Expanding tangible tabletop interfaces beyond the display.

Daniel Gallardo Grassot

TESI DOCTORAL UPF / ANY 2014

DIRECTOR DE LA TESI

Dr. Sergi Jordà i Puig,

DEPARTAMENT DE TECNOLOGIES DE LA INFORMACIÓ I COMUNICACIONS

Universitat Pompeu Fabra Barcelona

Al seu suport, Ramón i Montse

Acknowledgments:

A thesis is not only the work of one author; it collects all the author's discussions and experiences gathered during a period of time exchanged with many other people that somehow contributed to the creation of this document.

I would like to thank to all the people, former and actual members, of the Music Technology Group that I have had the pleasure to exchange many experiences and discussions with. To Xavier Serra, the group's director for supporting my research and Sergi Jordà, my thesis director who let me participate in such of amazing projects that I could never imagined. To Microsoft Research Cambridge for their financial support, the access to its astonishing technology and laboratories and the colleagues that welcomed me as a research intern. To Shahram Izadi, the director of the Microsoft's Research group of Sensors and Devices and to everyone that I met there.

During this thesis I had the pleasure to collaborate with many people that helped and contributed to some parts of this final work: Carles F. Julià who shares with me the authorship of all the previous work of this thesis with the projects TDesktop, TurTan, TableGestures and MTCF; David Molyneux who helped me to build the SecondLight unit from scratch; to all my students that have contributed to this thesis: TSI and CDSIM courses for testing the different generated codding frameworks and the students from the computer science degree and master students whose projects I have directed.

Finally this thesis would not have been possible without the unconditional support of my family and Esther for their patience and comprehension during all this work. I would also like to mention Sarah for her English review; all the people of the MAIn lab with that I've been working: Sergi, Carles, Mathieu, Joana, Sebastian, Gianmaria, Marc and all of its collaborators; to Alba, Cristina and Sonia for helping me with all the bureaucratic documents; to Lydia for her understanding and to all of the university staff members.

This work / publication is supported by the Spanish Ministry of Turism and Commerce (MITYC) through the Classical Planet project (Plan Avanza Grant agreement No TSI-070100-2009-407)

This work / publication has been fully granted by Microsoft Research.

Microsoft®
Research

Resum:

L’augment de popularitat de les taules i superfícies interactives està impulsant la recerca i la innovació en una gran varietat d’àrees, incloent-hi maquinari, programari, disseny de la interacció i noves tècniques d’interacció. Totes, amb l’objectiu de promoure noves interfícies dotades d’un llenguatge més ric, potent i natural. Entre totes aquestes modalitats, la interacció combinada a sobre i per damunt de la superfície de la taula mitjançant tangibles i gestos és actualment una àrea molt prometedora. Aquest document tracta d’expandir les taules interactives més enllà de la superfície per mitjà de l’exploració i el desenvolupament d’un sistema o dispositiu enfocat des de tres vessants diferents: maquinari, programari i disseny de la interacció.

Durant l’inici d’aquest document s’estudien i es resumeixen els diferents trets característics de les superfícies interactives tangibles convencionals o 2D i es presenten els treballs previs desenvolupats per l’autor en solucions de programari que acaben resultant en aplicacions que suggereixen l’ús de la tercera dimensió a les superfícies tangibles. Seguidament, es presenta un repàs del maquinari existent en aquest tipus d’interfícies per tal de concebre un dispositiu capaç de detectar gestos i generar visuals per sobre de la superfície, per introduir els canvis realitzats a un dispositiu existent, desenvolupat i cedit per Microsoft Reseach Cambridge. Per tal d’explotar tot el potencial d’aquest nou dispositiu, es desenvolupa un nou sistema de visió per ordinador que estén el seguiment d’objectes i mans en una superfície 2D a la detecció de mans, dits i etiquetes amb sis graus de llibertat per sobre la superfície incloent-hi la interacció tangible i tàctil convencional a la superfície. Finalment, es presenta una eina de programari per a generar aplicacions per al nou sistema i es presenten un seguit d’aplicacions per tal de provar tot el desenvolupament generat al llarg de la tesi que es conclou presentant un seguit de gestos tant a la superfície com per sobre d’aquesta i situant-los en una nova classificació que alhora recull la interacció convencional 2D i la interacció estesa per damunt de la superfície desenvolupada.

Abstract:

The rising popularity of interactive tabletops and surfaces is spawning research and innovation in a wide variety of areas, including hardware and software technologies, interaction design and novel interaction techniques, all of which seek to promote richer, more powerful and more natural interaction modalities. Among these modalities, combined interaction on and above the surface, both with gestures and with tangible objects, is a very promising area. This dissertation is about expanding tangible and tabletops surfaces beyond the display by exploring and developing a system from the three different perspectives: hardware, software, and interaction design.

This dissertation, studies and summarizes the distinctive affordances of conventional 2D tabletop devices, with a vast literature review and some additional use cases developed by the author for supporting these findings, and subsequently explores the novel and not yet unveiled potential affordances of 3D-augmented tabletops. It overviews the existing hardware solutions for conceiving such a device, and applies the needed hardware modifications to an existing prototype developed and rendered to us by Microsoft Research Cambridge. For accomplishing the interaction purposes, it is developed a vision system for 3D interaction that extends conventional 2D tabletop tracking for the tracking of hand gestures, 6DoF markers and on-surface finger interaction. It finishes by conceiving a complete software framework solution, for the development and implementation of such type of applications that can benefit from these novel 3D interaction techniques, and implements and test several software prototypes as proof of concepts, using this framework. With these findings, it concludes presenting continuous tangible interaction gestures and proposing a novel classification for 3D tangible and tabletop gestures.

