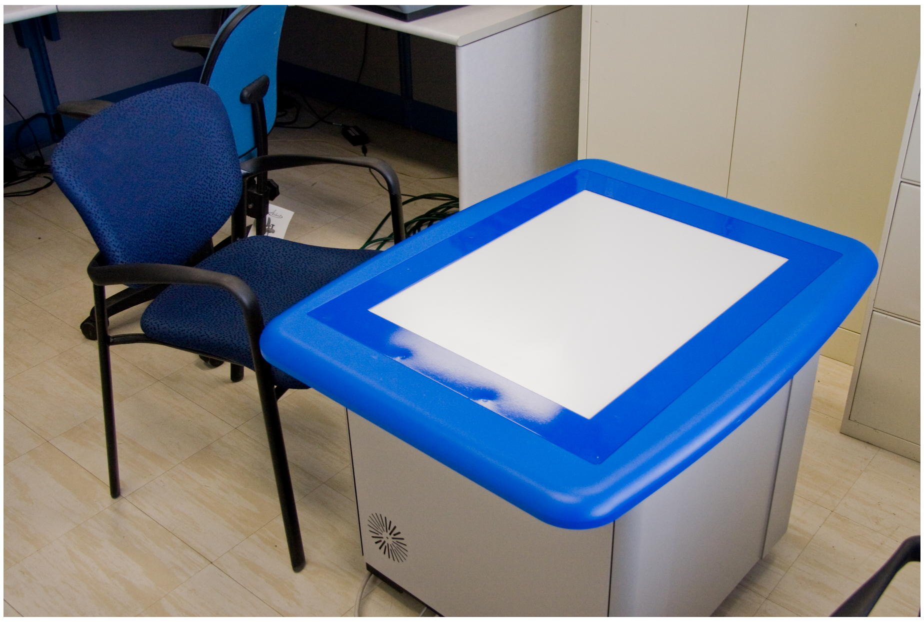


Figure 4.2: The SMART Table.

4.2 General design considerations

Sandtray interaction can be described in just one sentence: the patient plays in a tray with sand, water and a wide range of figurines. The generality of this description implies how broad the range of possible interactions actually is: the patient can interact in the full six degrees of freedom, use all ten fingers and other parts of the hand or even body, bring in outside objects, etcetera. We will have to severely limit this freedom in order to create a virtual sandtray on a tabletop. Instead of taking the physical sandtray and taking possibilities away to come up with a workable design, we take a bottom-up approach: starting with nothing, we add features until a sufficiently rich counterpart of a physical sandtray has been reached.

The patient should be actively involved with the scene she is constructing, and not with the interface that is used to construct that scene. It is therefore of vital importance that the interface is as natural and unobtrusive as possible. This is the first and most important consideration that guides our design. Whenever it is possible to trade off some power for a greater naturalness of interaction, we should probably choose to do so.

On the other hand, the interaction must not be too limiting. The patient is acting out a story, and if she feels too much constrained by the limits of the system, she will not be able to express herself adequately and the quality of the therapy will suffer.

4.2.1 Multi-user interaction

In most cases, only one person will be interacting at any given time. The role of the therapist is mostly passive. However, the patient can ask the therapist to participate in the play or help out when she is having difficulties, and in those cases it becomes necessary for the system to support multi-user interaction. This is also necessary when couples or families are using the application, or when we view the application as a general storytelling tool. We must therefore design the application to support multiple people interacting at the same time.

An important consequence of this assumption is that there can be no global mode switching. In other words, there must be no way in which the actions of one person can affect the interpretation of the touches of another. (Note that neither of the systems we design for are able to determine who is touching.) This excludes, for example, the paradigm of 'tools' that is common in paint programs on desktop computers.

4.3 Features

The digital world, and in particular the virtual sandtray, offers many possibilities that the physical does not allow. We must consider carefully which of these possibilities are worthwhile to implement. Inversely, the physical sandtray also allows for things that are extremely difficult or impossible to replicate in the digital realm. Yet, some of these are essential to sandtray therapy, so we must find an appropriate approximation in the digital world.

These considerations give rise to a list of features that will be considered for inclusion in the virtual sandtray. Because it is impossible within the scope of this project to implement everything on the list, a selection must be made. The full list of potential features will be presented shortly, but because we want to avoid repetition or excessive cross-referencing, we first discuss the criteria that are used to select a subset of the list for implementation. We can then discuss each potential feature at the point where it is mentioned.

4.3.1 Selection criteria

The items in the feature list give rise to the following question: which of these options provide an actual benefit for the therapy? Since this is a profound question that would require much research, most of which is in the field of psychology and not in computer science, we will not address this question further. However, our communications with sandtray therapists will allow us to make a selection. Our first selection criterium is thus: features should be beneficial for therapy.

Another question raised by reading through the upcoming feature list is: what interaction techniques should be used to provide these options? This is a question well within the field of computer science and, more specifically, interactions research, and will be the primary focus of this thesis. Our second selection criterium is therefore: features should allow for the investigation of new interaction techniques.

A last question that can be asked about some features is: can it even be done? It is of little use to try and do the impossible, or spend a disproportionate amount of time on the implementation. Our third selection criterium is thus: features should be feasible.

In the following two sections, we will present the list of features that were considered for implementation. Section 4.3.2 gives the list of features that will be implemented, together with a motivation; Section 4.3.3 lists features that will not be implemented within this project.

4.3.2 Features to be implemented

Below is a list of the features that will be included in the sandtray prototype, each with a motivation for its inclusion. For each feature, forward reference is included to the section in which the feature and its interaction issues are discussed in more detail.

- There is a clearly defined and delimited space that represents the sandtray.
  The fact that a sandtray is a clearly delimited, self-contained space is important, because it allows the patient to be above and outside the constructed scene. The boundaries of the tabletop will provide a clear boundary to the space. Camera movements and viewpoint changes are ruled out by this decision. Section 4.5 goes into more detail about the way the virtual sandtray is shown.

- Objects can be placed in the sandtray, moved and rotated, and removed.
  Obviously, without these basic abilities, not much of the original 'sandtray' concept would remain. More about the interaction with figurines can be found in Chapter 5.

- Objects in a virtual sandtray can be duplicated, so there is (theoretically) no limit on the number of copies created.
  From discussion with therapists, it follows that this is a worthwhile feature to consider: for example, it allows for the creation of a herd of cows, a family of people, or a forest. It should also be easy to implement this in an intuitive and unobtrusive way. Duplication is an implicit part of the figurine selection process and is described in Section 5.2.4.

- Objects can be grown, shrunk or otherwise modified in ways that physical objects do not allow.
  Resizing of objects was seen by therapists as a very useful option, because larger objects are perceived as more important, more powerful or more threatening. The interaction technique used to accomplish resizing can

and will be an interesting question in the field of interactions research and is addresses in Section 6.3. Due to time constraints, other transformations will not be considered.

- The sandtray floor can be 'painted' with different colours or textures.
  In a physical sandtray, the sand itself can be sculpted, piled up or dug into, which allows for the creation of an environment in which the figurines are placed. This creation of an appropriate 'backdrop' for the story is part of the storytelling process. However, it is difficult to provide simulated sand, and more difficult to allow for natural interaction with it. Although this would be a very interesting direction for future research, we will sidestep these issues and provide the much simpler 'painting' ability. The appropriate interaction technique for painting may not be as obvious as it seems, and is discussed in depth in Section 6.4.

4.3.3 Features not to be implemented

Many features were considered for implementation, but were discarded for various reasons, often due to time constraints. However, since many of them would provide interesting directions for future research, we briefly mention them below. We distinguish features related to the figurines themselves, the environment and others.

Figurines

- Objects can be buried beneath the surface.
  In a physical sandtray, burying objects is an action with important psychological connotations, and something similar should be provided in the virtual sandtray. With the interaction technique that was used, burying could be made possible by simply pushing them down until they are below the surface. However, there should also be some way to dig the object up again, and maybe some indication that something is buried. These open issues could not be addressed due to time constraints.

- Objects can be animated, either by predefined animations or by user-defined motions.
  Pre-animated 3D models were not readily available to us, and it is unclear whether they would provide a benefit for therapy over static models. User-defined animation would be an interesting direction to explore, but is too large a topic to tackle in this project.

- The influence of gravity can be changed, either globally or per object, to allow for light or even weightless objects that can be suspended in the air, possibly in combination with air resistance.
  Though different gravity and air resistance could easily be implemented for different models, there does not seem to be an appropriate real-life metaphor that can be turned into an interaction technique to change these properties dynamically. We would therefore have to resort to unnatural controls like buttons or sliders.

- Cloth and other soft, deformable objects can be added.
  The physics engine does support these, but they are difficult work with and

often show unpredictable or unstable behaviour. Moreover, not all interaction techniques used for rigid bodies are well-defined on deformable bodies, with the notable exception of the technique by Wilson et al. [Wil08].

- It can be possible to stick or glue objects together.
  This frequently happens in physical sandtray therapy through the use of tape or glue. However, given the versatility of those media, they are difficult to replicate on a tabletop. Instead, we could allow for sticking objects together by simply moving them both at the same time so that they intersect. Moving both objects apart independently could then be used unstick them. This would be an interesting feature to investigate, but requires more experimentation to determine whether it is feasible.

- Figurines can be coloured or painted.
  This would allow for considerably more creative freedom. However, this requires much work to determine the best interaction technique that can make this possible.

Environment

- A simulation of sand can be used instead of a flat surface.
  Physical sand was used by Wang et al. in their SandScape project [Wan02], but the dynamics are difficult to replicate in a physics simulation, and might also slow the system down too much. It is not clear how the richness of interaction with real-world sand (digging, piling, scraping, tunneling, ...) can be provided by a two-dimensional touch surface; however, this would be a very interesting question to explore in future research.

- Lighting conditions can be changed, allowing for more appropriate lighting to set a particualar mood for the scene.
  This would provide the patient with more expressive power, and could be implemented simply by adding 'lamp' objects to the scene. This feature was not added due to time constraints.

- The point of view can be made changeable.
  According to therapists, this would either give the patient a new perspective on the scene, or disrupt her own relation with it. Because of its questionable benefit this feature was not included.

- A virtual camera can be placed inside the sandtray, or one could look through the eyes of a figurine, allowing the patient to view the scene from the inside.
  This suggestion was considered by the sandtray therapists to be a very compelling option. It would evoke empathy with that particular figurine, and would be useful not only for therapy but also for education. However, we either need to place that view somewhere on the tabletop screen, which would partly or completely hide the 'objective' view of the sandtray, or provide a second, vertically oriented monitor.

- Sound effects can be used to enhance the physical feel of the scene.
  Though a good idea for an actual application, this does not add much value to our prototype, which is mainly used for interactions research.

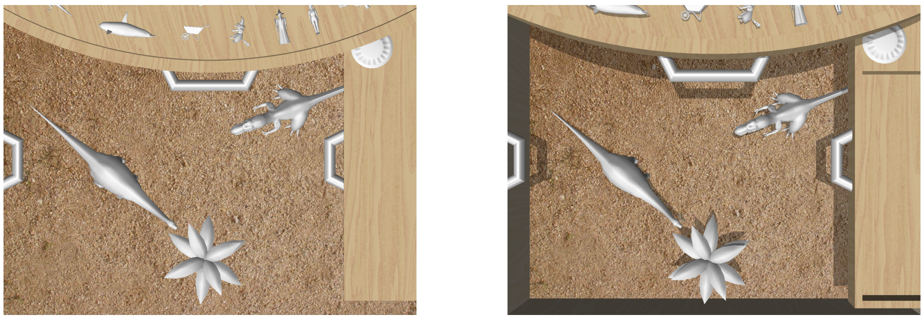

(a) Parallel projection (b) Perspective projection

Figure 4.3: Comparison of two common projection modes on the same scene. Because the light source is directional and straight above the scene, shadows are hardly visible in the parallel projected image.

Other

- A sandtray therapy session can be stored and replayed without the need for a video camera.
  This can be less intrusive for the patient and is therefore quite desirable. However, since we are only working on a prototype and not on a real application, and since this feature would not be interesting from an interactions point of view, we did not implement it.

4.4 3D projection

Now that we have made a selection of the features we do and do not want, we can begin to make decisions about what the actual sandtray program will look like.

A first choice concerns the method of projection. The problem of projecting a three-dimensional scene on a screen that is only capable of displaying a two-dimensional image is well-studied in computer graphics. This section discusses some of the choices made concerning the 3D to 2D projection process.

4.4.1 Projection type

Desktop applications that display three-dimensional scenes on a two-dimensional display screen commonly employ linear perspective projection, which makes objects farther away from the viewer appear smaller. In some special-purpose applications like CAD programs, parallel or ortographic projection is also used. This form of projection does not make faraway objects look smaller, but an advantage is that parallel lines in the scene will appear as parallel lines in the projection. The two projection methods are shown side by side in Figure 4.3.

When the depth of the scene (perpendicular to the screen) is small, such as in the case of our sandtray application, both projection types will give similar results. However, using a perspective projection might still result in a slight depth cue. Another compelling reason for choosing perspective projection is the interaction technique used to lift objects, as discussed in Section 5.3.5.

4.4.2 Viewpoint

In nearly all desktop applications, it is assumed that the point of view, and therefore the viewer, is somewhere straight in front of the centre of the screen. If the screen is viewed from the side, the image looks distorted. Usually, this does not pose any serious problems, and without special head-tracking devices (which can be as simple as a Nintendo® Wii™ remote [Lee07]) the software developer often has no other choice.

The assumption that the viewer's eyes are located straight above the centre of a tabletop computer, however, is more questionable. It might be more sensible to assume a viewpoint somewhere off to the side of the table, on the same side where the patient is standing or sitting. However, although we are often dealing with a single person interacting with the system, the therapist will be viewing the scene from the other side. Moreover, such an off-axis projection might be confusing for people who are used to the graphics displayed in, for example, computer games. For these reasons we decided to use a standard, on-axis projection.

4.4.3 Depth cues

To give the viewer a sense of three-dimensionality, even though she is looking at a two-dimensional image, several techniques have to be combined. One of the first and most important, perspective projection, has already been discussed.

Another obvious depth cue is occlusion: an object that partly covers another object will be perceived as being in front of the other. This can be trivially implemented using the depth buffer of 3D rendering libraries such as OpenGL.

To make objects look three-dimensional instead of flat, a suitable lighting model can be used. Surfaces that face directly towards the light will appear brighter than surfaces turned more away from it. This, too, is trivial to implement using modern 3D graphics libraries.

After these depth cues are added, the scene still lacks a certain sense of depth. Although the figurines themselves look three-dimensional, it is difficult to judge how high above the sandtray floor a figurine is positioned. This can be solved by a technique slightly more advanced than the previously mentioned ones, namely shadow casting. If the object casts a shadow on the floor, the distance between the object and the shadow gives a strong indication of the distance between the object and the floor.

Because of the perspective projection, objects will not remain in the same place on the screen when they are dropped from a height. Instead, they follow a path inward to the screen centre. To indicate where the object will land when dropped, we put the light source that casts the shadows straight above the table, and infinitely far away.

It would be even better if the shadow edges would soften as the distance between the shadow caster and the receiver becomes larger. However, this is difficult to implement and has not been attempted in this project.

4.5 Screen layout

The sandtray proper will be shown as an area underneath the table surface. We will therefore see the inside of its walls as a frame around the sand. These walls

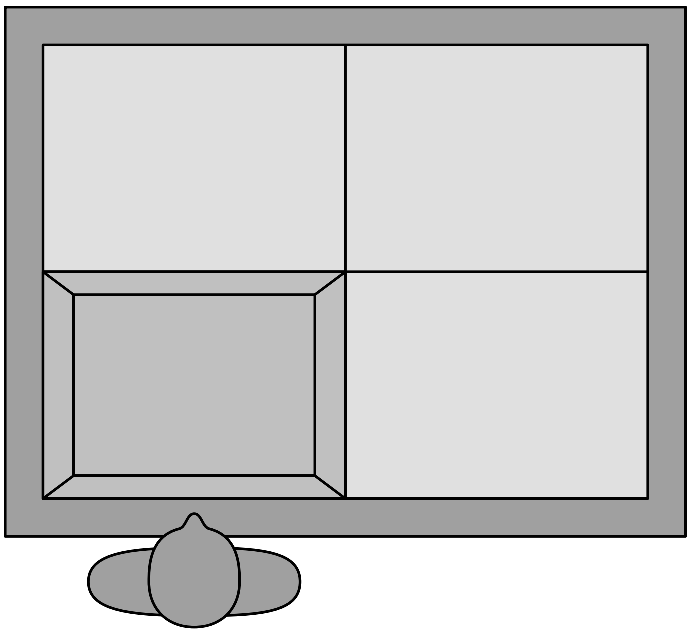


Figure 4.4: The four projection areas on the SMART Board table. Only one of these is taken up by the sandtray.

serve to enhance the perception that the interaction is going on *underneath* the table surface, rather than *on* it (see Figure 4.5 on the upcoming page 40).