Table of Contents

Chapter 1 MOTIVATION

In recent years we have seen a proliferation of tangible tabletop interfaces. Research on this topic started in the early nineties (Fitzmaurice, Ishii, & Buxton, 1995) but it has consolidated in the current millennium, bringing together researchers from various fields such as Human Computer Interaction (HCI) and multi-modal interaction, augmented reality, computer-supported cooperative work (CSCW), information visualization, input and sensing technologies or projector-based display systems. More importantly, these last decades have seen the development of multi-touch and tabletop devices such as the Perceptive Pixel's screens (Davidson & Han, 2006) , Apple's IPhone that was followed by a large proliferation of multi-touch smartphone competitors, Microsoft's PixelSense and the Reactable musical tabletop (Sergi Jordà, Kaltenbrunner, Geiger, & Bencina, 2005), which have become real and popular products. What all commercial tangible and tabletop interfaces have in common is the interaction space where all gestures and data input are performed. This interaction space is reduced to a distance of a few millimetres from the screen or surface. In fact, all hand, finger and tangible interactions on conventional tabletop surfaces can be reduced to a binary state interaction: touching or not touching the surface. Besides the multi-touch and blob tracking abilities of these interfaces, the mouse still has a richer input language: pointing, right clicking, left clicking and scrolling. Tangible tabletop interfaces differ from traditional input methods such as the mouse and keyboard in that they provide a direct access to the data by interfacing through a two-way communication channel that directly manipulates the information and preforms everyday gestures or what is known as a "direct manipulation" of the data (Shneiderman, 1993). Yet, much before the implications and potential of these new types of interfaces are fully explored and exploited, much less well-understood and newer technologies keep bringing astonishing new-fangled possibilities.

1.1 Background

Since I finished my undergraduate studies I have been focusing all my research on tangible and tabletop interfaces. My undergraduate thesis was based on the idea of bringing real and tangible actions, as well as communication, that take place on tables (e.g. for working, learning or collaborating purposes) to an environment where, instead of manipulating physical objects, users can manipulate virtual data by performing real-life quotidian gestures and actions. The result of this idea was the TDesktop (Gallardo & Julià, 2007), a full computer desktop environment for a tangible tabletop interface. In addition, some case study applications were included to strengthen the thesis behind it. After TDesktop, instead of continuing the development of an immense tangible desktop system and trying to resolve all the metaphors and virtual data manipulation processes simultaneously, I focused my attention on solving the problem of performing everyday gestures for virtual data manipulation by solving concrete interaction case studies (Gallardo & Jordà, 2010; Gallardo, Julià, & Jordà, 2013, 2008; Julià, Gallardo, & Jordà, 2009). During my stay at the Music and Advance Interaction lab (MAIn) at the Music Technology Group (MTG) of the university Universitat Pompeu Fabra (UPF), where I had access to the Reactable interface, a round shaped and rear-projected tabletop that detects fingers and objects, I had the opportunity to develop several tangible tabletop applications for browsing and sorting large data collections where tangibles had a dynamic role assignation (Gallardo & Jordà, 2010), a full programming language with a reduced subset of tangible instructions that were tested in schools and presented in different exhibitions (Gallardo et al., 2008) as well as other projects that were developed to explore the possibilities of these user interfaces. In parallel, and applying the knowledge and experience acquired from these projects, I built and developed a tangible tabletop framework that summarizes the gathered knowledge on tangible tabletop interaction with the purpose of teaching the interaction systems in an undergraduate UPF course and, recently, in the master program “Curso de postgrado en Diseño de Sistemas Interactivos Musicales” (CDSIM) (Gallardo et al., 2013).

Around the same time, Microsoft produced the first commercial tangible tabletop interface. The MS. Surface was the first popular commercial attempt to break into the popular market. It was designed for showrooms, stores, information points and fairs, and various companies and private individuals started to develop applications for this interface. After reviewing some of the already existent surface apps, as well the apps generated by my students who where encouraged and instructed to use tangibles instead of only multi-touch interaction, except in the case of some brilliant and unconventional applications, I was able to observe the following:

- Multi-touch gestures were the same as those for smartphones: pinch, move and rotate.
- The applications that only required the use of fingers were mainly for browsing framed elements (photos, texts, videos, etc.).
- On the MS. Surface, tangibles were primarily used as data containers, showing or acting as data short-cuts by displaying the data around them when placed on the table.
- Gestures with tangibles were only limited to putting the tangible on the table, move it, in very few cases rotate it and to remove it from the table. It was uncommon to observe finger gestures in combination with the use of tangibles (drag elements to the tangible, interact around it, manipulate the tangible’s virtual data shown data, etc.).
- Games are the most creative applications, whereby tangible objects and finger interactions are used.

Upon analysing the interaction metaphor of these applications, I realized that almost all the gestures on the tangible tabletop surfaces did not correspond or had a limited similarity with real and quotidian everyday actions. This cannot not be attributed to limitations of the technology or a lack of imagination of the designers. Gestures such as tapping the surface, using various tangible elements to manipulate a virtual one or changing the parameters of tangible elements only when they are on the table are more typical of traditional GUI interfaces. Differently, grasping virtual elements, changing the behaviour of tangible objects in the air and starting gestures not on the surface but above it, could change the manner of interacting with tangible tabletop surfaces in order to approach these interfaces to a more natural and fluid gesture vocabulary. Therefore, we need to expand Tangible tabletop interfaces to the third dimension to increase the interface bandwidth and be able to build a new tangible vocabulary.

1.2 AIM OF THIS THESIS

Interacting beyond the display requires software and hardware modifications and new developments for detecting hands, fingers and objects above the surface. At present, accessible or commercial 3D tabletops are not available. These 3D tabletops only exists in the context of research, although in the market it is possible to find 3D tracking technology that could be easily adapted for the purpose of creating new 3D tangible tabletop surfaces.

As no hardware solution is available, we propose to build a system able to track 3D gestures in the air above the surface, which enables continuous interaction and the tracking of objects with pose information. Several studies propose that hand tracking opens a new vocabulary of interaction on horizontal surfaces by detecting and identifying fingers(Marquardt, Kiemer, & Greenberg, 2010), analysing hand shapes (Epps, Lichman, & Wu, 2006; Genest & Gutwin, 2011), via bimanual interaction in the air (De Araújo, Casiez, Jorge, & Hachet, 2012) or continuous finger interaction (Marquardt, Jota, Greenberg, & Jorge, 2011). In order to do this, and for our need to expand tangible interaction beyond the display and thus search for the way of enhancing tangible tabletop surfaces, we propose the implementation of a system able to detect and analyse the following:

- Finger and blob detection on the table surface.
- Hand recognition on and above the surface.
- Finger identification above the surface.
- 6DoF tangible interaction (x,y,z, yaw, pitch, roll)

By implementing these points, users will be able to notice a bandwidth enhancement of the communication between them and machines. Nowadays, tangible and tabletop surfaces only “sense” what is placed on the table’s surface, ignoring the surrounding space that they occupy. Providing mechanisms to sense what is happening in the surrounding environment, tabletop interfaces will be able to anticipate the user’s actions by generating context-aware information [Figure 1], increasing user expressiveness and permitting a more natural and fluid multiuser collaboration (Genest & Gutwin, 2011). Apart from on-the-air hand and finger interaction, tangible interaction also doubles the data extracted from the surface interaction, thus going from three degrees of freedom (x, y and rotation angle) to six (x, y, z, yaw, pitch and roll). This way, the user can interact with a full-object 3D tracking system.