The particular angle from which the sandtray is observed can have an effect on the perception of the scene. Therefore, we prefer that the viewer either has a fixed perspective (that is, she is on the same side of the table at all times), or is able to change her perspective at will, but without ever being forced to do so.

On the large SMART Board table, it is difficult even for a tall Dutchman to reach across the table to the other side. Since we do not want to force the viewer to walk around to the other side, thus forcing her to change her perspective, the actual interaction has to take place within a limited area. As the seams between the images from the four projectors naturally suggest boundaries, we decided to reserve one of these areas for the sandtray proper; see Figure 4.4. The other three areas remain available for displaying other information. However, we must keep in mind that these areas are difficult to reach and thus to interact with.

The smaller SMART Table measures 59 by 44 centimetres, somewhat smaller than Kalff's original 72 by 50 centimetre sandtray. The best approximation, therefore, is to have the sandtray take up the entire screen. This leads to the question how we can access features that do not reside in the sandtray itself, such as the storage space for the figurines.

4.5.1 Drawers

Common solutions in desktop applications to the problem of not having enough screen real estate include pop-up menus, dialog windows, folding panels docked to the side of the window, and floating tool palettes that can be moved partly offscreen. Our solution is similar to the latter. However, instead of using the

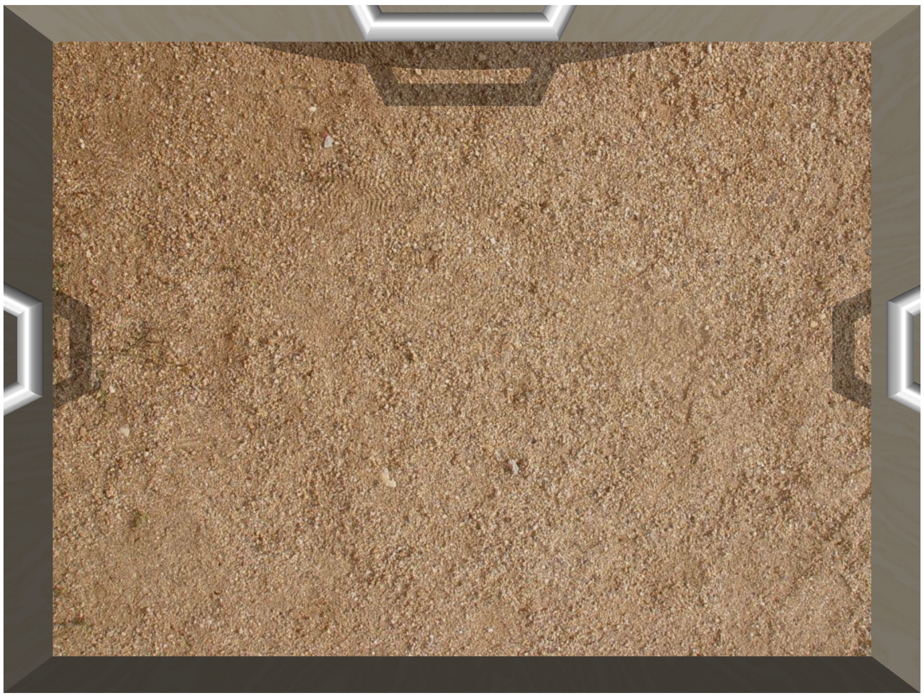


Figure 4.5: The empty sandtray, right after the start of the program. The walls of the sandtray are shown, as well as the handles of three drawers that reside off the sides of the screen.

abstract concept directly, we used a more physical metaphor, namely that of drawers. The role that is played by the title bar of a tool window is here played by the handle of the drawer. The main difference is that the drawer is more constrained in its movement: it can only move along a single axis, and there are limits to how far it can move along that axis.

The drawers can be partly or entirely offscreen, but have a handle protruding into the visible area that can be pulled to slide the drawer out into view. This concept is especially useful on the smaller SMART Table, where the entire screen will be taken up by the sandtray. On the large SMART Board table, the entire drawer might be visible in one of the three auxiliary areas, but since these might be hard to reach, the handle can be used to slide the drawer towards the storyteller in order to interact with its contents.

Figure 4.5 shows the sandtray in its initial state, on a system where the drawers reside offscreen. The look of opened drawers depends on their content. Open drawers in action will be shown later, in Figures 5.3, 6.2 and 6.4.

4.6 Physics simulation

To give the viewer as much as possible a sense of realism of the virtual sandtray, it is not sufficient to make it merely look realistic. It is of equal importance to make objects act and interact in a way that resembles the physical world as closely as possible. For this purpose, we employ a physics simulation engine [NVI]. The ‘bowling’ video referred to in appendix A demonstrates the use of the physics engine (although its use is also evident in the other videos).

Although it would be possible to have only the figurines in the actual sandtray be controlled by the physics engine, we chose to adopt a more radical approach. Nearly every object in the application is also an actor in the physics engine, including walls, drawers, figurine representations inside the drawer, and the painting hose (see Section 6.4). This approach ensures that everything in the application responds to outside interactions and to other objects in the most natural way possible. Drawers will retain their momentum when flicked, figurines put back into the drawer bounce realistically around the other figurines inside the drawer, and the painting hose will find its way in between the figurines inside the sandtray.

However, for the greatest possible usability, some concessions to physical realism have been made. These will be discussed in their respective sections: in particular, Section 5.1 and Section 6.4.2.

The use of a physics simulation in an interface and the interaction with that interface is a recurring theme throughout this work. Although this is by no means the first interface to employ physics [Frö00; Aga06; Wil08], this and all previous applications are ad hoc, lacking a general theoretical framework or design guidelines. This work does not attempt to present such a framework, but does contribute one more point to the design space.

4.7 Summary

In this chapter, much of the design of the virtual sandtray was discussed. We presented the SMART Table on which primary development of the prototype takes place. We motivated the importance of a natural, fluid interface and the fact that we should not assume single-user interaction. Then we listed many possibilities that a virtual sandtray could offer, and selected a subset of these for implementation and investigation. We then made some decisions regarding the projection of the 3D scene; in particular, the assumption of a viewpoint straight above the table, and the use of shadows. We decided where to place things on the screen and introduced the notion of retractable drawers to save screen space. Finally, the need for a physics simulation was motivated.

We will now turn to a more specific design issue, important enough to warrant its own chapter: the interaction techniques used to control figurines.

Chapter 5

Interaction with figurines

Most of the time spent with the virtual sandtray will be spent interacting with figurines. It is therefore of the utmost importance to ensure that this interaction is as smooth and unobtrusive as possible. In this chapter, we discuss how this is achieved.

First, in Section 5.1, we describe how we use the physics engine to make figurines respond in a natural way to each other and their surroundings. Section 5.2 describes how the collection of available figurines is organized and how figurines to play with can be selected. Finally, in Section 5.3, we present an in-depth discussion of how figurines can be moved around in the sandtray, and the choices that were made during the design of this interaction. This last section is one of the core parts of this thesis.

5.1 Figurines as physical objects

As said before, in Section 4.6, all figurines are also actors in the physics engine. This means that we get many real-world behaviours 'for free': figurines will fall when not resting on anything, bump into each other, knock each other over, bump into walls, etcetera.

Figurines are represented graphically as arbitrary triangle meshes. However, the meshes from the library we use need not be closed, connected or manifold, and the physics engine is fairly limited even in its support for 'well-behaved' mesh objects. We therefore present figurines to the physics engine in the form of a simplified convex hull; see also Sections 7.5.2 and 7.5.3 in the implementation chapter. This implies that highly nonconvex objects will sometimes collide when, visually, there is no contact between the two. In practice this turns out not to be a significant problem.

A more serious problem arises when trying to make tall figurines, such as people, stand upright. The 3D models we use were not designed for this purpose. More often than not, their centre of mass is somewhere outside the area of contact with the ground, so the figurines will fall over. This can be solved in different ways, all involving 'cheating' the physics engine:

1. When the object is close to being upright, apply a force or torque that pulls it towards the upright position.

2. Move the centre of mass to a different position.

3. Add a ‘pedestal’ at the bottom of the figurine to expand its area of contact with the ground, much in the same way as toy soldiers have.

We chose a combination of 2 and 3: invisible square pedestals were added as separate shapes at the bottom of unstable figurines. Because the pedestals have nonzero mass, they also pull the figurine’s centre of mass down somewhat. (The pedestals are square because the physics library does not natively support cylindrical shapes.)

Both the convex hull and the pedestals are illustrated in Figure 5.1.

5.2 Selection of figurines

The current sandtray application contains around 160 different figurines. These were selected from the Viewpoint e-Catalog, a large commercial library of 3D models. A list of these figurines is provided in appendix B. Even though the range of figurines might be adequate, it might well prove to be too limited; it is not unreasonable to imagine a collection of several thousands of models.

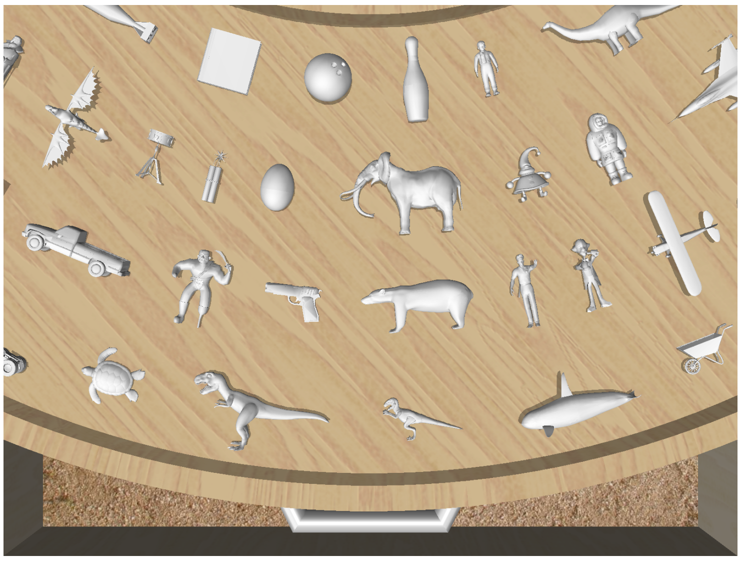
Because of the large number of figurines available, we need to devote some thought to how these can be found and selected. Different approaches to impose structure on and to display the collection are discussed in Section 5.2.1 and Section 5.2.2 respectively. The way to display individual figurines in the collection is discussed in Section 5.2.3. Finally, Section 5.2.4 discusses how the figurines from this collection can be selected for use in the sandtray.

5.2.1 Structuring

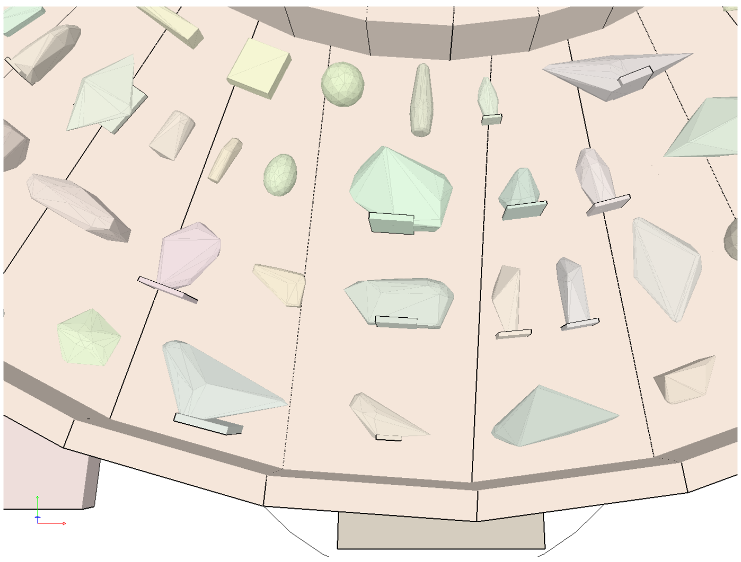
In general, there are several different methods that allow someone to select an object from a collection. The simplest is to present the collection in a ‘flat’ and unstructured way. Another possibility is a hierarchical approach, where objects are presented much like files in a filesystem, organized into directories and subdirectories. A third possibility is a labeling approach, in which multiple labels can be associated with an object, and subsets of the collection are shown based on a selection of labels. Combinations of these methods are also possible.

In the physical version of sandtray therapy, the figurines are usually on wall-mounted shelves or in boxes. Whether or not there is there any specific ordering or organization to them depends on the therapist. When figurines are presented in a more or less random way, the patient will form associations that would not be formed if a more structured search had been used. On the other hand, when the figurines are sorted by category, the patient will better be able to select matching figurines to depict, say, a fairy tale or a domestic scene.

Despite the fact that a large collection of figurines may become unwieldy, we chose to present figurines as one large set, without any imposed ordering or structure. The reason for this choice is simply that we were not aware at the time that not all therapists use the same method. Arranging figurines in different groups would be a fairly trivial extension.



(a) On the screen.



(b) In the physics engine.

Figure 5.1: An illustration of how figurines are shown on the screen, and how they are represented in the physics engine. Note the convex hulls and the pedestals underneath some of the figurines.

5.2.2 Layout

To decide how to organize the figurine collection on the screen, we look once again at the physical case. A box or basket, in which figurines are piled up, would clearly be difficult to use when only two-dimensional input is available and it is not possible to use one's hands to dig into the collection. Thus, it is better to use a layout without any stacking, that is, constrained to at most two dimensions.

For any sufficiently large collection of figurines, the tabletop screen is not large enough to contain them all at the same time. Some form of scrolling or zooming, or a combination thereof, is therefore needed. Since zooming would give the impression of physically enlarging the figurines, this might reduce the sense of realism of the application. We therefore opted for a scrolling-only approach. We combine this with the sliding of a drawer, as discussed in Section 4.5.1. The figurine drawer can be slided into and out of view by dragging its handle as normal, but the floor of the drawer itself can also be grabbed and scrolled around with a single finger. Tossing (having it retain its momentum after the touch is released) is also possible, and the drawer will slowly spin to a halt.

In order not to bias against figurines placed in the far regions of the scrollable area, we made the drawer ring-shaped, as shown in Figure 5.2. As with a regular scrolling area, the largest part of the drawer is off screen. The size of the drawer is computed at program startup to accomodate all figurines. Figure 5.3 shows the final implementation of the ring-shaped drawer.

This design bears similarities to the concept of 'interface currents' by Hinrichs et al. [Hin05], which was inspired by Lazy Susan-type devices like corner cupboards and luggage conveyor belts. However, an interface current will flow continously, whereas our drawer will come to a halt as a result of friction. Whereas interface currents can be moved freely and resized, our drawer can only be moved in a constrained manner.

5.2.3 Representation

It is possible and in some situations beneficial to represent objects by something else than their physical appearance, such as a name, description or icon. However, in our case the most natural and unobtrusive way to display the figurines is simply to show their actual image as it would appear inside the sandtray.

The ring-shaped drawer initially contained only one row of figurines, at their full size. This decision was made based on the available screen space; it is certainly possible to show multiple concentric rows of figurines above each other on the screen, but they would have to be shown at less than their full size. Note that the figurines in the drawer appear larger than they would on the sandtray floor, because the drawer is higher up and thus closer to the viewpoint.

Based on later informal feedback, it was decided that the drawer needed to show more figurines on the screen at the same time. For that purpose, a concession to realism was made and the figurine representations in the drawer were shrunk to a fraction of their original size. The final sandtray prototype shows figurines in four rows, which gives a good tradeoff between the ability to see many figurines at once, and the ability to see each in sufficient detail and to touch them without requiring too much precision. See again Figure 5.3 for the

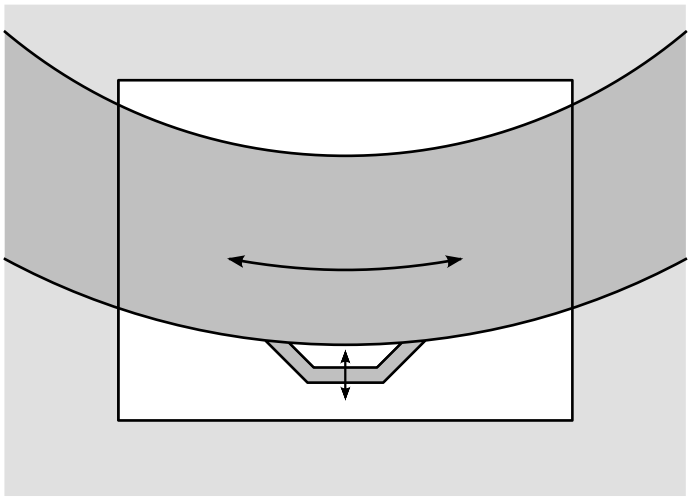


Figure 5.2: The circular drawer; the white rectangle represents the area that is visible on the screen. When the handle is dragged, the drawer will slide in and out. When the ring itself is dragged, it will spin around.