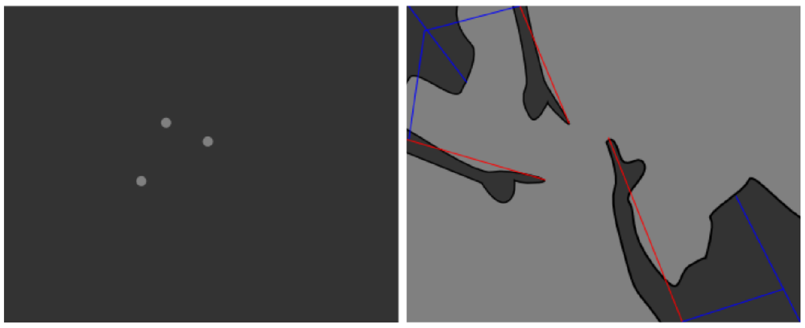


FIGURE 1: CONVENTIONAL TABLETOP INTERFACE TOUCH INFORMATION (LEFT); ELEMENTS THAT CAN BE TRACKED WITH A 3D TABLETOP INTERFACE (RIGHT).

This dissertation addresses the expansion of conventional tangible and tabletop interfaces to 3D tabletop interfaces, and it does so by exploring it from three different areas:

- 3D tabletop hardware.
- 6DoF Software trackers.
- 3D tabletop gestures and framework.

1.2.1 3D TABLETOP HARDWARE

Tracking objects in the 3D space is not an unexplored area, virtual reality (VR) and augmented reality (AR) systems used 3D tracking techniques for full body interaction (M. W. Krueger, Gionfriddo, & Hinrichsen, 1985a) and virtual object positioning (Mark Fiala, 2005) even before traditional tangible tabletops existed.

Nowadays, 3D tracking technology has experienced a large proliferation in the consumer market, having a great impact on gaming stations with 3D tracking devices (more than 100 million Wii units sold up to June 2014[1] and 24 million Kinect devices sold up to 2013[2]). This mass consumption of 3D tracking devices has generated an astonishing increase in the research conducted on this field and the development of home-made devices that include this technology (De Araújo, Casiez, Jorge, & Hachet, 2013; Hilliges, Kim, & Izadi, 2012; Lee, 2008; Schlömer, Poppinga, Henze, & Boll, 2008).

At present, tangible tabletop interfaces use diverse technologies for accomplishing the same purpose: tracking objects and the fingers placed on the surface (Schöning & Brandl, 2008). These technologies can be divided into three areas based on the hardware used: the ones that use electronic sensing such as resistive and capacitive screens, the ones that use specific optical hardware such as infrared emitters and receivers and those that use one or more cameras in combination with a computer vision software.

3D tangible tabletop prototypes take advantage of both technology sides (3D tracking devices and traditional tangible tabletop interfaces) to create new prototypes, for example, augmented capacitive surfaces with a Kinect camera (Huppmann, Luderschmidt, & Haubner, 2012), a multi-touch display under a motion capture system (Marquardt et al., 2011) or in Holodesk (Hilliges et al., 2012) that uses an augmented reality glass with a Kinect camera.

Based on the analysis of the hardware components of 2D tabletop devices and the 3D tabletop prototypes, we propose some further developments of 3D tabletop solutions using different technologies for tracking fingers, hands, objects or all of them, placed both on and above the surface. After designing or modifying a 3D tabletop device that best matched our objectives, we were able to proceed to the next points in this thesis: designing a six DoF position tracking software for the new tabletop and creating a 3D tabletop gesture dictionary and framework definition.

[1] http://www.nintendo.co.jp/ir/en/sales/hard_soft/index.html
[2] http://bgr.com/2013/02/12/microsoft-xbox-360-sales-2013-325481/

1.2.2 6DoF SOFTWARE TRACKERS

Most of the camera-based tangible tabletop interfaces use special markers to track the 2D position and orientation of its tangibles. These markers, known as fiducials, are usually made of an encoded pattern sequence that identifies each marker with a unique ID (Kaltenbrunner & Bencina, 2007; Marquardt et al., 2010; Nishino, 2010).

Virtual reality tags introduced at the end of 90's are the predecessors of tabletop fiducials but, instead of reporting a 2D position, they were designed to encode a 3D vector in order to position virtual 3D objects on the real world (M. Fiala, 2005; Mark Fiala, 2005). These tags were not commonly used until the arrival of tablets and smartphones, when they become popular again as they began to be used in thousands of augmented reality applications (AR)[3].

In this part of the thesis, we will focus on our interest in developing a fiducial marker-based system for tabletop interaction that could report six degrees of freedom of a tangible located above the surface. This fiducial system will be evaluated and compared with similar AR fiducials and traditional tabletop markers, and will be included in a vision tracker that will detect these fiducial markers, hands and fingers in the air as well as on the surface.

1.2.3 3D TABLETOP GESTURES AND FRAMEWORK

The objectives of this part of the thesis are to create an above-the-surface gesture guideline and define a 3D interaction framework based on concepts evolved from my previous work on frameworks for 2D tabletops.

The proposed methodology for the creation of the 3D framework is based on incremental steps starting from basic surface interaction and finishing on on-the-air continuous interaction by following the following steps:

- Traditional 2D tangible tabletop presentation.
- Augmented 2D tangible tabletop.
- On-the-air binary behaviour.

[3] http://invizimals.eu.playstation.com/es_ES/home; https://code.google.com/p/andar/

- On-the-air continuous interaction.

Conventional tabletop tangible interfaces have a binary interaction state: either touching the surface or not touching the surface. By tracking 3D data, we propose to **improve the 2D tangible tabletop** by providing context aware information such as button reaction on hovering fingers, touch disambiguation using the hand tracking data or user identification. Starting from this expanded 2D interface and adding new interaction metaphors extracted from the list of the previous incremental steps, a full 3D tabletop framework will be defined and evaluated by presenting some resulting applications and a 3D gesture classification.

1.3 FUNDING AND SPONSORSHIP OF THIS THESIS

This thesis has been partially funded by Microsoft research, Cambridge (MSR) and Universitat Pompeu Fabra (UPF). Thanks to UPF I was able to access the Reactable interface with which I was able to develop some frameworks and applications for a rear-projected camera-based interface, which were later tested in some undergraduate courses, postgraduate courses and student's projects. MSR provided the technical expertise as well as access to new-fangled 3D tabletop interfaces that I was able test and was given the opportunity to modify one of its units in order to obtained the outcome expected in this thesis.