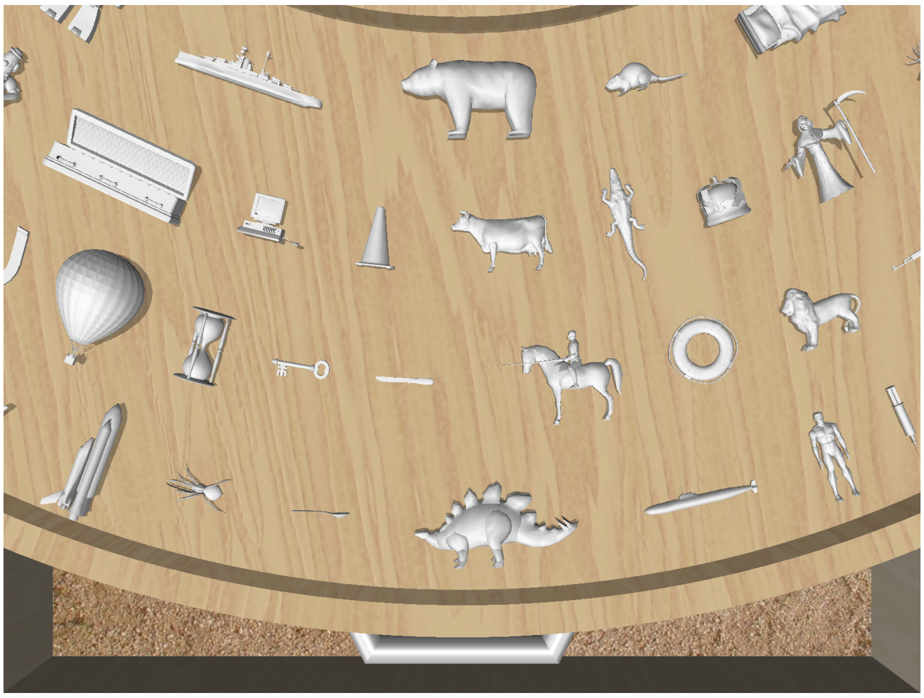


Figure 5.3: The ring-shaped figurine drawer in the sandtray program.

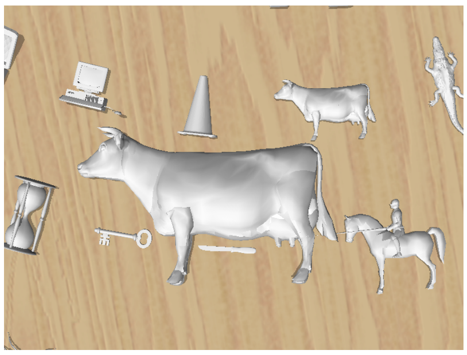


Figure 5.4: After tapping the cow figurine, a copy of it was made and grew to its full size. The copy rests on the drawer and can now be moved around.

resulting layout.

5.2.4 Selection

When a figurine has been chosen to be placed in the sandtray, it can be dragged out with a single finger from the drawer into the sandtray to place it there. As soon as the figurine is picked up from the drawer, its smaller representation will grow to its full size with a short animation to indicate what is happening.

When a figurine is dragged from the drawer, a copy remains behind. One can thus to create multiple copies of the same figurine, for example to create an army or a forest. This is a significant advantage over the physical limitations imposed by the real world. Figure 5.4 illustrates both the copying and the resizing.

The interaction technique for moving figurines out of the drawer is exactly the same as the technique used to move figurines within the sandtray. For example, it is also possible to move a figurine up and out of the drawer by placing two fingers on it and moving them apart. Moreover, figurines in the drawer are controlled by the physics engine in exactly the same way as those in the sandtray and will therefore give the same impression of being actual, physical objects.

5.2.5 Removal

Obviously, once figurines have been placed in the sandtray, there is a need for the ability to remove them again. Several interaction techniques could be chosen to do this:

- A button that, when clicked, makes the next touch delete a figurine. However, this would imply a mode switch, and thus break down in a multi-user setting (see also Section 6.1).
- A button that appears alongside a figurine. However, any kind of button would adversely affect the feel of physical reality.

- A gesture, such as crossing out the figurine. However, gestures are hard to discover; one has to be told about them. Moreover, the gesture might interfere with the normal way of interaction.
- A tool, such as a magic wand, that will delete any figurine it touches. This would integrate well with the concept of physicality, but might be difficult to use.
- Simply moving the figurine (partly) off the screen, as with the post-its in the collaborative application by Hilliges et al. [Hil07]. Though potentially a good method, it might also negatively affect the physical feel of the application if figurines can be pushed right through the sandtray wall.

Again, we took a hint from the physical world, in which figurines are not simply 'deleted'. Rather, they are put back on their shelf or in their box. We therefore implemented deletion simply by putting figurines back into the drawer. As soon as they move off the screen, they vanish. This does not require any additional machinery that clutters the interface, and proved to be a natural and even somewhat discoverable technique.

5.3 Moving figurines

To move figurines around in the sandtray, we use an interaction method developed by Hancock et al. [Han07b], modified to suit the needs of the sandtray application. This method provides direct control over objects. More indirect methods such as the physics-based proxy method by Wilson et al. [Wil08] do not provide sufficient control, and, moreover, do not allow for vertical movement of objects.

Hancock et al. describe three different interaction techniques, requiring one, two, or three touches, respectively. All three methods developed by Hancock et al. allow for the rotation and translation (RNT) of three-dimensional objects on a touch-sensitive surface in so-called 'shallow-depth 3D'. This means that, although the objects exist in a three-dimensional virtual space, their depth coordinate does not change; the objects' origin is constrained to a plane at a certain fixed depth. Therefore, there are five degrees of freedom that can be controlled: three of rotation and two of translation.

We will describe these three techniques briefly in the following three sections. For a more complete and precise discussion, refer to the paper by Hancock et al. [Han07b].

The descriptions of the original one-, two- and three-touch techniques are followed by a description of our own modifications of the three-touch technique, including a natural extension that also allows for vertical movement, and a discussion of the advantages and disadvantages of these modifications.

5.3.1 One-touch

With only a single touch point, the two-dimensional RNT method developed earlier by Hancock et al. [Han06] can be extended to three dimensions. It is possible to rotate and translate the object in three dimensions by dragging it

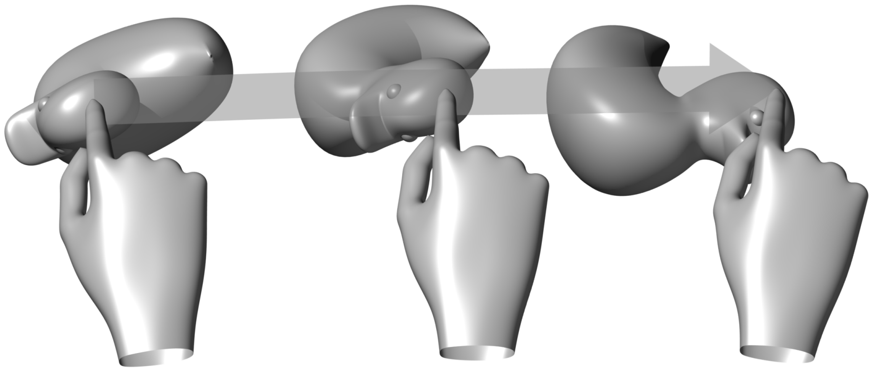


Figure 5.5: Illustration of the use of the one-touch interaction technique. Compare this to Figure 2.4 on page 21. The object is rotated and translated as if an opposing force is applied to the centre of the object. The point of contact remains under one's finger. When performing a straight motion, as shown here, the object's centre will eventually drag behind the finger. (The special regions for constrained movement are not shown in this example.)

with a single finger (3D RNT). The point of contact on the object remains under one's finger at all times, as if one has a 'sticky finger'.

This method assigns special significance to a point in the object, called its centre. This will ususally be the object's coordinate origin or its (perceived) centre of mass, but it does not have to be. Informally stated, the method acts as if there is a weight at the centre of the object: the object moves in such a way that the point of contact remains under one's finger, but rotation is preferred over translation while doing this. The example in Figure 5.5 illustrates the operation of the one-touch technique.

Formally, the method works as follows. The depth buffer is probed at the initial location of the touch to find a depth value associated with the touch point. This depth value is used to determine the 3D point of contact on the surface of the object. The same depth value is also used to interpret the final location of the touch. The object is rotated about its centre to bring the point of contact as close to the final touch location as possible, then translated in the x- and y-directions to bridge any remaining distance.

To perform more constrained interaction, two special regions on the object are provided. Touching the object in one of these regions enables certain constraints on the motion. One region provides translation only, much like the title bar in common windowing systems allows for translation of the window. The other region provides translation and rotation in two dimensions only (2D RNT, as in [Kru05]) like in Figure 2.4, restricting the rotation to be about the z-axis.

Although this interaction technique feels quite natural, its speed and accuracy are relatively low. Moreover, the need for special regions unnecessarily complicates the interaction technique. The use of more fingers at the same time can alleviate these problems, as the following section describes.

5.3.2 Two-touch

This technique can obviously only be used on surfaces capable of detecting more than one touch at the same time. Such multi-touch surfaces are becoming more mainstream, for example with the introduction of the Apple® iPhone™ and iPod® Touch. Both the SMART Board and the SMART Table used in this project are also capable of detecting multiple touches at the same time (although the SMART Board is limited to two).

When only a single finger is used with this interaction technique, it uses the 2D RNT method as described previously. That is, the object can be translated in the plane, but rotates only about the z-axis, so that the same side of the object remains facing upwards; see Figure 5.6a.

When a second touch is detected, this touch will be used to rotate the object about the global x- and y-axes. Moving the second finger in the y-direction rotates the object about the x-axis and vice versa. See Figure 5.6b for an illustration of this.

The interaction with the second finger feels as if one is rolling a trackball to rotate the object. Although this is a natural way to perform rotations, the second finger does not remain in contact with the same point on the object throughout the interaction, and in fact need not be in contact with the object at all.

Again, a special dedicated region is provided to perform a translate-only movement. To eliminate the need for this region as well, a third touch can be added, as described in the following section.

5.3.3 Three-touch

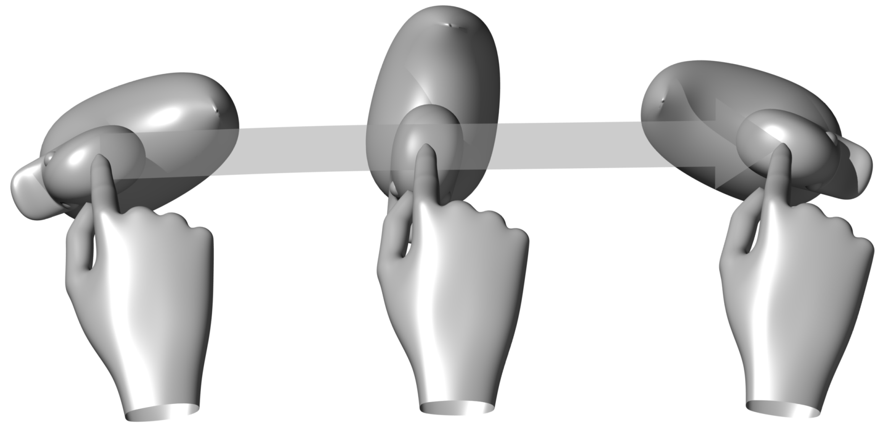
It is important to note that the SMART Board does not support more than two touches, so this interaction technique limits the hardware platforms to devices like the SMART Table.

This technique is demonstrated in Figure 5.7. The first touch will perform translation only, as shown in Figure 5.7a. The second touch performs rotation about the first touch point, but constrained to the z-axis; see Figure 5.7b. The third touch performs rotation about the x- and y-axes as described in the previous section. All these rotations are still done about the object's centre.

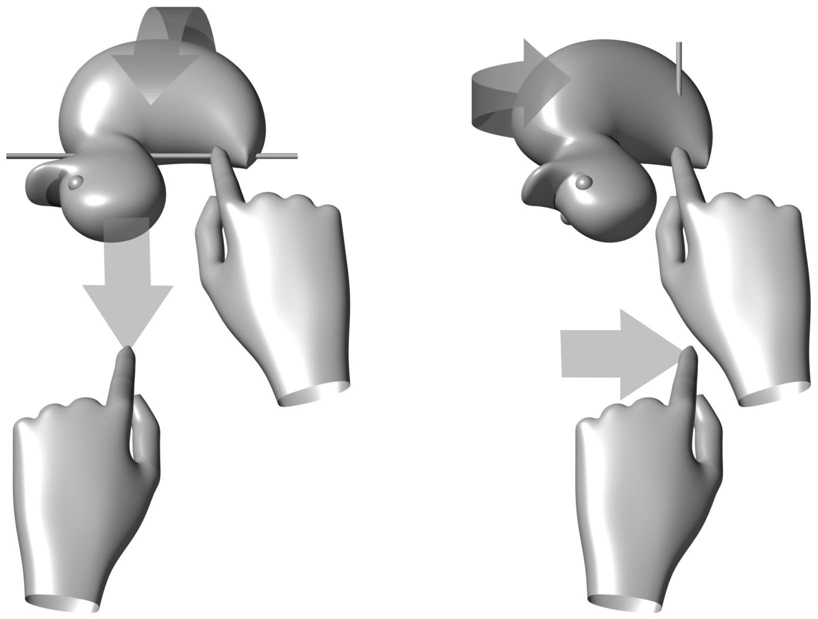
Although this may sound complicated and unintuitive, a user study by Hancock et al. [Han07b] indicated that subjects prefer the three-touch technique over the one- and two-touch techniques. Moreover, tasks are performed faster using the three-touch technique. We will therefore base our own interaction technique on the three-touch method.

5.3.4 Multiple objects and crossing

On a true multi-touch surface such as the SMART Table, it would be desirable if multiple people could manipulate different objects at the same time, or (especially in the case of the sandtray) if one person could manipulate two objects simultaneously using different hands. The methods by Hancock et al. do not support this, because the first touch defines an 'active' object, and all other touches are assumed to be related to this object. However, these methods can easily be extended to support the manipulation of multiple objects at the same

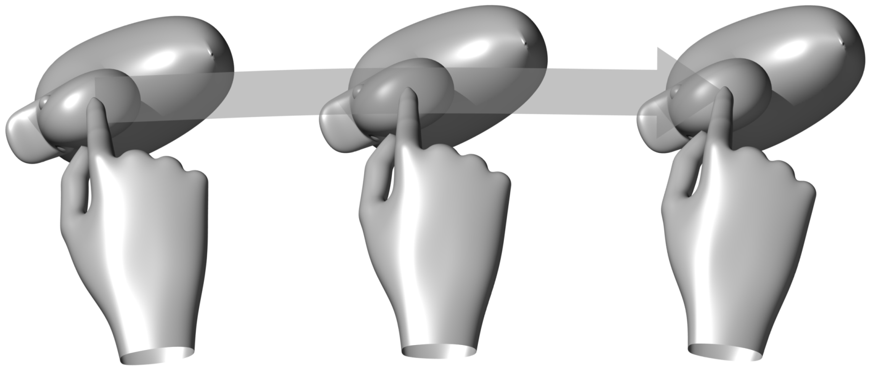


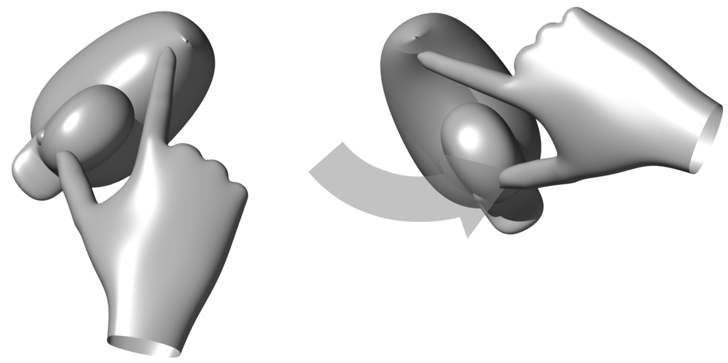
(a) In contrast with the one-touch technique shown in Figure 5.5, dragging with one finger will still perform a translation, but the object now only rotates about the z-axis.



(b) The second touch rotates the object about an axis in the xy-plane, perpendicular to the finger's motion. The second touch can be a second finger of the dominant hand, or (as shown here) a finger of the nondominant hand. In this example, the centre of rotation is the first touch point; we can also choose to rotate about the object's centre.

Figure 5.6: Illustration of the use of the two-touch interaction technique.


(a) Using the three-touch technique, the first touch will only translate the object.


(b) The second touch rotates the object about the z-axis. This can either be a second finger of the dominant hand (as shown here) or a finger of the nondominant hand.

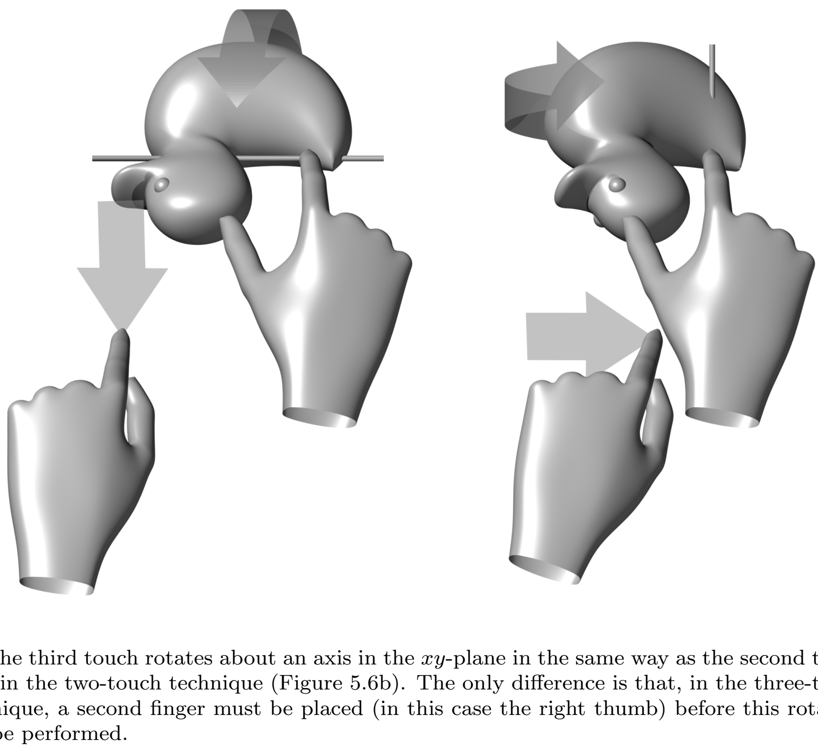

(c) The third touch rotates about an axis in the xy-plane in the same way as the second touch does in the two-touch technique (Figure 5.6b). The only difference is that, in the three-touch technique, a second finger must be placed (in this case the right thumb) before this rotation can be performed.

Figure 5.7: Illustration of the use of the three-touch interaction technique.

time, by requiring that every touch starts initially on the object that one intends to manipulate.

This requirement may make it difficult to manipulate small objects, because multiple fingers have to touch within a small area. It may also be difficult to grab a very thin object. To partly mitigate these problems, we implement 'crossing': a touch does not have to start on an object, but as soon as it comes into contact with one, it is assigned to that object. This makes it possible to 'sweep' a finger across an object and pick it up along the way.

5.3.5 Vertical movement

To be able to use the drawers, and to be able to stack objects, it must be possible to move objects vertically. The most natural way to provide this ability is to let go of the shallow-depth 3D concept and modify the interaction technique to allow for movement in the z-direction as well, providing control over all six degrees of freedom.

In the three-touch technique, two degrees of freedom (movement in the x- and y-direction) are assigned to the first touch, one (rotation about z) is assigned to the second touch, and again two (rotation about x and y) to the third touch. It therefore makes sense to assign the z-movement to the second finger. This can be done in a natural way by mapping the distance between the first and second touch to the z-movement: moving the fingers apart raises the object, and moving them closer together lowers it.

If a perspective projection is used, moving down makes the object appear smaller and moving it up makes it appear larger. This method then becomes very reminiscent of zooming in or out, and in fact uses the same gesture that is already being employed by mainstream devices such as the Apple® iPhone™. An illustration of this interaction can be found in Figure 5.8. The 'figurines' video in appendix A demonstrates the use of this interaction technique in the sandtray prototype.

As long as one or two fingers are used, this method feels very direct and natural. Objects can easily be moved around and into the desired position and orientation.

A slight disadvantage of the technique is that 2D rotation and z-movement are performed with the same finger, and are therefore difficult to perform separately. Often, a rotation will lead to a movement in z and vice versa. However, this is easily noticed and corrected for.

Also, when the first two touch locations are close together, the z-movement becomes very sensitive. A small movement of the fingers will lead to a large movement in the z-direction. This makes z-movement hard to control, but can be compensated for by placing the fingers farther apart initially.

5.3.6 Stickiness

It can be observed that, during some of the interactions described previously, the initial point that was touched on the object's surface remains under one's finger throughout the interaction. It is as if the finger is 'sticky', which is the term we will use for this phenomenon. Note that it is possible that a finger that is initially sticky becomes unstuck, because some *other* finger causes the object to move.

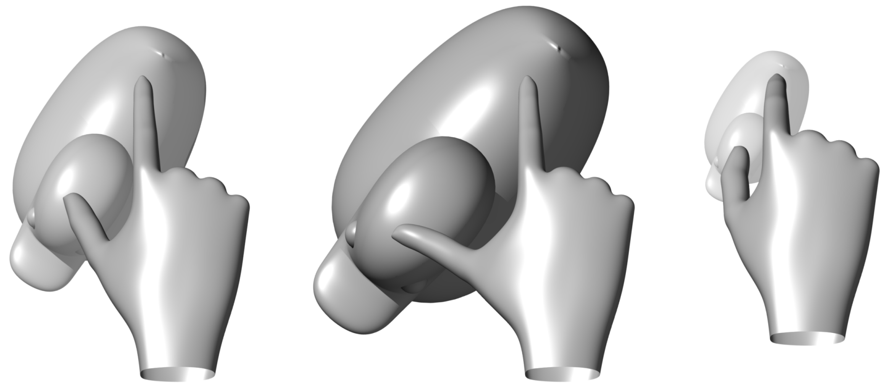


Figure 5.8: Controlling the depth coordinate with the modified three-touch technique. Moving the first two fingers apart raises the object; moving them closer together lowers it. This figure also illustrates the necessity of depth cues; even with an added 'fog' effect, it is still possible to perceive the action as scaling instead of z-movement.

Let us take Hancock's original three-touch technique as an example. Translation with the first finger is sticky. When a rotation with the second finger is performed, this is done about the first finger, so the first finger remains sticky under this interaction.

The second finger, however, is not sticky: it can be moved towards and away from the first finger without resulting in any motion of the object, so the point of contact between the second finger and the object changes. When we use the second finger for depth movement, however (Section 5.3.5), the object's image grows along with the distance between the first two fingers. In that case, the second finger does become sticky.

When a rotation is performed with the third finger (which is not sticky at all), the object rotates about its centre. This means that the initial points of contact of the first and second fingers rotate away from the finger itself, breaking the stickiness of the first two fingers.

We believe that stickiness is an important property, because it strengthens the sense of physical contact between the finger and the object. Thereby, it makes one feel more 'in touch' with the virtual scene. Depth movement already made the second finger more sticky; the following two sections describe two more attempts to make the three-touch technique more sticky.

Rotation about the touch point

In the described three-touch technique, xy-rotation using the third finger causes the point of contact of the first finger to be rotated away from the first touch, breaking the 'stickiness' metaphor. To solve this problem, we can make the

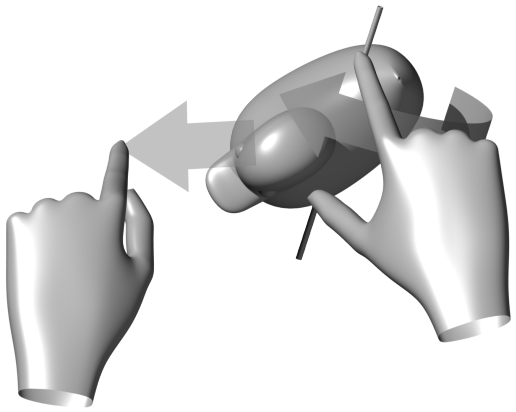


Figure 5.9: The three-touch technique using rotation about a single axis, that passes through the first two touch points. Note how the rotation is constrained to this axis, even though the third finger does not move perpendicular to that axis.

object rotate about the point of contact of the first finger.[1] Obviously, the point of contact itself will then in the same location – under the first finger – during such a rotation, making the first finger sticky.

However, this leads to another problem, when the object is rotated in such a way that its centre gets close to the first contact point. In this case, the 2D RNT (controlled by the second finger) becomes unstable, since the direction of the short vector from the centre to the contact point is ill-defined.[2] This causes the object to randomly jitter about the z-axis. The problem was solved by creating a 'dead zone' around the centre of the object. If the first touch is inside the dead zone, only translation and no rotation is performed. Unfortunately this solution is less than elegant.

Single-axis rotation

As we saw previously, when the third finger is used to perform rotation around the first touch point, about both the x- and y-axes, the stickiness of the second finger breaks down as well. If we want to retain it, our only option is to let the third finger perform a rotation about the axis through the contact points of the first two touches, as shown in Figure 5.9. The amount of rotation is then controlled by the change in distance from the third touch point to the line through the first two. Note that, instead of controlling two degrees of freedom, the third finger now only controls one.

This technique was evaluated informally, but found to be harder to use than the original rotation technique, even though the second finger is now sticky. Although no formal study was performed to investigate this further, we can

[1]During such a rotation, the object's centre can be lifted out of the plane to which it should be confined. This is easily solved by projecting it back onto the plane after the other transformations have been performed.

[2]This problem actually occurs also when no second touch is being used and the first touch point is close to the object centre. However, this rarely occurs since in 2D RNT interaction people tend to drag an object by a point away from the centre.

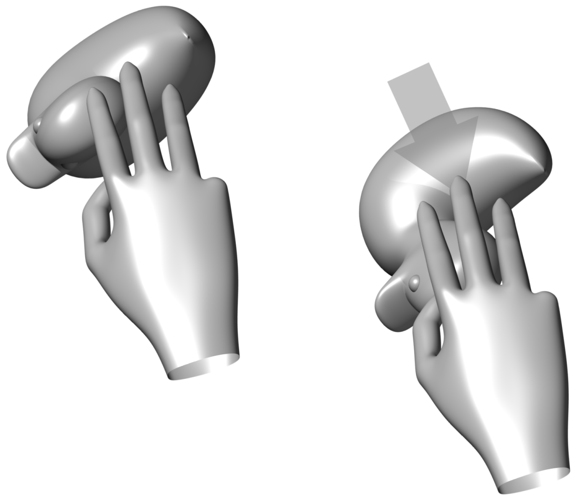


Figure 5.10: When an object is dragged using three fingers, the object will rotate, even though this was probably not the intended meaning of the action.

point out three potential causes.

- The technique feels more limited because instead of two, now only one degree of freedom can be controlled.
- It is difficult to define a rotation axis perpendicular to narrow objects.
- When the first two fingers are close together, the axis of rotation is ill-defined.

In light of these issues, the rotation with the third finger was changed back to the original technique.

5.3.7 Relative rotation

The discussion in this section applies to both the two-touch and the three-touch technique, but for readability we present it in terms of the three-touch technique only.

In the original implementation, the 3D rotation using the third finger depends on the absolute motion of the finger. As a consequence, if an object is dragged across the screen using three fingers that do not move relative to each other, it will still rotate (Figure 5.10).

This problem can be solved in a simple way by interpreting the third finger's movement not in the absolute sense, but relative to the location of the first finger. This does not change anything in the case when one just wants to rotate the object without moving it. In the case when the first two fingers are moved when the third is stationary, the effect will now be a rotation where there was no rotation before; however, in our experiments we have never observed anyone using this unnatural gesture in practice, and it is unclear what its result should be.

5.4 Summary

We began this chapter by detailing how figurines are represented in the physics engine, detailing some sacrifices that were made to realism for the sake of performance, reliability and, especially, usability. Then we discussed how the collection of figurines is presented on the screen for selection, and how they can be selected and placed in the sandtray.

Finally, once a figurine is in the sandtray proper, the largest amount of time will be spent moving figurines around when acting out a story. For that, we use the three-touch technique by Hancock et al., augmented with crossing and vertical movement. Rotation about the touch point was used to made the technique more 'sticky'. For even more stickiness, we also experimented with rotation about an axis through two touch points, but found it inadequate and reverted to the original technique. To avoid undesired rotation when dragging an object with multiple fingers, we made rotation depend on the relative instead of the absolute position of the third finger.

This concludes our discussion of the interactions with figurines and other virtual objects with six degrees of freedom. We now turn to other manipulations that can be performed on figurines and their environment, through the use of virtual tools.

Chapter 6

Virtual tools

It is extremely common in computer programs that multiple different actions can be performed on the same object. For example, in a painting application, one can paint on the canvas, draw circles, and select regions. All these are done by clicking or dragging the mouse on the canvas. Considering our sandtray, moving and rotating figurines is the only action discussed so far, but what if more possibilities are to be added? Evidently, some mechanism is needed to tell the application how to interpret our input. When the same input event can be interpreted in different ways, according to some 'mode' that the system is in, the interface is said to be 'modal'.

Virtual tools provide a way around mode switching, but to understand its significance, we first need to discuss mode switching itself. This is done in Section 6.1. We then describe, in Section 6.2, how virtual tools can provide a way around the inherent problems of mode switching. We then present two examples of virtual tools, in the form of the scale drawer (Section 6.3) and the paint drawer (Section 6.4). The latter has an interesting generalization, which is discussed in Section 6.4.4.

6.1 Mode switching

The most commonly used paradigm used in desktop applications is mode switching. By being in a different mode, the system will interpret our input in a different way. (The definition of what constitutes a mode can be made more precise, but this notion will suffice for the following discussion.) For example, by clicking and thereby activating different buttons on the side of the screen, we can choose whether we want to paint, draw circles or select. In general, we can draw a distinction between the extents of the effects of the mode, the feedback that is given to indicate the current mode, the means by which the mode is kept active, and the means by which the mode is activated.

6.1.1 Mode extent

The effect of a mode can have different extents. Some, but not all, possibilities are the following.

- A ‘global’ mode has an effect throughout the program. On systems where multiple people interact simultaneously, such as tabletops, the use of a mode with global effects is, in general, undesirable, because the input of one person would depend on a mode possibly selected by someone else. Although scenarios can be imagined in which people collectively decide to switch the global mode, this does not account for the common case of multiple people working on different tasks simultaneously.
- A ‘per-user’ mode has effects only for a particular individual. It is unclear how this should be implemented on systems that are unable to distinguish between input of different individuals.
- A ‘per-object’ mode only affects a certain object in the program. This can only be made to work if there are clearly separate objects, and no more than one person will be interacting with an object at the same time. Our sandtray is an excellent example of a situation where such a solution might be appropriate.

This discussion only addresses the effects of the mode switch, sidestepping the questions of how to activate the mode, and how feedback about the current mode is given.

6.1.2 Mode feedback

Although mode switching might be the best we can do given the current desktop hardware, it is not optimal [Ras00]. It burdens people with an additional cognitive load because they have to be constantly aware of what mode the application is in.

Often feedback is provided to alleviate this problem, for example a change of the mouse pointer shape, but this is not always effective, desirable or even possible. For feedback to be effective, it needs to be in the locus of attention, thereby necessarily distracting from the task at hand. Even the mouse pointer is not always in the locus of attention: often, the attention is focused instead on the object *under* the cursor.

6.1.3 Mode maintenance

Sellen et al. [Sel92] draw a useful distinction between ‘user-maintained’ and ‘system-maintained’ modes.

- A system-maintaned mode is one that will remain active without any action on the part of the person interacting. System-maintained modes are the main cause of the general aversion against mode switching, because one has to be constantly aware of the mode that the system is in.
- A sometimes better alternative is to turn the mode into a user-maintained mode: the mode is only active as long as the some physical action is performed, such as holding down the Shift key on the keyboard. Pressing down a mouse button is, in some cases, also interpreted as a user-maintained mode switch. User-maintained modes implicitly provides direct, tactile feedback, generated by the person instead of by the system.

User-maintained modes thereby create much more awareness of the current mode and greatly reduce mode errors as a result. As we will see in Section 6.4, the concept can very well be mapped to tabletop systems.

6.1.4 Mode activation

Every possible input that we can give to the system can serve as a mode switch. It is even possible to interpret this mode-switching input differently, depending on the current mode. This is not as exotic as it may sound: a toggle button is just one example. Depending on the current state the button is in, clicking or tapping it will either activate or deactivate it.

Since any possible input can be used to switch modes, the following sections outlines some of the more commonly used possibilities.

Buttons

Pressing a button is one of the most common ways of instructing a system to perform a certain action. Buttons can be used on physical devices, on mouse-based desktop interfaces and also on touch-based devices. Their popularity is explained by their ease of use, their ease of implementation and their wide applicability.

Buttons can be used to switch into a system-maintained mode, such as clicking the brush button to switch to the brush tool in Photoshop, or pressing Esc to switch the modal text editor Vi into navigation mode. However, buttons can also be used for user-maintained modes; the Shift key is an example of this, the mouse button in a drag-and-drop operation is another.

Gestures

There seems to be no concensus in the literature about the definition of the word 'gesture'. Raskin [Ras00] defines a gesture as "a sequence of human actions completed automatically once set in motion." While it is arguably useful to define a term for this concept, being essentially the 'atom' of input, the word 'gesture' might be too specific. Other authors implicitly use the word to mean hand gestures as made during human speech [Kje96], or as paths of a particular shape, for example drawn with a pen on a tablet [Wob07; Aga06; Cle09].

For our purposes, we define the word 'gesture' to mean "a human pose or sequence of human actions with a particular, significant shape." This encompasses both hand gestures and specific pen strokes, but is narrower than Raskin's definition because it excludes actions like button pressing, in which the shape of the action does not play a role.

Some gestures are suitable for switching to system-maintained modes, such as the mouse gestures in the 3D modelling program Blender that activate the translate, rotate and scale tools. Some can be used for user-maintained modes. For example, a pointing gesture could move objects in a virtual reality application as long as the someone is pointing at them.