1.4 THESIS ROADMAP

This thesis is divided into 6 chapters defined as follows:

- **Chapter 2** introduces the traditional tabletop interaction systems, establishes the basis for going beyond the display and presents the author's previous work on tabletop frameworks and the "tabletop applications that suggest 3D interaction" generated with these frameworks.
- **Chapter 3** presents different hardware developments for expanding tangible and tabletop interfaces and describes the prototype used.
- **Chapter 4** presents the developed vision system for tracking hand gestures, 6DoF markers and on-surface finger interaction.

- **Chapter 5** presents the framework developed for 3D interaction by establishing interaction metaphors and gestures on the surface and above. It also introduces proof of concept developments resulting of applying the framework and finalizes proposing a 3D tangible tabletop gesture classification.
- **Chapter 6** summarizes the process of expanding a tabletop to the third dimension and presents some of the possible future work.

Chapter 2 TANGIBLE TABLETOP INTRODUCTION

Before going deeper into 3D tangible and tabletop surfaces, a good reflexion to do is to answer the following questions: From where did the tabletop interaction come? What is tabletop interaction? Why would we need to expand interaction beyond the display? The first and second questions are easy to answer, plenty of information can be found, but the last one requires the definition of 3D tabletop interfaces and our need for the expansion of traditional tabletops that in some cases differs from the original purpose of 3D tabletop interfaces.

2.1 TANGIBLE AND TABLETOP INTERACTION

HCI research is continuously trying to improve the communication between humans and computers. In the last decade, most of the efforts have been focused on changing the conventional way of controlling and communicating with computers (using buttons, keys, mouse, joysticks, etc) by developing more natural and unconventional devices such as data gloves, 3D cameras, speech recognition, wearable sensors such as accelerometers in smartphones, among others. In the 90's, since the ubiquitous presence of personal computers in households, graphical user interfaces (GUI) were under the watchful eye of companies. Often a visually attractive and fluid interface was the key difference between two similar pieces of software.

2.1.1 TRADITIONAL PERSONAL COMPUTERS

The popularization of personal computers was not a fortuitous coincidence. At the beginning, computers where machines produced to solve complex calculations and equations, and their use was mainly restricted to governments and large companies. When computers evolved from a complex calculator to a programmable tool for writing documents, sharing information, administrating databases, controlling other devices and even playing digital games, they started to be seen as the "Swiss knife" of the digital era and were employed by non-expert users as a multipurpose tool denominated Personal Computer (PC).

Graphic user interfaces are the most extended practice on the current PC. They can be found in any commercial operating system (OS) that uses the Desktop metaphor (Borg, 1990; Modugno, Corbett, & Myers, 1995). These systems are strongly linked to Windows, Icons, Menus and Pointers (WIMP) (Nielsen, 1993), whereby different workspaces are delimited by a window: a little screen portion where a specific task could be applied using tools represented by icons or text into menus.

Taking a look at the conventional computer, we can separate all of its visible components into two groups of hardware that are easily detachable, the ones that we use for introducing and manipulating data and the ones that we use for seeing that data. These two large groups, that we will call control and representation, are strongly separated. As seen in the Model-View-controller (MVC) schema in [Figure 2], control devices act as remote control of a virtual tools included at the representation part with the virtual data (Ishii, 2007).

Although traditional personal computers have started to move towards a new tangible paradigm with the consolidation of multi-tactile screens, most of the OS and common applications are still based on a pointing device (either mouse or finger) to interact with windows, menus and icons.

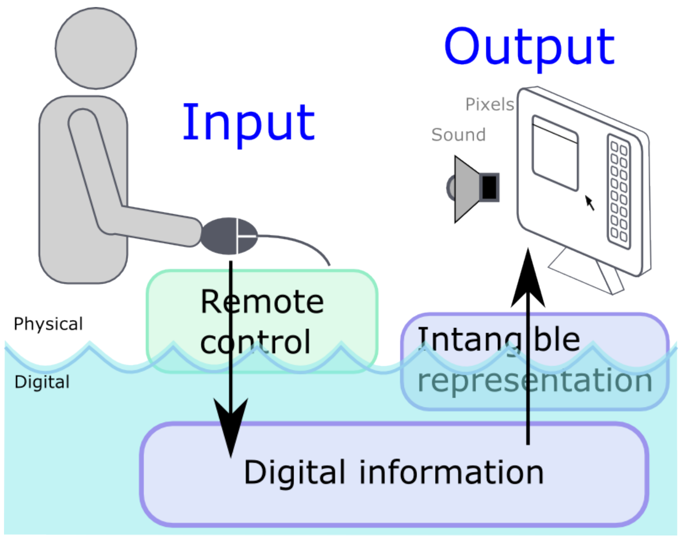


FIGURE 2: MODEL VIEW CONTROLLER SCHEMA OF A GUI USING WIMP METAPHOR.

In [Table 1] a brief list of input and output devices can be found. Excluding force feedback devices and tactile screens, the devices shown are strongly distinct: input devices are only for data input while output devices are for representing and visualizing that data. Although tactile screens and force feedback devices can open a two-way communication channel by allowing input and output in the same device, their use is limited. Differently, multi-tactile screens, with the emergence of smartphones and tablets, have started beyond the use of a single finger as a pointer and introduce finger gestures. However, up to now, there are only a few gestures widely used (pinch, rotate and swipe) (Sergi Jordà, Julià, & Gallardo, 2010).

Common input devices	
Tool or remote control	Action or virtual tool
Mouse	Pointer, pen or drawing tool. In general terms, it is the representation of our hand in the virtual world.
Keyboard	Writing tool with auxiliary position shortcuts buttons (tabulation, arrows, etc.).
Joysticks, pads, pen tablets	Pointing and position tools.
Wheels, force feedback controllers and other mechanical devices	Pointing and position tools but with the peculiarity that some information can be represented on the device by vibrations or force-feedback representation.
Common output devices	
Monitor	Representation of virtual objects and virtual tools.
Speakers	Sound representation.

TABLE 1: PERSONAL COMPUTER COMMON INPUT AND OUTPUT DEVICES

From 1983 to nowadays, command based interfaces, GUI interfaces and the WIMP metaphor have had very little changes: the development of RGB screens, more powerful menus and computers with more memory store capacity as well as different menus and icon stacks. However, the improvement of all the WIMP based OS seem to have stopped in time (at least until the development of mobile and tablet computing). In 1993, Nielsen proposed how next generation interfaces should be in the future (Nielsen, 1993) by comparing 12 dimensions of what interfaces were in 1993 and what they should become. Some of them need to be mentioned below before introducing tabletop interaction:

- Computer role: obey orders (commands) → Interpreting user actions.
- Bandwidth: low (keyboard) to fairly low (mouse) → High (VR)
- Turn-taking: yes (computer awaits for orders) → No (the system keeps going)
- Interface locus: Workstations → Embedded in users environment
- Programming: Imperative → programming-by-demonstration, graphical languages.

Comparing the future dimensions with what recent workstations are (screen, keyboard and mouse), the current interfaces cannot be considered to have been modified in more than 20 years. At present, society is using modern screens, keyboards and mouse devices, but in reality, no significant change has been made.

2.1.2 TANGIBLE INTERACTION

Tangible interaction can be seen as an extension and the deepening of the concept of “direct manipulation,” a term that was first introduced by Ben Shneiderman in 1993 within the context of office applications and the desktop metaphor (Shneiderman, 1993). Although this direct manipulation concept has been closely associated with GUI and the WIMP-based interaction, it is easy to apply the underlying idea to the tangible interaction area. This idea would allow users to “directly manipulate” objects presented to them using actions that imitate or correspond to the ones used in the physical world; assuming that real-world metaphors (on objects and actions) would make it easier for users to learn and use the interface.

The term Tangible User Interface (TUI) was coined in 1997 by Hiroshi Ishii (Ishii & Ullmer, 1997). Ishii envisioned TUIs as interfaces meant to augment the real physical world by coupling digital information to everyday physical objects and environments, consequently allowing users to grasp data with their hands and enabling the representation and control of digital data and operations with physical artefacts. Ishii picked the abacus as the source of inspiration and as the ultimate tangible interaction metaphor because, unlike pocket calculators or computers, in the abacus, input and output components coincide and arithmetical operations are accomplished by the direct manipulation of the results.

From the analysis of the MVC of a TUI in [Figure 3], it is easy to observe that a two-way communication channel exists on the control part whereby the controller becomes part of the data to be manipulated. In general terms, the virtual information contained in TUIs can be represented by physical objects, non-graspable representations (visual feedback, sound, etc.) or both.

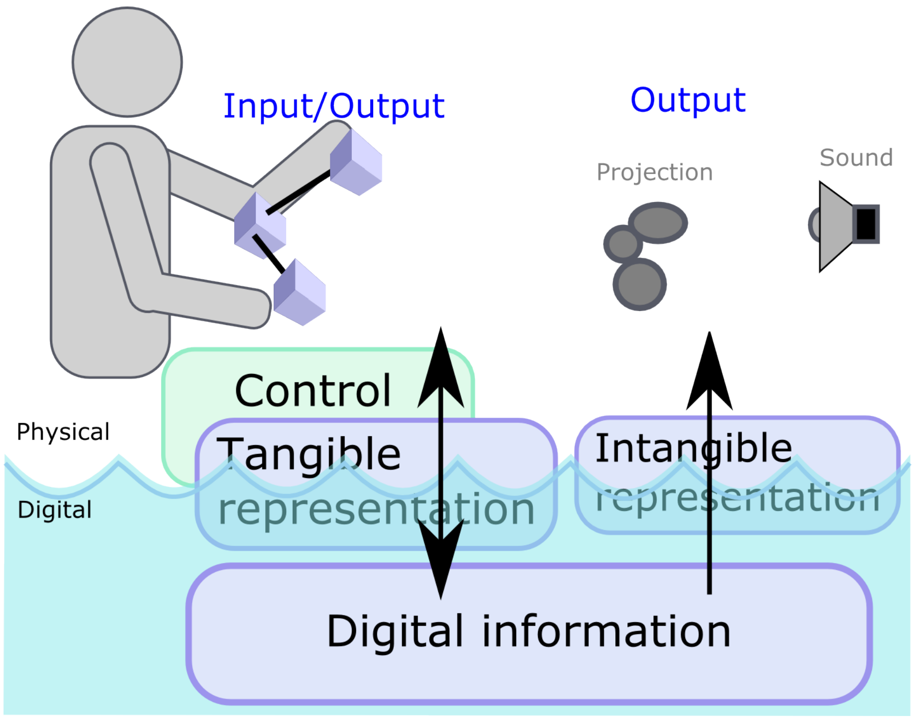
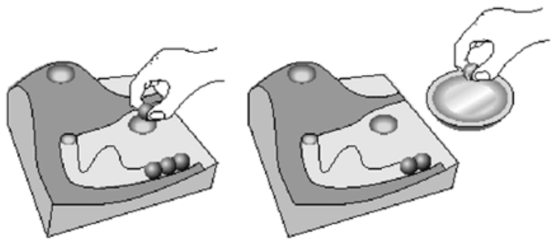


FIGURE 3: MODEL VIEW CONTROLLER SCHEMA OF A TUI. (ISHII, 2007)



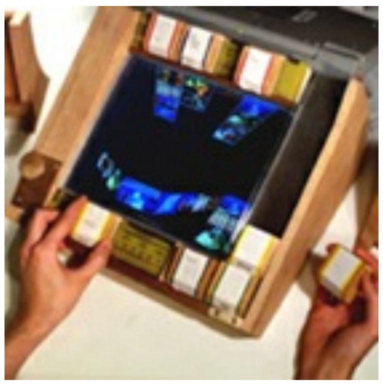


FIGURE 4: BISHOP'S ANSWERING MACHINE (LEFT)[4]; MEDIABLOCKS SCREEN (RIGHT)[5]

The first interface considered as a TUI was a marble answering machine introduced by Durrell Bishop in 1992 (Bishop, 1992). The marble answering machine is a prototype telephone answering machine whereby marbles represent incoming voice messages that the user can grasp and then drop into trays to play the messages or dial the caller automatically [Figure 4, left]. This example shows that computing doesn't have to specifically take place at a desk, but can be integrated into everyday objects. The Marble Answering Machine demonstrates the great potential of making digital information graspable.