Although gesture-based interfaces can be very powerful in the hands of an expert, there is the inherent problem of learnability of gestures. The mapping from gestures to actions is often largely arbitrary, and there is no cue to indicate which gestures are understood by the system.

Another problem is that some gestures, especially when based on vision systems, are difficult to detect reliably. This problem is exacerbated when the space in which the gesture takes place is also used for other interactions, as might be the case on a tabletop. In such cases, a mode switch is needed to put the system in 'gesture-recognition mode' [Li05].

Pressure

Since some tabletop systems provide the programs running on them with pressure information, this information can be used to implicitly switch modes, thereby creating a user-maintained mode. Li et al. [Li05] investigated five different ways of mode switching on a tablet PC, to switch between drawing and gesture modes. They showed that pressure-based switching is feasible. However, they also found that different people use different amounts of pressure. Although people will partly adapt to a fixed threshold, for best performance the threshold should be adapted to the particular person. On a tabletop that is incapable of distinguishing different people, this is not possible.

6.2 Virtual tools

Looking at Sections 6.1.2 and 6.1.3, it is clear that user-maintained modes have significant advantages over system-maintained modes in terms of mode errors. However, many ways of switching into a user-maintained mode are not possible on tabletop computers, where the only means of input is touch.

A viable option for mode switching in 'object-based' interfaces is to switch mode based on the location of the object. For example, we could designate specific areas for each different way of modifying the object. The scaling drawer discussed in Section 6.3 is an example of this.

For another option, we take a clue from the physical world. When building a bike shed with various tools, such as a hammer, a screwdriver and a paint brush, it is nearly impossible to make a mode error; one wouldn't even try to drive in a nail using a paintbrush, or turn a screw using a hammer. Now that we have a sandtray with a consistent way of manipulating virtual objects, it is worth investigating whether we can create virtual objects that can be used as 'tools' to act on other objects. This eventually resulted in the painting tool described in Section 6.4.

6.3 Scaling tool

Scaling (resizing) figurines is of the compelling possibilities that a virtual sandtray offers and a physical one does not. The size of an object has important psychological connotations: larger objects are perceived as more important, more powerful or more menacing. It would therefore be good if the virtual sandtray offered a way to change the size of the figurines.

There is a very intuitive way to scale objects, which is the two finger 'pinch' motion. Unfortunately that motion is already mapped to vertical movement (see Section 5.3.5), so we cannot use it to scale objects. Although this mapping could be changed, we felt that vertical movement was a more frequently used action than scaling, warranting its mapping to this easy and intuitive ‘pinch’ gesture.

Several options were considered for the interaction of this tool.

1. Provide figurines of different scales in the first place. However, this would clutter the drawer, and there is no way to change the size after choosing a figurine.

2. A tool that can be picked up and touched to the object that should be scaled. A slider or dial on the tool could then be used to adjust the size. However, when attempting to fill in the details, many problems arise. For example, if we use the three-touch technique for moving the tool, what finger could then be used to adjust the dial? Does the tool have to be in contact with the object throughout the scaling process? If so, this will be difficult to do because the object boundaries change. If not, then the tool would have to ‘remember’ what object was touched last.

3. A pump to inflate or deflate objects, with a nozzle that can be connected to them. The physical metaphor is clear, but it suffers from the same problem as the slider tool: how would one connect the nozzle to an object?

4. A special box in which an object can be placed; then duplicates of different sizes would appear. No special interaction technique would be needed to operate this tool. Moreover, it could also double as a tool for cloning objects. A disadvantage is that the object needs to be taken out of its context and moved to the scale box.

5. Two boxes: one that will slowly make its contents grow, one that makes them shrink. No special interaction techniques would be needed for this option either, but it also requires taking the object out of its place. Another disadvantage is the dependence on time.

Eventually, we decided on a combination of options 2 and 4. A special box is provided in a drawer, pulled in from the right side of the screen, in which objects can be placed that are to be resized. Because multiple objects can be placed inside the box, each with different size limits, a slider control with a maximum and minimum value would be misleading. Instead, we offer a round dial that is attached next to the box. The dial is drawn with a ridged surface to indicate its affordance, and can be spun with a single finger in an intuitive fashion. When the dial is turned to the right, the object(s) in the drawer grow; when it is turned to the left, they shrink (up to certain limits).

The drawer is illustrated in Figure 6.1; a screenshot of the implementation is shown in Figure 6.2. The scaling tool is also shown in the ‘scaling’ video referred to in appendix A.

Because the drawer requires that the figurine be taken out of its context, it may not be the best choice in terms of usability. However, we chose it over some in-place tool, because the painting tool described next (Section 6.4) already provideded the opportunity to study such a tool.

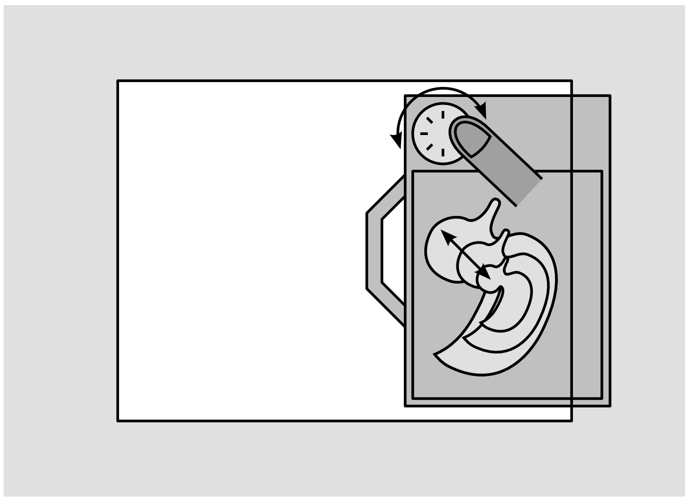


Figure 6.1: Schematic illustration of the scale drawer. When the dial is turned to the right, as shown here, the figurine grows. When turned to the left, it shrinks.

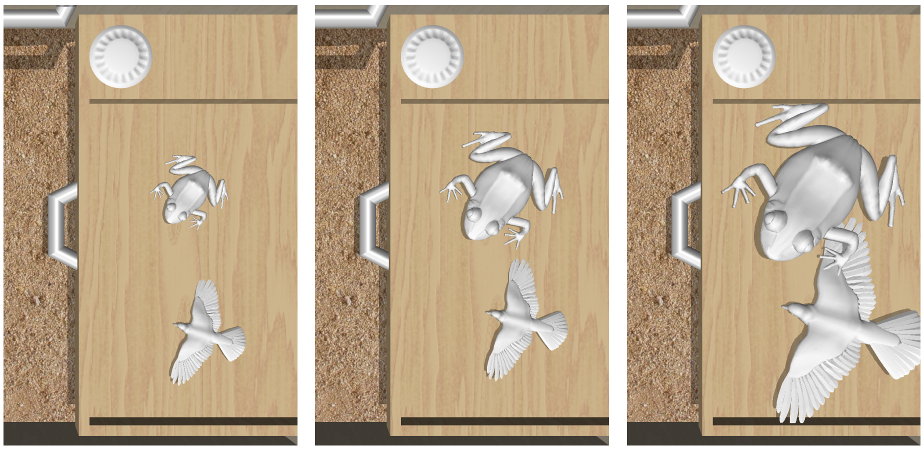


Figure 6.2: A series of screenshots of the scale drawer in action. Spinning the dial grows and shrinks all figurines in the drawer.

6.4 Painting tool

So far, we have largely ignored the sand in the physical sandtray in our design of the virtual sandtray. But we cannot ignore it forever, because the sand does have several important purposes:

1. hills and mountains can be created by piling sand up,
2. valleys can be created by digging,
3. rivers, lakes, ponds and seas can be created by digging until the blue sandtray bottom is revealed,
4. figurines can be made to stand upright by pushing them down into the sand,
5. figurines can be partly or completely buried.

Because we are constrained to tabletop hardware, we will not use physical sand like Wang et al. [Wan02]. Simulation of virtual sand is certainly possible, but fluid dynamics like this are a difficult problem in general. Although software exists to perform such simulations, and the physics library we use does offer fluid dynamics to a certain extent, this does not solve the larger and mostly unexplored problem of interacting with the sand. It was therefore decided early on that sand simulation would, unfortunately, be outside the scope of this project.

Instead, we decided to add the possibility to 'paint' with a texture image on the sandtray floor. Although this does not make it possible to change the depth (items 1 and 2), with a suitable water-like paint it could serve as a substitute for item 3. Item 4 is addressed differently by giving unstable figurines a pedestal (Section 5.1), and item 5 has not currently been implemented.

For the design of the painting interaction, several options were considered.

1. A 'fingerpainting' style interaction, where fingers can be dipped into a paint bucket, and then painted with. Unfortunately, this would come down to a per-finger or per-person mode switch, neither of which are supported by our hardware.
2. A brush that can be moved around using the standard three-touch technique (see Section 5.3.5) and can be dipped into a paint bucket to set its colour. However, such a tool would be difficult to control: it would have the full six degrees of freedom although for the painting it would really need only two. Moreover, if we make it brush-shaped it will suffer from the difficulty of controlling narrow objects (Section 5.3.4).
3. A spray can that can be moved around using the standard technique. This would have the same problems as the paint tool, but since it is less narrow, might be easier to control. However, how would we turn it on and off? Simply turning it on as soon as it is picked up is insufficient, because we need the ability to move the spray can to another location without actually painting anything.

Eventually, we settled on a variation of item 3, more reminiscent of an airbrush. A nozzle with large handles on each side for easy interaction is used

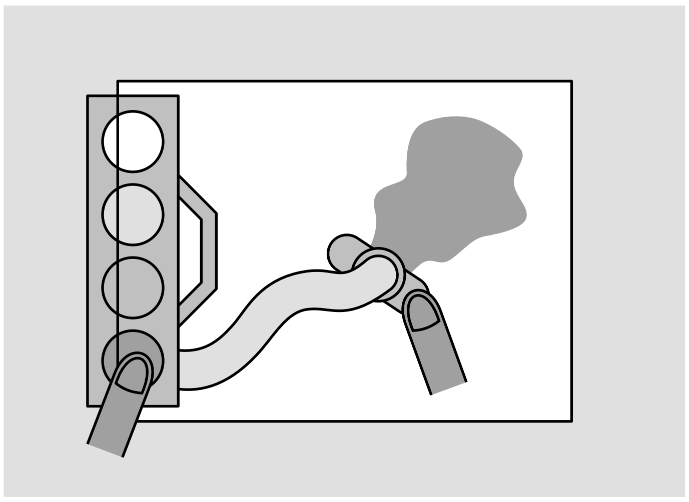


Figure 6.3: The paint drawer, showing four different paint buckets, the hose and the nozzle. Because both the nozzle and a bucket are being touched simultaneously, paint will flow from the nozzle.

to paint. Paint buckets in a drawer, pulled in from the side of the screen, are used to select the type of paint. A hose connects the nozzle to the chosen paint bucket. See Figure 6.3 for an illustration of the usage of the paint drawer, and also the 'painting' video referred to in appendix A.

Only when the nozzle is held, *and* a paint bucket is touched, will the nozzle start to paint on the sandtray floor. The bucket will usually be selected using the nondominant hand.[1] If the bucket is released, it is no longer clear what paint should be used, so the painting stops. If the nozzle is released, it is no longer clear where the painting should take place, so the painting stops.

Note that this is a user-maintained mode switch (Section 6.1.1), and thus does not suffer from the problems of system-maintained modes. Moreover, it is an object-specific mode, and therefore will not interfere with possible other actions going on outside the painting system. Though no formal study was performed, our initial impressions of this form of bimanual interaction are quite positive.

6.4.1 Drawer and paint buckets

The paint drawer is made as narrow as possible in order not to obscure too much of the underlying floor, which could be subject to painting. The paint buckets are arranged in a row and each of them shows the texture of the paint that is contained in it.

6.4.2 The hose

When the paint drawer is pulled into view, the hose will be folded underneath it, and the nozzle will stick out on the side. When the nozzle is picked up from

[1]For this reason, it would be better to switch the location of the paint drawer to the right side if a left-handed person is using the system.

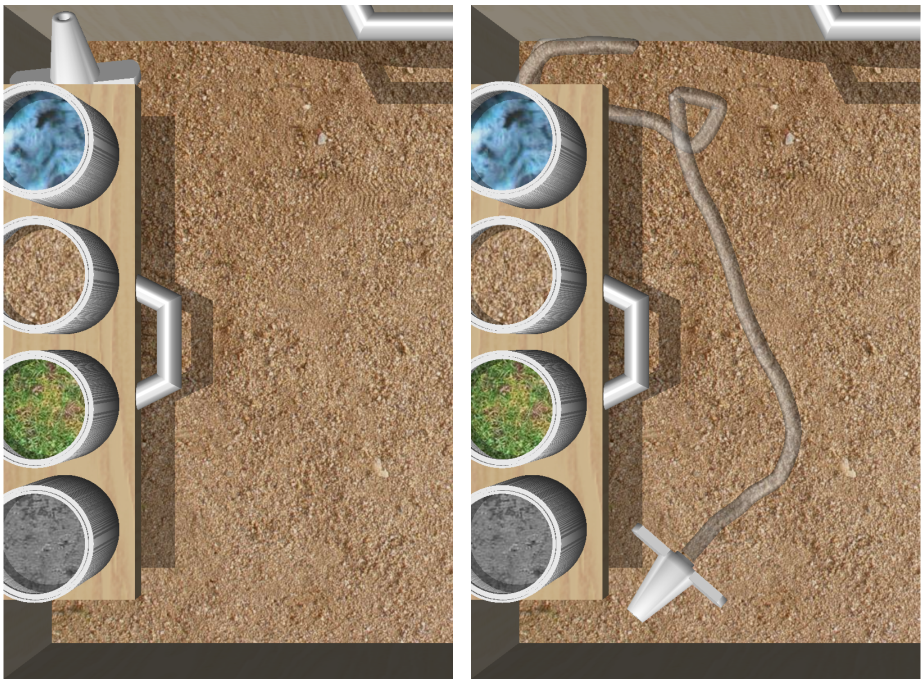


Figure 6.4: Initially, the painting hose is folded underneath the drawer. As soon as the nozzle is touched, the hose is released, and will not be retracted again until the drawer is closed.

this position, the hose is released and follows the nozzle around. The other end of the hose is connected out of sight to the bottom of the drawer. When a paint bucket is touched, this end will move to the bottom of the bucket. When the drawer is closed again, the virtual springs that held the hose in place are reactivated and the hose is pulled back into its starting position. The initial situation and a subsequent state are shown in Figure 6.4.

In the physics engine, the hose is represented as a series of capsule shapes, connected to each other by spherical joints (Figure 6.5). These joints are configured with springs to give the hose a preference for remaining straight. To prevent the hose from accidentally pulling figurines around during painting, the physics engine was instructed to have figurines affect the hose, but not vice versa.

Normally, the hose is shown as white and slightly transparent. When painting, it will however fill up with the paint texture used, which is animated to flow through the hose as long as the painting continues.

6.4.3 The nozzle

The nozzle can be moved around in the standard way, with one exception: upon picking it up, the nozzle will automatically rotate (about the touch point) to point down towards the sandtray bottom. This is the orientation that the nozzle will always be used in, so it is simply a shortcut, and does not require any special or new interaction techniques.

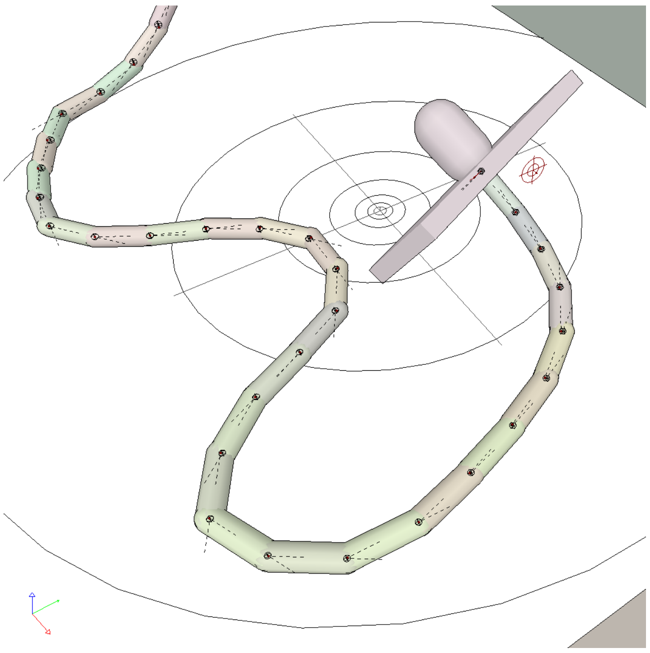


Figure 6.5: The painting hose as represented in the physics engine. It consists of a string of capsule shapes, connected by spherical joints.

When the nozzle is painting, it will draw the chosen texture on the sandtray floor with a circular shape. The radius of the circle depends on the proximity of the nozzle to the floor. If the nozzle is held high, far away from the floor, it paints a big circle, suitable for filling in large regions at once. If the nozzle is held low, close to the floor, it will paint a small circle, suitable for adding details. This is illustrated in Figure 6.6.

6.4.4 Mouse emulation

The hose was designed to solve the problem of painting on a surface. This problem is also solved, albeit in a different way, by the paint brush tool in drawing applications for desktop computers. With such a tool, one can paint on the canvas using the mouse.

It is interesting to note that the this problem yielded very similar solutions in the desktop and the tabletop world. On the desktop, the brush is positioned by moving the mouse and thereby the mouse pointer to the desired location; on the tabletop, this is done by dragging the nozzle with one or more fingers. On the desktop, the brush is activated by pressing the mouse button; on the tabletop, this is done by holding down a finger on a paint bucket.

This similarity can in fact be generalized: in the painting hose, we have nothing short of an emulation of a mouse on a tabletop. One finger controls the cursor, whereas another is used to press buttons. This gives us the 'hover' ability that mice have, but is lacking on tabletops. Moreover, it gives us multiple buttons, to which different meanings can be assigned, where normally all touches are equal. Without any modifications, it is also possible to provide multiple

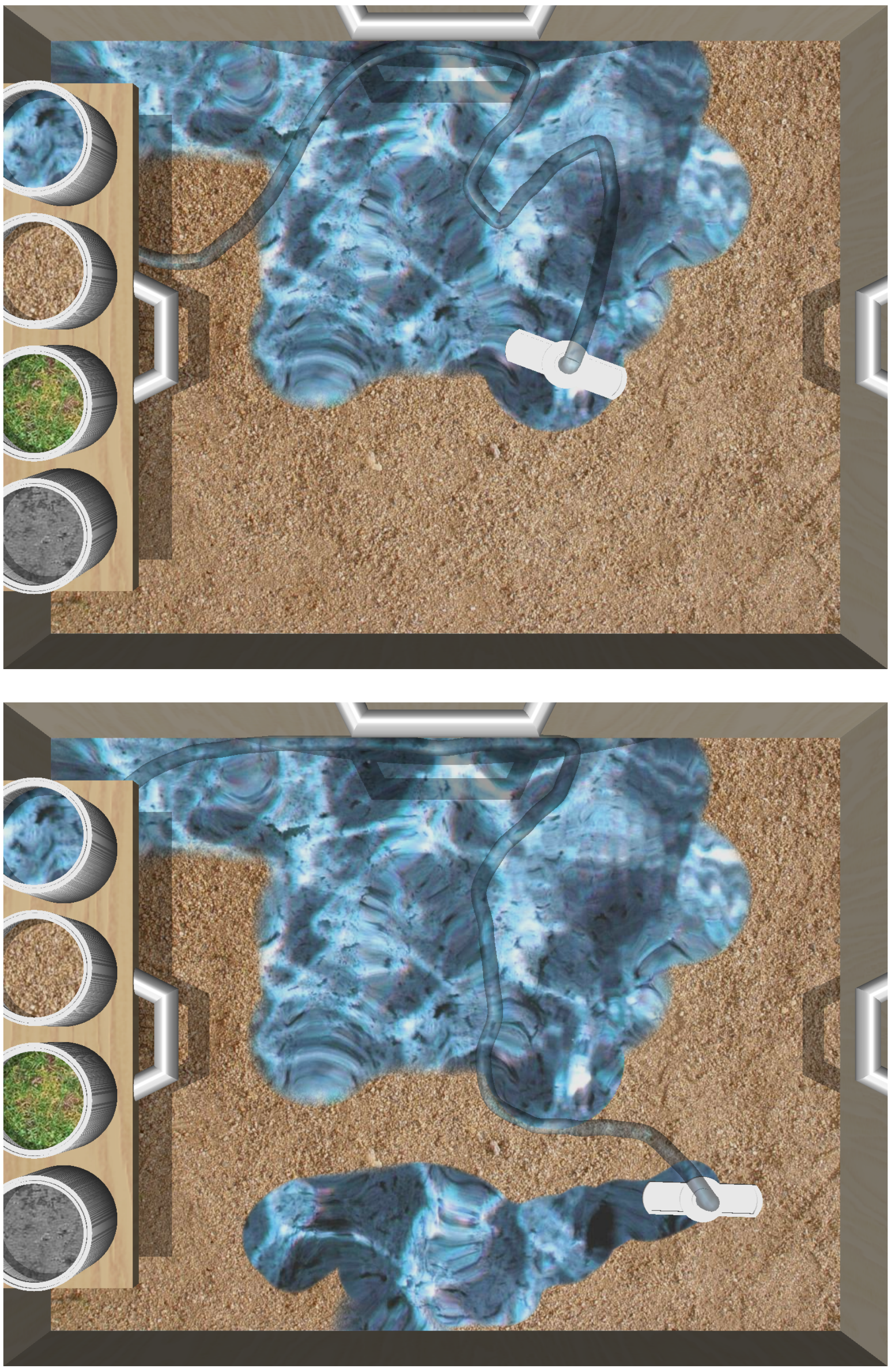


Figure 6.6: Screenshots of the painting tool in action. The lower the nozzle is held, the smaller the radius of the painted area becomes.

mice on one system. How well this mouse emulation works, and comparing it to alternatives, would be an interesting topic of further research.

One might argue that this type of interaction does not play to the strengths of tabletop systems. However, it can possibly be used to make desktop applications run without modification on tabletop systems. Moreover, as the painting hose shows, even when not trying to emulate desktop systems, there is sometimes a need for the action of 'pointing without clicking'.

6.5 Summary

We defined the notion of 'mode switch' and made the distinction between system-maintained and user-maintained modes. We saw why system-maintained modes break down in a multi-user setting, and showed that virtual tools are a form of user-maintained modes that provide a way around this problem. We then presented two virtual tools, one that switches an object's mode based on its location, and another that switches mode based on a virtual object being 'held' under one's finger. We saw that the latter, the painting tool, is actually a possible representation of a desktop computer mouse on a tabletop computer.

The notion of virtual tools is an interesting one, and this chapter only revealed the tip of the iceberg. More research is needed to categorize and structure the possibilities and to determine what works and what does not. In the meantime, we will outline some implementation details of the sandtray prototype in the following chapter.

Chapter 7

Implementation

This chapter describes the implementation of the virtual sandtray in software. The aim is to give the reader an impression of how the implementation works, not to provide in-depth developer's documentation. Many details are therefore omitted.

The chapter begins with a general overview of the system in Section 7.1. It then describes in Section 7.2 how touch input is read and processed. Section 7.3 describes the workings of the rendering pipeline and some implementation choices made concerning rendering. Section 7.4 describes the physics engine and how it is being used. Finally, Section 7.5 details how the figurine meshes are preprocessed, stored and used.

7.1 General

The sandtray program was built in Java [Suna] on top of a framework by Mark Hancock, which provides a uniform interface to different touch hardware platforms, and implements the various multi-touch techniques.

This framework heavily depends on the Java3D library [Sunb] for vector and matrix operations and for texture handling. Although this library also contains classes for construction of a scene graph, these are not used.

7.2 Touch input

In order for the program to be easily portable to different hardware platforms, the framework abstracts away the details of the platform, providing touch data to the program in a uniform way. Essentially, this touch data consists of a set of current touch points for each moment in time. The touch points themselves obviously have x- and y-coordinates, but actually provide more data fields to represent the capabilities of each device. For example, on the DiamondTouch each touch point also carries the identifier of the person who is touching, and on the SMART Table the size of the touch point is provided.

Interfacing with the different pieces of touch hardware (and some other devices) is done through their various APIs. We describe briefly how input is handled for each device.

The DiamondTouch [Die01] provides a Java API that sends out 'enriched' mouse events. It is wrapped by the framework, which listens to these events and updates the set of current touch points accordingly.

The SMART Board [SMAa] has a C++ API only. This is wrapped by C++ code that calls Java event handlers through JNI, the Java Native Interface.

The SMART Table [SMAb] provides its touch data through a .NET API, but also outputs an XML stream over a TCP/IP connection. Although this feature, and the format of the stream, is undocumented, the XML is quite simple and the relevant fields could easily be found. Again, the incoming sets of touch points are converted into our universal format for consumption by the application.

Finally, for easy testing on a desktop computer, a simple class is provided that simulates touch input using a three-button mouse. The left button (assuming a right-handed setting) simulates a single touch, placed when and where the button is pressed, following the cursor movement, and removed when the button is released. The middle mouse button does the same, but touch points placed in this way remain on the surface even after the button is released. All touch points placed in this way are removed by clicking the right mouse button. Note that it is not possible to move these touch points after the middle button is released. Although by no means the most flexible or intuitive scheme, it works well enough for testing our application on a desktop machine.

7.3 Rendering

For rendering, the program uses hardware acceleration with OpenGL through the JOGL (Java OpenGL) library [JOG]. This library provides a very thin wrapper around the OpenGL API; essentially, all OpenGL functions are contained in a `GL` class and have the same name and signature as their C counterparts.

The current collection of figurines consists of nearly half a million vertices. Combined with multi-pass rendering for the shadow mapping, this turned out to be a problem: on an NVIDIA GeForce 7900 GS graphics card, the frame rate varied between 10 and 20 frames per second, which is too low for smooth interaction. Two significant optimizations were implemented: vertex buffer objects, and frustum culling. These are described below, followed by a discussion of the technique used to render shadows.

7.3.1 Vertex buffer objects

To speed up rendering, since the meshes do not change, we store these in OpenGL vertex buffer objects (VBOs), defined in the `ARB_vertex_buffer_object` extension [Ope03]. These are stored on the OpenGL server; usually, this means that they are uploaded to memory on the graphics card, saving the bandwidth that would otherwise be used to transfer the mesh data every frame. This gives a significant speedup in rendering.

7.3.2 Frustum culling

Another optimization that can be done is frustum culling. Because of the size of the ring drawer, most of the figurines in the drawer will actually be off the screen. We can therefore save much vertex processing power by not rendering these offscreen figurines in the first place. Determining whether a figurine is visible is done simply by checking whether its bounding ball intersects the view frustum. More about the computation of the bounding ball is said in Section 7.5.4.

7.3.3 Shadow mapping

As mentioned in Section 4.4.3, providing drop shadows is essential to give the viewer a sufficient sense of depth. Different techniques have been developed over the past decades to make objects cast shadows onto other objects. We opted for one of the simplest and oldest techniques, namely shadow mapping. A full discussion of this algorithm is outside the scope of this thesis; see Williams [Wil78] for the original description of the method. Shadow mapping can be insufficient for large and general scenes, but its limitations do not pose a problem in our limited setting.

Shadow mapping is done in multiple passes. The rendering of a frame consists of $2n + 1$ steps, where n is the number of shadow-casting lights:

- First, the scene is rendered from the point of view of each of the n lights. We use OpenGL framebuffer objects (FBOs), as defined in the extension EXT_framebuffer_object [Ope08], to draw the depth buffer into an offscreen texture for use as a shadow map.
- Then, the scene is rendered once from the point of view of the camera, without any of the shadow-casting lights enabled. This produces the image of the scene as if everything were in shadow.
- Finally, the scene is rendered, with additive blending, from the point of view of the camera for each of the n lights. Only the current light is enabled in each of these passes, but it is modulated by the shadow map lookup. This produces light pixels wherever the pixel is visible from the point of view of the light, and dark pixels elsewhere.

It is worthwhile to note that the rendering of the final scene can be condensed from $n + 1$ passes into a single pass by offloading the lighting calculation to a fragment shader. Especially in the case of many lights, this would give a significant speed boost. We did not implement this because in our case there is only one shadow-casting light, and moreover, the framerate was already sufficiently high for smooth interaction.

A limitation of this approach arises when transparent surfaces are used. Because of depth buffering, objects behind a transparent surface (from the point of view of the camera, *not* from the light) would receive no light, because the light passes would fail the depth test. In the current version of the sandtray application this poses no serious problem. However, if the need for a solution arises, the shader approach would also eliminate this problem.

7.4 Physics

For the rigid-body physics simulation, we use NVIDIA's (formerly Ageia's) PhysX engine [NVI]. Because this engine was specifically developed for use in games, it is optimized for high performance, and utilizes the graphics processing unit to perform parts of the computations.

To use PhysX, which has a C++ API, from Java, we use a wrapper called JPhysX [Kra]. This is a very thin wrapper, and most of it is even automatically generated by SWIG, the Simplified Wrapper and Interface Generator [SWI]. As a result, the full power and potential of the PhysX library is exposed to our Java program.

However, to make use of some features of PhysX, such as the cooking API and error handling, we found that JPhysX was not sufficient. Although the source of the wrapper itself is available, the SWIG interface file from which it was constructed is not part of the JPhysX distribution (probably because it includes portions of the proprietary PhysX header files). Without this interface file, modifying the auto-generated JPhysX code would be very cumbersome. We therefore constructed some functionality in C++ and linked these with Java through another set of SWIG-generated JNI wrappers to overcome these limitations.

7.5 Figurines

The aim of this project was not to model a large collection of figurines by hand. Instead, we used a library, namely the 2002 edition of the Viewpoint Premier e-Catalog. This library contains over 14,000 pre-made models of many different types, and at varying levels of detail. Although some of these models are NURBS-based, most consist of triangle meshes. The models are stored in a custom format, but can be exported by the catalogue program to standard formats such as Wavefront OBJ. Even though the catalogue displays coloured and textured thumbnail previews, no colours or textures are actually exported, and whether this data is actually present in the library is still unclear.

Based on input from sandtray therapists, we selected 158 models from the catalogue, taking care not to pick meshes with too little detail because they would look ugly, or with too much detail because they would slow down the system. The largest model contains around 17,000 vertices; the average is approximately 3,000. A list of descriptive titles of all models is given in appendix B.

7.5.1 OBJ file loading

The meshes were exported to OBJ files. Because OBJ is a fairly simple plain-text format, it was easy to write a parser that imported these files into Java data structures.

However, even after optimizations, this parsing turned out to be far too slow, taking minutes of loading time at each launch of the program. We therefore built a preprocessor that loads the OBJ files one by one, writes the data into a Java object containing a small number of large arrays, and serializes this object to a file using the `Serializable` interface. This reduced the startup time of the sandtray program to mere seconds. This speed is mostly limited by hard drive bandwidth, because a second launch of the program (when the files are still in an operating system or hard drive cache) is significantly faster.

7.5.2 Mesh cooking

Even though NVIDIA's physics engine, PhysX, is quite powerful, it has some limitations. It supports two types of meshes: convex hulls and arbitrary triangle meshes. However, we found that collision detection using arbitrary triangle meshes can be unreliable and in some cases even crashes the engine completely. We therefore chose to use convex hulls only. PhysX comes with a separate API, the so-called 'cooking' API, to convert a set of vertices into a convex hull, represented in a format that the engine can directly use.

During the preprocessing step already mentioned in Section 7.5.1, we also cook the meshes and store the cooked data in a file. This prevents having to re-cook the meshes at each startup of the sandtray. It is this part of the preprocessing that gave the preprocessor program the name Kitchen.

7.5.3 Mesh simplification

Another (badly documented) limitation is that convex meshes can contain at most 256 polygons. Even though the cooking API will happily cook meshes with larger numbers of polygons, this results in runtime crashes when one tries to use the produced data. Because most of our meshes contain more than 256 faces, often even when only the convex hull is considered, it was necessary to simplify the meshes before handing them off to the cooking API.

Simplification of meshes is a difficult problem in general, and much work has been done to investigate this. Many different algorithms exist, each with specific properties that make it suitable or unsuitable for a particular application. Our requirements are as follows:

- The number of faces in the output must be bounded by a prespecified constant. Some algorithms only allow us to specify a certain tolerance, while the actual number of faces produced is unpredictable.
- Either the algorithm must be simple to implement, or a library must be readily available with a suitable license for our use.
- The algorithm must work on many different kinds of shapes, since the shapes of the figurines are highly diverse.
- Preferably, the algorithm should be able to deal with n-gons (faces with more than three vertices), because these can be stored in the OBJ format and might thus be present in the input. However, under the assumption that all n-gons are planar and convex, we can easily triangulate them, so this is not a hard requirement.
- Non-manifold surfaces should be handled, because we cannot be certain that all models from the catalogue will be manifold.
- Preferably, the produced simplified mesh encloses approximately the same volume as the original mesh; that is, it should not only contract or only expand the volume. This will give the best approximation of the original mesh when simulated in the physics engine.
- It should not take more than a minute to simplify a mesh of the size we are using. We do not want the preprocessing for all the meshes to take overly long.

On the other hand, the following properties may be desirable in some applications, but not particularly in ours:

- The ability to produce, in a single run, multiple simplifications at varying levels of detail. We need only one simplified version.
- Preservation of the original mesh topology. Since we are computing the convex hull of the vertices, the triangles and thus the topology produced are of no concern to us.

Several methods from the literature were examined based on their properties as stated by the original authors. Various algorithms by Hoppe et al. [Hop93; Hop96] were found to be too complex and possibly too slow. Turk's retiling algorithm [Tur92] is not suitable for models with sharp corners and edges.