Three years after Bishop's machine was presented, "Bricks" (Fitzmaurice et al., 1995) demonstrated how the sensing of multiple physical tangibles on a digital desktop could be used for controlling graphics with booth hands. Consequently, the notion of graspable interface was introduced, which two years later would become "tangible interfaces". After Bricks was presented, Mediablocks (Ullmer, Ishii, & Glas, 1998) was developed, a tangible interface made up of a control screen and other office devices with token trays on which little wood pieces could be placed to interact with the system. The interesting part of Mediablocks was that it presented an interface where tokens were used for both the control of the data and to contain it [Figure 4, right].

With the start of more projects and the evolution of the technology, researchers increasingly started to focus on this area, consolidating TUIs and bringing them to a much more mature state until, in 2007, a dedicated conference known as

[4] Image extracted from (Bishop, 1992).
[5] Image extracted from (Ullmer et al., 1998).

Tangible, Embedded and embodied Interaction (TEI), was created such that all the research groups that worked in this field could share their experiences.

Tangible interfaces differ from computing in their specific use. Tangible interfaces can only do the functions for which it is designed, for example, the marble answering machine can be used only as an answering machine and MusicBottles (Ishii et al., 1999) is for “storing and controlling” media files. Hundreds of these tangible user interfaces exist that, unless they are particularly successful in meeting their purpose, they cannot be used for other different purposes, and thus, exploit the multipurpose machine they are interfacing with.

2.1.3 Tabletop interaction

Tangible tabletop interaction was born as part of the TUI research, which aims to identify a way to perform everyday gestures and object manipulations on a flat area to obtain visual feedback by projecting images or having an integrated screen.

The idea of using a table-based interface instead of a screen or other devices was not a coincidence. Almost all everyday western society activities and communication take place around a horizontal flat surface such as a dining table, a workbench, a sofa table, etc. Tables and horizontal surfaces are a structures where we work, build things, share experiences, eat and much more. Therefore, the social advantages associated with tables directly encourage concepts such as “social interaction and collaboration”(Hornecker & Buur, 2006) or “ludic interaction”(Gaver et al., 2004). Tangible tabletop interfaces recuperate the multipurpose machine philosophy behind personal computers.

The research area on tangible and tabletop interfaces has kept growing since late 90’s and many tabletop prototypes and applications have been built or coded. Some of the most relevant tabletops are explained below to exemplify the potential of these devices.

Active Desk (Fitzmaurice et al., 1995), a rear-projected electronic table, was explicitly developed to be modelled on a traditional drafting table, in size and form factor. The concrete purpose of its design was to support activities such as drawing and designing. This table, among other projects, was used in Bricks as a tangible and tabletop interface and later as a collaborative drawing and gestural communication table. In 1996 it was launched in the market as a product named “VisionMaker Digital Desk” under the denomination “stylus-table”.

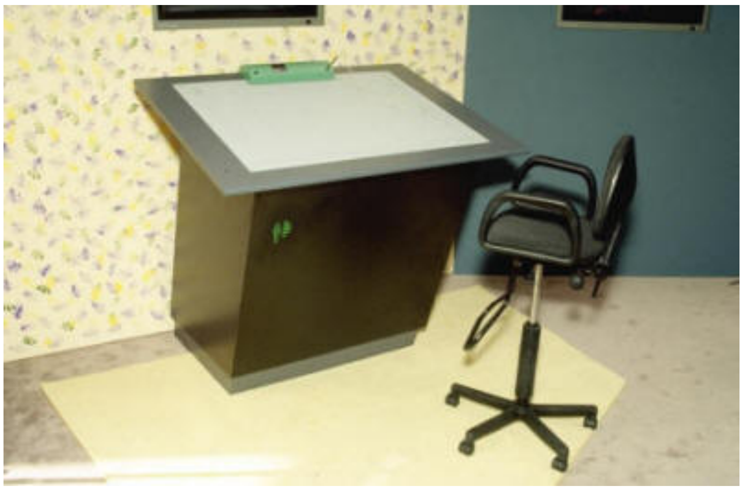


FIGURE 5: FIRST GENERATION ACTIVE DESK[6].

Another early example of a tangible tabletop interface was Urban Planning (URP), a system developed as a town planning aid at the MIT Medialab (Underkoffler & Ishii, 1999). This interface allowed digital simulations of airflow, shadows, reflections, and other data according to the position and orientation of the physical wire-frame models of buildings on the surface. With URP, users could easily see the shape and dimensions of the buildings by the various buildings objects’ intrinsic shape and size and, by standing wherever they wanted around the table, users were able to observe the buildings’ shadows at any time of the day, wind dynamics with other buildings and terrain accidents.

A similar hardware is Sensetable (Patten, Ishii, Hines, & Pangaro, 2001), which also used top-down projected surfaces but used an electromagnetic tracking system instead of camera-based one to detect the position and orientation of tangibles. Each tracked object had embedded electronics that reported a digital state, which could be modified by using dials or tokens. This table became the base for most of the tabletop-related projects developed at the MIT Media Lab,

[6] Image extracted from http://www.billbuxton.com/ActiveDesk.html

such as a business simulation[7] or the Audiopad (Patten, Recht, & Ishii, 2002), a tangible tabletop application for electronic music creation and composition.

In 2001, Mitsubishi Electric Research Laboratories started to explore multi-user and multi-touch tabletop interfaces with Diamond Touch (Dietz & Leigh, 2001). The primary characteristic of this tabletop interface was the user recognition input whereby users were directly detected via their distinct electronic fingerprint. The aim of this table was to facilitate face-to-face collaboration, brainstorming, and decision-making.

What all these tables have in common is the interaction space, in other words, the area where the user can perform gestures and place tangibles. This interaction space, in contrast to that of TUIs [Figure 3] is reduced to the table's surface, which forces the user to interact on a 2D planar surface [Figure 6].