Eventually, the quadric-based method by Garland [Gar97; Gar99] was found to be suitable. This algorithm works by repeatedly contracting edges into a single point, reducing the adjacent triangle faces to zero area, then deleting them. The edges to contract are selected based on a cost function, that is computed as the distance to a quadric surface that locally approximates the original shape of the mesh. The algorithm does not attempt to preserve the mesh topology, which allows for more agressive optimization.

Although a library called QSlim [Gar04] is available that implements this method, it is written in C++. Interfacing between Java and C++ is possible, but quite painful, so the algorithm (except for some optimizations for edge cases – literally) was reimplemented in Java.

7.5.4 Bounding ball and box computation

For fast and simple view frustum culling, as described in Section 7.3.2, it was necessary to know the smallest bounding ball for each mesh. Although the problem is well-defined, algorithms to compute the minimum bounding ball are nontrivial. We therefore used the Miniball library for this, developed by Gärtner [Gär99].

The smallest axis-aligned bounding box of meshes is needed to line them up in the drawer, among other applications. This box is trivial to calculate, but to save some time during startup of the main sandtray program, it is also calculated in the preprocessing phase.

7.6 Summary

In this chapter, we gave the reader an overview of the implementation of the virtual sandtray prototype. We discussed the technologies used, and touched upon some of the larger implementation hurdles. We also briefly described some of the more interesting algorithms that were used in the prototype.

We now turn to the evaluation of the developed prototype.

Chapter 8

Evaluation

Besides the involvement of sandtray therapists in the design process, we also evaluated our prototype in an informal, day-long session together with two sandtray experts. Parts of the session were videotaped for later reference. This chapter presents our qualitative findings. The contents of this chapter raise many interesting issues and can serve as a guide in determining the path for further research.

In Section 8.1, the programme of the evaluation day is outlined. Our findings and observations are presented in Section 8.2, interleaved with some – sometimes tentative – conclusions we can draw from the data.

8.1 Setup

The virtual sandtray prototype was evaluated in a one-day session with two sandtray therapists. Neither of them had any previous experience with tabletop computers, and they had only a very high-level notion of the nature of our work.

In order to get a story that was as unbiased as possible, we asked the therapists to describe their way of working before showing them any of our work. We asked questions whenever something was unclear, but did not give away anything about the design of our prototype, so that the answers would not be affected by our own point of view. We took written notes, but did not videotape this interview.

Next, we showed a program on the SMART Table that demonstrated the use of the one-, two- and three-touch techniques described in Section 5.3. After this, we moved on to the sandtray program and demonstrated its features and how to use it. During all this, the therapists were encouraged to share their thoughts and comments. This part of the session was videotaped.

The therapists were then given the opportunity to play with the application themselves. Again, they were encouraged to think out loud, ask questions and comment on what they were doing or trying to do. This, too, was videotaped for later review.

After a break, we simulated a sandtray therapy session, in which the author played the role of patient. The purpose of this was to see how useful the prototype would be in a real therapy session. This was videotaped as well.

The day ended with a free-form discussion about the experiences and impressions, and a brainstorm about possibilities for future extensions and improvements.

8.2 Findings

This section describes the outcome of the evaluation session with the two sandtray therapists. It is organized by topic. Section 8.2.1 gives some general remarks, Section 8.2.2 presents findings related to the displaying of the virtual sandtray on the screen, Section 8.2.3 comments on the selection of figurines, Section 8.2.4 evaluates the interaction techniques, and finally Section 8.2.5 discusses some possible future extensions that were suggested during the evaluation session.

8.2.1 General

Some words used by the therapists to describe the virtual sandtray prototype were "relaxing and pleasurable," "attractive" and "appealing." In its current form, the prototype might perhaps already be usable for therapy. A negative was the lack of sensory feedback: touch, sound and even smell. However, the application was described as being "still quite tactile."

An interesting point raised was that the virtual sandtray does not so much invite to storytelling, but rather to the construction of a static scene. In that light, it might be more related to art therapy, for example the making of a collage. How to make storytelling more inviting, and whether this is even desirable, remain open questions.

Other applications of the sandtray program were discussed, such as adapting it for educational purposes. For example, its realistic physics simulation could be used to teach the laws of (Newtonian) physics. The ability to easily rotate an object and look at it from all angles could be useful when teaching geometry.

8.2.2 Display

The walls of the virtual sandtray, shown as a frame on the screen, were found unnecessary, because the screen is already framed by the table's edges. However, removing the walls might disrupt the idea that the scene is *underneath* instead of *in* the table surface. As a way to hide figurines, it was suggested to remove the scene walls and allow the figurines to move offscreen. Maybe this could be used to remove them permanently; if not, how to get them back remains an open issue.

One of the therapists commented: "When looking down, I'm not as sure what I'm looking at," and "Camels look weird from above," (which, incidentally, they do). This is a clear indication that a purely top-down view might not be the best choice, and that we should look into projections that also show figurines partly from the side.

8.2.3 Figurines

The selection of figurines was found fairly adequate, but not as comprehensive as the collections used in actual therapy. Two categories that were lacking were fences and buildings. Fences are important to deliniate areas and to create barriers. Buildings provide shelter to the people and creatures in the story.

Another class of objects that was missing are arbitrary objects that could be brought in by the therapist or the patient and play a more metaphorical role. A small cardboard box could serve as a house, a stick could be used as a sword, or a pine cone could represent a baby, covered by a handkerchief to represent a blanket. The inability to bring such objects in could be found limiting if many therapy sessions are performed with the same, limited collection of figurines. It was suggested that a possibility should be added to draw or otherwise create one's own figurines, but this would be difficult to implement in an intuitive fashion.

The presentation of figurines in the drawer was a problem, because once a figurine was selected, the lack of structure made it difficult to find related figurines. Although the therapists with whom the prototype was evaluated normally present the figurines in their therapy sessions in an organized way, they did comment that the lack of ordering in the virtual sandtray prototype "stimulates more random aspects of the psyche." Possibly a hybrid approach is in order here, starting with a random presentation, but allowing the patient to easily find related figurines once a few have been selected.

8.2.4 Interaction

We observed repeated attempts to move figurines with multiple fingers. As discussed in Section 5.3.7, this resulted in an unintentional rotation along with the translation of the figurine. This problem was later solved, as described in the aforementioned section.

Due to the physics engine, combined with some noise in the touch input, figurines would sometimes fly off unintentionally. Maybe a sort of speed limit or friction could be added to the physics simulation to make its behaviour more stable.

Creation of objects upon touching their representation in the drawer turned out to be somewhat too sensitive. Often, more than one copy is created unintentionally. A possible solution is to actually remove the figurine from the drawer and not replacing it with a copy until after some time, or until the original has been dragged away over a certain distance.

Sliding of drawers worked well, even when multiple fingers were used to grab the handle, something we did not design for. Rotating and tossing the circular figurine drawer also rarely presented a problem, except when figurines inside the drawer were accidentally touched, resulting in unwanted creation of new figurines. Perhaps a special area around the edge of the drawer could be added, with a rough surface like the dial on the scale drawer, to stimulate grabbing the drawer in a clear area.

The painting hose turned out to work very well, and there was no trouble using its two-handed interaction method. Especially considering the possible generalisation to an emulated desktop mouse, this is a promising finding.

On the other hand, the scaling drawer was not used often. We theorize that this is because it does not allow for in-place scaling, but requires the figurine to be removed from its environment, creating a psychological barrier to its use in a story. Using a technique similar to the painting hose might be a better option here.

Vertical movement

Several problems were noted in relation to vertical movement of figurines. First, with the current top-down projection, it is not clear that the object is actually moving up or down, instead of simply changing size. This was strongly reflected in the terminology used while discussing this action: even though people know that the object is actually moving up and down in the scene, they often still talk about “making it bigger” and “making it smaller”. This might partly be blamed on the location of the object’s shadow straight underneath it, which often causes the shadow to be partly or completely obscured. A second shadow, cast from the side, might solve the problem partly; a projection that is not strictly top-down could also help.

A second problem is that the ‘sticky fingers’ paradigm implicitly makes lifting an object very sensitive. Especially when the two fingers start close together, a small movement of the fingers will result in a large vertical motion. Doubling the distance between the fingers will move the object twice as close to the virtual camera, which is quite a large distance. Perhaps it is better to let go of the stickiness of one of the fingers, but a formal user study is probably necessary to objectively determine which is better.

A third problem is that it is possible to move a figurine in such a way that it is invisible. For example, a figurine can be pushed down right through a drawer from above, causing the figurine to become hidden underneath the drawer. This can be very confusing. It might be better to keep a figurine always visible, by forcing it to be always above everything else at its location in the scene. This would cause ‘jumps’ when moving it over something else, break stickiness, and make it impossible to knock objects over with other objects, so it might create more problems than it solves.

We also observed three attempts to lift an object by touching it, then letting go of the table surface and lifting the hand up, as if the figurine were connected to the fingers with strings. Obviously, this action did not have the desired effect, but this was quickly learned and adjusted for.

8.2.5 Possible extensions

The ability to paint not only on the floor, but on the figurines themselves, was indicated as an important possible extension. This would be a better option than providing pre-coloured figurines.

Adding cloth would provide compelling new options that could be relevant for storytelling. In particular, the covering of figurines would have significant psychological connotations of hiding, containment or shelter.

The ability to view the scene from ‘inside’, or from the point of view of one of the figurines, would be very useful in therapy. It would evoke empathy with the character through whose eyes we are looking, and would thereby be a valuable addition not only in therapy, but also in education.

8.3 Summary

A great deal was learned from the evaluation session. Many of the problems we expected with the display and the interaction techniques were indeed confirmed, such as the inability to distinguish vertical motion from resizing and the excessive sensitivity in vertical movement. On the other hand, techniques such as the circular drawer and the painting hose turned out to be very intuitive and easy to use. The overall response to the prototype was positive, and it turned out to be likely that such an application could have practical uses in therapy.

Chapter 9

Conclusion

We designed, developed and evaluated a virtual sandtray application in collaboration with sandtray experts. Much was learned and discovered during this endeavour; the main conclusions are summarized in Section 9.1. On the other hand, each answered question seems to spawn ten new ones that beg to be answered; directions for future work are therefore summarized in Section 9.2.

9.1 Findings

Probably the most important result, or at least the most encouraging, is that a virtual sandtray application similar to our prototype would definitely be useful in therapy. This also demonstrates once more the usefulness of tabletops in general for therapeutic purposes. It seems highly likely that other uses of tabletop computers could also be successful, for example in art therapy.

Apart from this general finding, there are more specific ones, which will be discussed in the following sections.

9.1.1 3D on tabletops

Beyond therapeutic purposes, this work can also serve to inform the design of future 3D applications on tabletop systems. It provides insight into interaction with 3D objects on tabletops and on direct-touch screens in general. Moreover, the prototype demonstrates how a physics simulation can play a role in interaction.

There are some significant, but not unsolvable problems with the strictly top-down projection that is currently used to show the sandtray on the screen. It is hard to recognize objects, and difficult to distinguish vertical movement from resizing. These are fairly general problems, and several solutions are readily available, such as a slightly changing the viewpoint or adding extra shadows.

9.1.2 Manipulation of 3D objects

A new interaction technique was developed that allows for manipulation of 3D objects in the full six degrees of freedom. This technique was applied to move and rotate figurines in the sandtray, and fared well in this practical context. Although there are some rough edges, such as overly sensitive vertical movement and problems manipulating narrow objects, these are minor problems that can be worked out.

The notion of 'sticky fingers' was introduced and subsequently applied in several attempts to make the fingers in the three-touch technique more sticky. Although some of these attempts traded off too much power to maintain stickiness, others, such as the vertical movement, did prove successful.

9.1.3 Modes and tools

The work demonstrates how virtual tools can be used to circumvent global mode switching, which is impossible to use on multi-user systems that do not have the capability of distinguishing between users.

The two particular implementations of virtual tools are a mixed success. The scale drawer works well, but may be too clumsy or too slow to use in practice. The fact that figurines have to be removed from the sandtray might also form a barrier for the tool's frequent use. The paint tool, on the other hand, turned out to work very well. It is easy to use and control, even though (or perhaps because) two hands are needed.

The usability of the paint tool is a promising finding, because the very same method of interaction can be used to emulate a desktop computer mouse, or multiple mice, on a direct touch screen. More generally, it allows for touches, which were previously all interpreted in the same way, to be assigned different functions or actions, but without the need for a global mode switch. This might prove to be a powerful generalization, especially as multi-touch applications grow increasingly complex.

9.2 Future work

We divide the future work into three levels: sandtray-specific features that could be experimented with, more general interaction techniques, and finally, other potential applications of the sandtray prototype and similar programs.

9.2.1 Sandtray features

There are many possible extensions to the current sandtray prototype that would be interesting to explore. The full list has already been presented in the unimplemented features list in Section 4.3.3. We will repeat the most interesting options here.

The addition of cloth and soft bodies would not only give the patient more creative freedom, but also raises many interesting questions about interaction. The current three-touch technique is not directly suited to manipulate deformable objects, but could perhaps be modified to work with them. Another possibility might be that springs (which cause instability in the case of rigid bodies) are a feasible interaction technique here.

The possibility to combine objects by sticking them together allows for significantly more freedom. For example, think of houses being constructed from simple, elementary bricks. Again, the interaction techniques used for this kind of object and scene assembly have not been thoroughly investigated. Similar facts hold for the painting of objects.

Simulated sand is technically possible to implement, and would add much richness to the sandtray simulation. Enabling sufficiently rich interaction with this sand, however, could prove to be a very challenging, but also very interesting issue.

9.2.2 Interaction techniques

It would be very interesting to see how the physics-based interaction technique by Wilson et al. [Wil08] would fare in the sandtray. However, this technique suffers from a certain lack of precise control. In particular it is impossible to pick objects up and move them in the vertical direction.

We therefore propose to implement a hybrid method. When a touch starts on a figurine or other object that can be interacted with directly, it will respond in the way described in this thesis. When a touch starts outside all objects, for example on the sandtray floor, a proxy object in the sense of Wilson et al. could be created.

This technique, however, would make crossing (Section 5.3.4) impossible, thereby making small objects difficult to manipulate. It might therefore be necessary to assign touches to an object if they do not start on the object, but are close to it, something that can be implemented in a fairly straightforward way by drawing the object's edges to the pick buffer using a large line width, or by probing the pick buffer in multiple locations around the touch point.

9.2.3 Applications

It has already been mentioned that this sandtray application has a use in sandtray therapy. However, because it seems to encourage scene construction rather than storytelling, it might be more appropriate to use it as such, and modify it to suit this task better. It would be used differently than a storytelling tool, but have similar purposes for therapy.

The question was raised whether the sandtray could be adapted to be of use in an educational setting. It would be interesting to investigate how this can be done, and for what subjects it would be effective.

Bibliography

[Aga06] Anand Agarawala and Ravin Balakrishnan (2006). "Keepin' it real: pushing the desktop metaphor with physics, piles and the pen." In *CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1283–1292. ACM, New York, NY, USA. ISBN 1-59593-372-7. — Cited on pages 22, 24, 41, and 60.

[Ali] "Alice: An educational software that teaches students computer programming in a 3D environment." Retrieved January 20, 2009, URL http://www.alice.org/. — Cited on page 23.

[Bal99] Ravin Balakrishnan and Gordon Kurtenbach (1999). "Exploring bimanual camera control and object manipulation in 3D graphics interfaces." In *CHI '99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 56–62. ACM, New York, NY, USA. ISBN 0-201-48559-1. — Cited on page 22.

[Ber98] Marina Umaschi Bers, Edith Ackermann, Justine Cassell, Beth Donegan, Joseph Gonzalez-Heydrich, David Ray DeMaso, Carol Strohecker, Sarah Lualdi, Dennis Bromley and Judith Karlin (1998). "Interactive storytelling environments: coping with cardiac illness at Boston's Children's Hospital." In *CHI '99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 603–610. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA. ISBN 0-201-30987-4. — Cited on page 23.

[Bra06] Kay Bradway (2006). "What is sandplay?" In *Journal of Sandplay Therapy*, vol. 15, no. 2, pp. 7–9. — Cited on page 25.

[Cao08] Xiang Cao, Andrew D. Wilson, Ravin Balakrishnan, Ken Hinckley and Scott E. Hudson (Oct. 2008). "Shapetouch: Leveraging contact shape on interactive surfaces." In *TABLETOP 2008: 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*, pp. 129–136. — Cited on page 21.

[Cas99] Justine Cassell and Kimiko Ryokai (May 1999). "StoryMat: A playspace for collaborative storytelling." In *CHI '99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA. — Cited on page 23.

[Cas01] Justine Cassell and Kimiko Ryokai (2001). "Making space for voice: Technologies to support children's fantasy and storytelling." In *Personal and Ubiquitous Computing*, vol. 5, no. 3. — Cited on page 23.

[Cle09] Writser Cleveringa, Maarten van Veen, Arnout de Vries, Arnoud de Jong and Tobias Isenberg (April 2009). "Assisting gesture interaction on multi-touch screens." In Steve Seow, Dennis Wixon, Scott MacKenzie, Giulio Jocucci, Ann Morrison and Andy Wilson (eds.), *Proceedings of the CHI 2009 Workshop on Multitouch and Surface Computing*. ACM Press, New York, NY, USA. — Cited on page 60.

[Dav08] Philip L. Davidson and Jefferson Y. Han (2008). "Extending 2D object arrangement with pressure-sensitive layering cues." In *UIST '08: Proceedings of the 21st annual ACM symposium on User Interface Software and Technology*, pp. 87–90. ACM, New York, NY, USA. ISBN 978-1-59593-975-3. — Cited on page 21.

[Die01] Paul Dietz and Darren Leigh (2001). "DiamondTouch: a multi-user touch technology." In *UIST '01: Proceedings of the 14th annual ACM symposium on User Interface Software and Technology*, pp. 219–226. ACM, New York, NY, USA. ISBN 1-58113-438-X. — Cited on pages 18 and 71.

[For05] Clifton Forlines, Chia Shen, Frédéric Vernier and Mike Wu (2005). "Under my finger: Human factors in pushing and rotating documents across the table." In *Human-Computer Interaction - INTERACT 2005*, vol. 3585, pp. 994–997. Springer Berlin / Heidelberg. — Cited on page 20.

[Frö00] Bernd Fröhlich, Henrik Tramberend, Andrew Beers, Maneesh Agrawala and David Baraff (March 2000). "Physically-based manipulation on the Responsive Workbench." In *IEEE Virtual Reality Conference 2000 (VR 2000)*, pp. 5–12. ISBN 0-7695-0478-7. — Cited on pages 22 and 41.

[Gär99] Bernd Gärtner (1999). "Fast and robust smallest enclosing balls." In *Proceedings of the 7th Annual European Symposium on Algorithms (ESA)*, pp. 325–338. Springer-Verlag. — Cited on page 75.

[Gar97] Michael Garland and Paul S. Heckbert (1997). "Surface simplification using quadric error metrics." In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 209–216. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA. ISBN 0-89791-896-7. — Cited on page 75.

[Gar99] Michael Garland (1999). *Quadric-based polygonal surface simplification.* Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA. Chair-Paul Heckbert. — Cited on page 75.

[Gar04] Michael Garland (2004). "QSlim Simplification Software." Retrieved March 4, 2009, URL http://mgarland.org/software/qslim.html. — Cited on page 75.

[Gro07] Tovi Grossman and Daniel Wigdor (Oct. 2007). "Going deeper: a taxonomy of 3D on the tabletop." In *TABLETOP '07: Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pp. 137–144. — Cited on page 16.

[Han05] Jefferson Y. Han (2005). "Low-cost multi-touch sensing through frustrated total internal reflection." In *UIST '05: Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, pp. 115–118. ACM, New York, NY, USA. ISBN 1-59593-271-2. — Cited on pages 19, 20, 24, and 32.

[Han06] Mark S. Hancock, Sheelagh Carpendale, Frederic D. Vernier, Daniel Wigdor and Chia Shen (2006). "Rotation and translation mechanisms for tabletop interaction." In *TABLETOP 2006: First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pp. 79–88. IEEE Computer Society, Los Alamitos, CA, USA. — Cited on pages 21 and 48.

[Han07a] Mark Hancock and Sheelagh Carpendale (2007). "Supporting multiple off-axis viewpoints at a tabletop display." In *TABLETOP '07: Second International Workshop on Horizontal Interactive Human-Computer Systems*, pp. 171–178. IEEE Computer Society, Los Alamitos, CA, USA. — Cited on page 13.

[Han07b] Mark Hancock, Sheelagh Carpendale and Andy Cockburn (2007). "Shallow-depth 3D interaction: design and evaluation of one-, two- and three-touch techniques." In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1147–1156. ACM, New York, NY, USA. ISBN 978-1-59593-593-9. — Cited on pages 13, 22, 24, 48, and 50.

[Hil07] O. Hilliges, L. Terrenghi, S. Boring, D. Kim, H. Richter and A. Butz (Jul. 2007). "Designing for collaborative creative problem solving." In *C&C '07: Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition*. ACM, New York, NY, USA. ISBN 978-1-59593-712-4. — Cited on page 48.

[Hin05] Uta Hinrichs, Sheelagh Carpendale, Stacey D. Scott and Eric Pattison (2005). "Interface currents: Supporting fluent collaboration on tabletop displays." In *Proceedings of the 5th Symposium on Smart Graphics*, pp. 185–197. Springer Verlag. — Cited on page 45.

[Hop93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald and Werner Stuetzle (1993). "Mesh optimization." In *Computer Graphics*, vol. 27, no. Annual Conference Series, pp. 19–26. URL http://citeseer.ist.psu.edu/hoppe93mesh.html. — Cited on page 75.

[Hop96] Hugues Hoppe (1996). "Progressive meshes." In *Computer Graphics*, vol. 30, no. Annual Conference Series, pp. 99–108. URL http://citeseer.ist.psu.edu/hop96progressive.html. — Cited on page 75.

[Ish04] H. Ishii, C. Ratti, B. Piper, Y. Wang, A. Biderman and E. Ben-Joseph (2004). "Bringing clay and sand into digital design – continuous tangible user interfaces." In *BT Technology Journal*, vol. 22, no. 4, pp. 287–299. ISSN 1358-3948 (Print) 1573-1995 (Online). — Cited on page 22.

[Jac94] Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane and M. Preston Mullen, Jr. (1994). “Integrality and separability of input devices.” In *ACM Transactions on Computer-Human Interaction*, vol. 1, no. 1, pp. 3–26. ISSN 1073-0516. — Cited on page 17.

[JOG] “Java bindings for OpenGL.” Retrieved March 4, 2009, URL https://jogl.dev.java.net/. — Cited on page 71.

[Kal] Dora M. Kalff. “Introduction to sandplay therapy.” Retrieved April 11, 2009, URL http://www.sandplay.org/intro_to_sandplay_therapy.htm. — Cited on pages 25 and 26.

[Kje96] R. Kjeldsen and J. Kender (Oct 1996). “Toward the use of gesture in traditional user interfaces.” In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 1996*, pp. 151–156. — Cited on page 60.

[Kra] Yuri Kravchik. “JPhysX.” Retrieved March 4, 2009, URL http://www.jphysx.com/. — Cited on page 73.

[Kru05] Russell Kruger, Sheelagh Carpendale, Stacey D. Scott and Anthony Tang (2005). “Fluid integration of rotation and translation.” In *CHI ’05: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 601–610. ACM, New York, NY, USA. ISBN 1-58113-998-5. — Cited on pages 21, 22, and 49.

[Lee07] Johnny Chung Lee (2007). “Head tracking for desktop VR displays using the Wii remote.” URL http://johnnylee.net/projects/wii/. — Cited on page 38.

[Li05] Yang Li, Ken Hinckley, Zhiwei Guan and James A. Landay (2005). “Experimental analysis of mode switching techniques in pen-based user interfaces.” In *CHI ’05: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 461–470. ACM, New York, NY, USA. ISBN 1-58113-998-5. — Cited on page 61.

[Liu06] Jun Liu, David Pinelle, Samer Sallam, Sriram Subramanian and Carl Gutwin (2006). “TNT: improved rotation and translation on digital tables.” In *GI ’06: Proceedings of Graphics Interface 2006*, pp. 25–32. Canadian Information Processing Society, Toronto, Ontario, Canada. ISBN 1-56881-308-2. — Cited on page 21.

[Mic] Microsoft Corporation. “Microsoft Surface.” Retrieved January 20, 2009, URL http://www.surface.com/. — Cited on page 19.

[NVI] NVIDIA Corporation. “NVIDIA PhysX.” Retrieved January 20, 2009, URL http://www.nvidia.com/object/nvidia_physx.html. — Cited on pages 41 and 72.

[Ope03] (2003). “ARB_vertex_buffer_object.” Retrieved March 4, 2009, URL http://www.opengl.org/registry/specs/ARB/vertex_buffer_object.txt. — Cited on page 71.

[Ope08] (2008). "EXT_frame_buffer_object." Retrieved March 4, 2009, URL http://www.opengl.org/registry/specs/EXT/framebuffer_object.txt. — Cited on page 72.

[Pip06] Anne Marie Piper, Eileen O'Brien, Meredith Ringel Morris and Terry Winograd (2006). "SIDES: a cooperative tabletop computer game for social skills development." In *CSCW '06: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, pp. 1–10. ACM, New York, NY, USA. ISBN 1-59593-249-6. — Cited on page 17.

[Pip08] Anne Marie Piper and James D. Hollan (2008). "Supporting medical conversations between deaf and hearing individuals with tabletop displays." In *CSCW '08: Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, pp. 147–156. ACM, New York, NY, USA. ISBN 978-1-60558-007-4. — Cited on page 17.

[Ras00] Jef Raskin (2000). *The Humane Interface*, chap. Meanings, Modes, Monotony and Myths. Addison-Wesley. ISBN 0-201-37937-6. — Cited on pages 59 and 60.

[Ree06] Adrian Reetz, Carl Gutwin, Tadeusz Stach, Miguel Nacenta and Sriram Subramanian (2006). "Superflick: a natural and efficient technique for long-distance object placement on digital tables." In *GI '06: Proceedings of Graphics Interface 2006*, pp. 163–170. Canadian Information Processing Society, Toronto, Ontario, Canada. ISBN 1-56881-308-2. — Cited on page 21.

[Rya04] Kathy Ryall, Clifton Forlines, Chia Shen and Meredith Ringel Morris (2004). "Exploring the effects of group size and table size on interactions with tabletop shared-display groupware." In *CSCW '04: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, pp. 284–293. ACM, New York, NY, USA. ISBN 1-58113-810-5. — Cited on page 17.

[Sel92] Abigail J. Sellen, Gordon P. Kurtenbach and William A. S. Buxton (1992). "The prevention of mode errors through sensory feedback." In *Human-Computer Interaction*, vol. 7, no. 2, pp. 141–164. ISSN 0737-0024. — Cited on page 59.

[SMAa] SMART Technologies ULC. "SMART – For flat-panel displays." Retrieved December 17, 2008, URL http://www2.smarttech.com/st/en-US/Products/SMART+Boards/Overlays/Flat-Panel+Displays/Default.htm. — Cited on pages 19, 31, and 71.

[SMAb] SMART Technologies ULC. "SMART – SMART Table." Retrieved December 17, 2008, URL http://www2.smarttech.com/st/en-US/Products/SMART+Table/default.htm. — Cited on pages 19, 31, and 71.

[SMA03] SMART Technologies Inc. (2003). "DViT: Digital Vision Touch Technology – White Paper." URL http://smarttech.com/DViT/DViT_white_paper.pdf. — Cited on page 19.

[Str99] Norbert A. Streitz, Jörg Geißler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz and Ralf Steinmetz (1999). "i-LAND: an interactive landscape for creativity and innovation." In *CHI '99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 120–127. ACM, New York, NY, USA. ISBN 0-201-48559-1. — Cited on page 17.

[Suna] Sun Microsystems, Inc. "Developer Resources for Java Technology." Retrieved March 4, 2009, URL http://java.sun.com/. — Cited on page 70.

[Sunb] Sun Microsystems, Inc. "Java SE Desktop Technologies – Java 3D API." Retrieved March 4, 2009, URL http://java.sun.com/javase/technologies/desktop/java3d/. — Cited on page 70.

[SWI] "Simplified Wrapper and Interface Generator." Retrieved March 4, 2009, URL http://www.swig.org/. — Cited on page 73.

[Ter07] Lucia Terrenghi, David Kirk, Abigail Sellen and Shahram Izadi (2007). "Affordances for manipulation of physical versus digital media on interactive surfaces." In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1157–1166. ACM, New York, NY, USA. ISBN 978-1-59593-593-9. — Cited on page 16.

[Tur92] Greg Turk (1992). "Re-tiling polygonal surfaces." In *SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 55–64. ISSN 0097-8930. — Cited on page 75.

[Wal08a] Kristina Walter (2008). "Sand Play Therapy / Sandspieltherapie nach Dora M. Kalff." Retrieved April 11, 2009 (public domain), URL http://commons.wikimedia.org/wiki/File:Sandspiel1.jpg. — Cited on page 26.

[Wal08b] Kristina Walter (2008). "Sandspieltherapie nach Dora M. Kalff / Figuren." Retrieved April 11, 2009 (public domain), URL http://commons.wikimedia.org/wiki/File:Sandspiel_Figuren2.jpg. — Cited on page 27.

[Wan02] Yao Wang, Assaf Biderman, Ben Piper, Carlo Ratti and Hiroshi Ishii (2002). "Sandscape." Retrieved January 20, 2009, URL http://tangible.media.mit.edu/projects/sandscape/. — Cited on pages 22, 36, and 64.

[Wil78] Lance Williams (1978). "Casting curved shadows on curved surfaces." In *SIGGRAPH Comput. Graph.*, vol. 12, no. 3, pp. 270–274. ISSN 0097-8930. — Cited on page 72.

[Wil08] Andrew D. Wilson, Shahram Izadi, Otmar Hilliges, Armando Garcia-Mendoza and David Kirk (2008). "Bringing physics to the surface." In *UIST '08: Proceedings of the 21st annual ACM symposium on User Interface Software and Technology*, pp. 67–76. ACM, New York, NY, USA. ISBN 978-1-59593-975-3. — Cited on pages 22, 23, 24, 36, 41, 48, and 83.

[Wob07] Jacob O. Wobbrock, Andrew D. Wilson and Yang Li (2007). "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes." In *UIST '07: Proceedings of the 20th annual ACM symposium on User Interface Software and Technology*, pp. 159–168. ACM, New York, NY, USA. ISBN 978-1-59593-679-2. — Cited on page 60.

[Wu03] Mike Wu and Ravin Balakrishnan (2003). "Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays." In *UIST '03: Proceedings of the 16th annual ACM symposium on User Interface Software and Technology*, pp. 193–202. ACM, New York, NY, USA. ISBN 1-58113-636-6. — Cited on page 21.

[Zag04] Jose Zagal, Anne Marie Piper and Amy Bruckman (2004). "Kids telling fables through 3D animation." GVU Technical Report 23, Georgia Institute of Technology. URL http://hdl.handle.net/1853/3732. — Cited on page 23.

[Zag06] Jose Zagal, Anne Marie Piper and Amy Bruckman (2006). "Social and technical factors contributing to successful 3D animation authoring by kids." GVU Technical Report 14, Georgia Institute of Technology. URL http://hdl.handle.net/1853/13120. — Cited on page 23.

Appendix A

Videos

Five videos accompany this work, each showing a particular feature of the sandtray prototype. The table shown in the videos is the SMART Table.

figurines.avi demonstrates how figurines can be taken out of the drawer, dragged around with a single touch, moved and rotated with two touches, moved up and down, and rotated in 3D using three touches.

scaling.avi shows a figurine being grown in the scaling drawer, then another figurine being shrunk.

painting.avi shows the usage of the painting drawer. It demonstrates how the nozzle can be moved around and how the hose follows it in a physically accurate manner. It goes on to show how the sandtray floor can be painted, and how the affected area can be grown and shrunk by moving the nozzle up and down.

bowling.avi demonstrates of the physics engine. A bowling ball is taken from the drawer, and a set of bowling pins are added programmatically. The ball is thrown to knock over the pins. Finally, extra pins are added and demonstrate the limitations of the physics engine.

demo.avi demonstrates all features of the sandtray prototype being used in acting out a short prehistoric story.

Playback of the videos requires an AVI player with an XviD or compatible codec. The videos have no sound.

Appendix B

Listing of figurines

airliner
alarm clock
alien
ambulance
apple
armchair
astronaut
ax
baby carriage
barrel
basketball
battle robot
battleship
bear
beaver
bed
bicycle
biplane
bird
blimp
body guard
bomb
book
bowling ball
bowling pin
boy
brontosaurus
buffalo
butterfly
cactus
camel
can
cangaroo
cannon
car
cat
cauldron
cellphone
chair
chest
christmas tree
clown
coffin
computer
cone
cow
crocodile
crown
deer
desk
detective
devil
die
dog
dolphin
domino
Dracula
dragon
drum
dynamite
egg
elephant
elf
eskimo
fighter plane
fire truck
flower
flying saucer
football
fork
frog
galleon
ghost
giraffe
girl
globe
gnome
grim reaper
guitar
hammer
helicopter
horse
horseshoe
hot air balloon
hourglass
key
knife
knight
life preserver
lion
machine gun
magnet
man
missile
motorbike
octopus
palm tree
pan
pegasus
phone
piano
pickup
pirate
pistol
polar bear
policeman
professor
propellor plane
pumpkin
raccoon
race car
revolver
rhinoceros
robot
rocket
rocking horse
rose
saber
sailboat
santa
school bus
shark
shell
shovel
skeleton
snail
snake
snowman
soccer ball
sofa
soldier
space shuttle
spider
spoon
stegosaurus
submarine
Superman
syringe
table
tank
teapot
television
tiger
toboggan
toilet
trash can
tree
triceratops
tricycle
truck
turtle
tyrannosaurus
velociraptor
whale
wheelbarrow
witch
wizard
woman