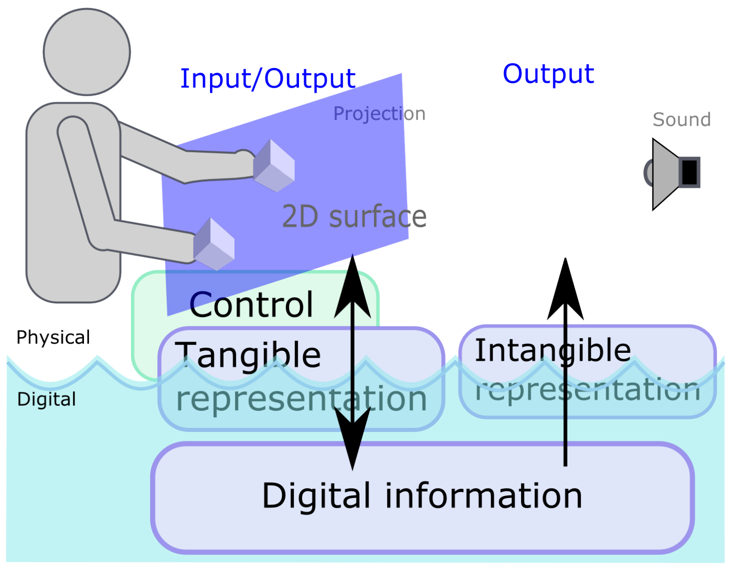


FIGURE 6: MODEL VIEW CONTROLLER SCHEMA OF A TABLETOP

In comparison to the two aforementioned MVC schemas (GUI and TUI), Tabletops lose the z component for enriching its gestural vocabulary and feedback (visual or tactile) beyond the screen. Apart from the tangibles and their physical meaning, the tabletop's surface tends to be a neutral surface that can

[7] http://tangible.media.mit.edu/project/sensetable/

adopt the role of any application without the need of designing and implementing a specific tangible interface for each application.

Some of the previous work on tabletop frameworks developed prior to this resulted in the production of applications that could be considered 3D tabletop interfaces. In the next section these frameworks are presented, along with some of the resulting applications that have a notable potential in 3D tabletop interfaces.

2.2 INTRODUCING TABLEGESTURES AND SOME RESULTING APLICATIONS

As mentioned in the motivation chapter, before expanding tangible tabletops beyond the surface, I developed some previous work in the area of tabletop interaction using the Reactable(Sergi Jordà et al., 2005) tabletop device: a round shaped table with a projector and a camera underneath. Three coding frameworks, initially thought for 2D tangible tabletop interaction, have been implemented and used for teaching in a computer science degree and some Master programs at UPF. The first one, "TableGestures"[8], made as an OpenFrameworks[9] add-on, allows and simplifies the coding tasks for "any purpose" tangible application. The second one, the Musical Tabletop Coding Framework (MTCF[10]) (Julià, Gallardo, & Jordà, 2011), helps those people that work in the field of audio but who may not have the necessary programming skills to program a musical application on tangible interfaces via real-time coding with PureData. Similar to MTCF, mobile MTCF (mMTCF) (Gallardo et al., 2013) is a coding framework for mobile android devices[11] that can also interact with tangible tabletop surfaces.

Having been designed for traditional tangible interaction or tabletop devices that only track objects and fingers on the surface, these frameworks have produced quite a few interesting applications that could be considered in the area of the 3D tabletop.

[8] https://github.com/chaosct/ofxTableGestures
[9] http://openframeworks.cc/
[10] https://github.com/chaosct/Musical-Tabletop-Coding-Framework
[11] https://play.google.com/store/apps/details?id=mobilemtcf.dani.main

2.2.1 TableGestures

TableGestures is a coding framework initially designed to instruct students to implement their own tabletop gestures on a tangible tabletop device. It requires ideas and parts of some previously developed apps: TDesktop(Gallardo & Julià, 2007), TurTan(Gallardo et al., 2008), Tjukebox (Gallardo & Jordà, 2010) and Songexplorer(Julià & Jordà, 2009). This coding framework can be divided into three parts:

- Tabletop interface helper
- Gesture manager
- Graphic areas

Tabletop interface helper is the part in charge of facilitating the process of programming for a tabletop interface. This module is in charge of listening TUIO-messages (Kaltenbrunner, 2009) from reacTIVision(Kaltenbrunner & Bencina, 2007), a finger and fiducial tracker[12], transforming output graphics for tabletop calibration and simulating tabletop events when a tabletop interface for testing is not available. This helper is completely transparent to the user and when the app is compiled, these functionalities are automatically enabled.

Gesture manager is the piece in charge of listening to events from reacTIVision and generating gesture events. It is structured from the composition of other events, having as the basic input a reacTIVision received-message event. It implements gesture generators that can receive events from other gestures for generating high-level gestures. (i.e. A high-level gesture that receives events from “tap gestures” and “finger movement” to implements the gesture “tap and drag”). As it is meant to be used by students that will build their own gestures, by default, it includes a few basic gestures with its respective events:

- Basic fingers: reports the three basic finger events: add, update and remove.
- Basic objects: reports the three basic object events: add, update and remove.

[12] ReacTIVision would be explained in detail at 4.2.2Tabletop markers

- Direct fingers: the same as basic fingers but returning an updatable instance.
- Direct objects: the same as basic objects but returning an updatable instance.
- InputGestureHand: reports grouped fingers as hands.
- InputGestureLongPush: reports a long finger tap.
- InputGestureTap: reports finger taps.

Instead of orienting the graphics library to a widget library, TableGestures' logic has been built with **graphic areas**. These areas can be defined by the programmer and only listen for certain input gestures that they have been subscribed to. The behaviour of graphic areas is governed by a priority system, the upper area has the focus ahead the others. However, there are mechanisms to change this, for example, by declaring always on top areas or changing the priority dynamically.

The use of composed gestures and graphic areas instead of a hard and defined gesture dictionary or widget list has the purpose of forcing the programmers to build their own gestures and graphic elements by accessing all the low level gestures, such as the raw data received from the reacTIVision, rather than using an arranged structure, such as the use of the surface SDK by defining elements in the XAML file.

Although this coding framework was developed for teaching purposes, it is currently also used for several applications in museums [Figure 7] and custom projects. Thanks to the "permissiveness" of this framework, the resulting applications may go beyond the traditional tabletop interaction whereby, through the use of complex gestures, the interaction space can be suggested beyond the display.

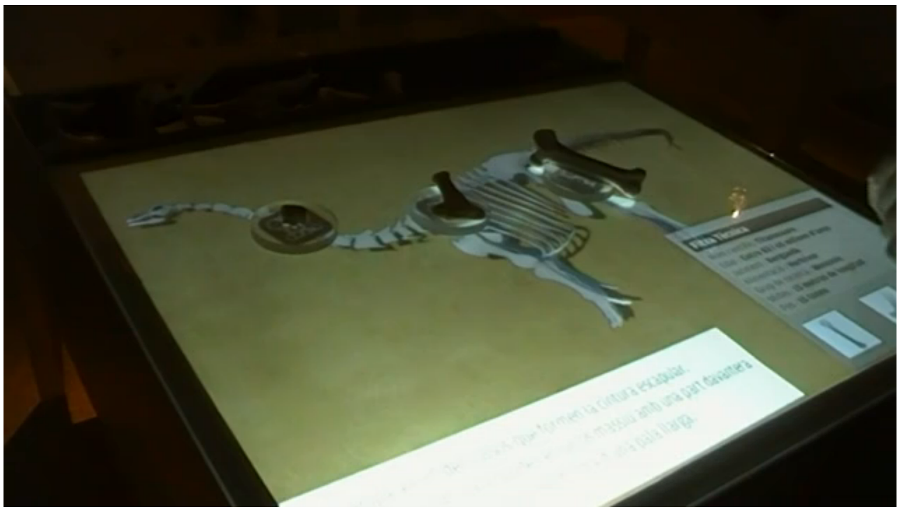


FIGURE 7: APLICATION THAT USES TABLEGESTURES: USERS CAN MANIPULATE A SERIE OF BONES ON THE SURFACE TO RECREATE A PREHISTORIC DINOSAUR.

TableGestures, being designed as an OpenFrameworks add-on, requires a programming background that sometimes our students do not have or a fast tabletop development for some rapid prototyping applications is needed. Based on this framework, in our lab, we created another coding framework that do not requires programming skills neither large compiling and testing processes. This framework, oriented for a master course where most of the students are not engineers but musicians is called Musical Tabletop Coding Framework.

2.2.2 MTCF AND MMTCF

The Musical Tabletop Coding Framework and its mobile version were developed as rapid prototyping tabletop interfaces. The core of MTCF is the aforementioned TableGestures framework inheriting all the built-in extras such as calibration, simulator and tangible areas. MTCF is programmed with PureData[13], a connecting box graphic programming language originally designed for sound synthesis and music creation. The main advantage of working with PureData is that the user can see the instantaneous real-time results of the code while programming instead of having to compile the code, upload it to the interface and only then test it. MTCF works as proxy program interfaced by a library created for PureData that sends instructions to the core application [Figure 8].

[13] http://puredata.info/

There are not too many differences between the version made for tabletops and that for mobile devices. On the tabletop version, apart from creating graphics areas and defining the application, users can interact with tangibles and complex waveform graphics. Differently, in the mobile version, the users have access to all the sensors of the device (accelerometer, magnetometer, gyroscope, light sensor, etc.).

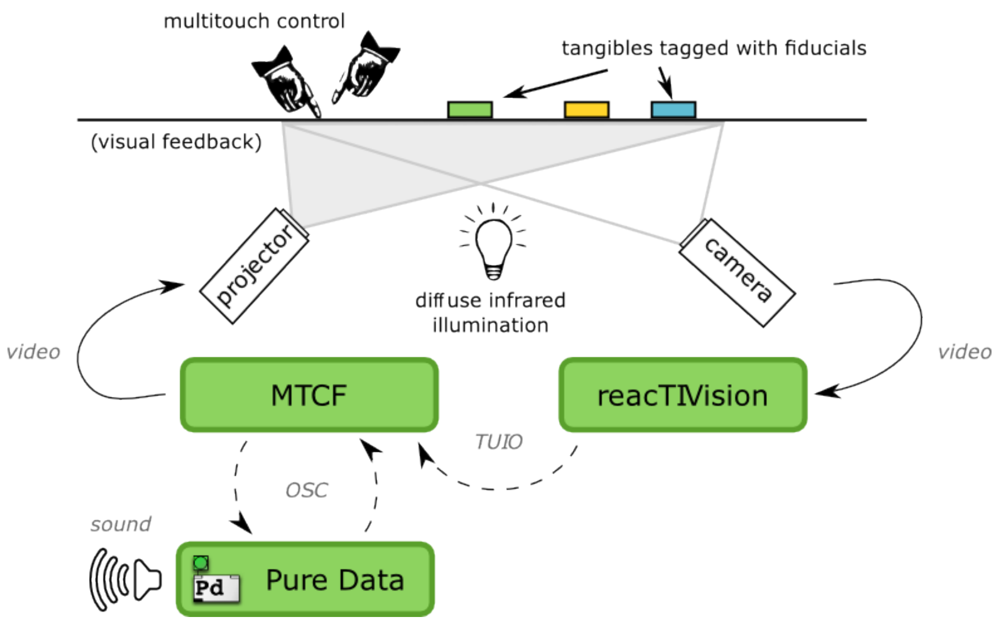


FIGURE 8: MTCF STRUCTURE.

2.2.3 RESULTING TABLETOP APPLICATIONS THAT SUGGEST 3D INTERACTION

More than thirty applications were created using these three frameworks. Thanks to the composed gestures, tangibles on TableGestures and the use of mMTCF for creating mobile apps that were connected to a tabletop device, some of the resulting tabletop applications suggested a third dimension or, at least, expected implicit interaction above the surface unless the tabletop interface used only detected fingers and objects on its surface. Below is a brief description of some of these applications:

Project Walk is a proof of concept of a high-level gesture interpreter. It was conceived as a terrain with different patterns placed randomly. The purpose of the application was to walk with two fingers along the table and explore the different sound textures that the “terrain” could offer (grass, sand, water, etc.). The gesture interpreter was designed to detect the walking movement by tracking step direction, cadence and step distance. The user, therefore, has to imitate the walking movement with finger gestures on the display that generates the effect of walking hands beyond the display [Figure 9].

Smashtable is a card-based game inspired by the popular game "UNO". It uses special tangibles made of sponges with fiducials attached to it for interacting with the application. The players have to show their cards in turn by smashing their sponge on the table. The cards that are shown could contain ability games consisting on various skill tests through which a user wins the battle (for example, all the players have to smash the sponge in the middle of the table, or the area of the next clockwise player, etc.). In this game the cards are virtually represented on the table, but the tangible interaction by feint with the sponges, or the process of trying to smash the sponge in a reduced area, implies the use of a third dimension (not detected by the system but intrinsic in the game) [Figure 10].

80's Table is a retro "pacman-space invaders-mariokart" game whereby players interact by spinning a round tangible as if it were a kart steering wheel and by brandishing a flyswatter to reduce the velocity of the opponents' karts. When a user sees that opponent players are going to hit their kart, theycan protect it with their hand or by activating a shield. Again, similar to Smashtable, in 80's table, the interaction in the air is used in the game but not detected by the system [Figure 11].

Our little choir (Katsaprakakis, 2011) is an audio-based application for synthesising vocal phonemes. By rotating tangibles on the table, the user can change the various parts of the synthesizer thus generating different sounds. The interesting part of this application is the use of a tablet and smartphones as external movable displays filled with accelerometers and a multi-touch screen sending the gathered interaction data to the tabletop device [Figure 12].

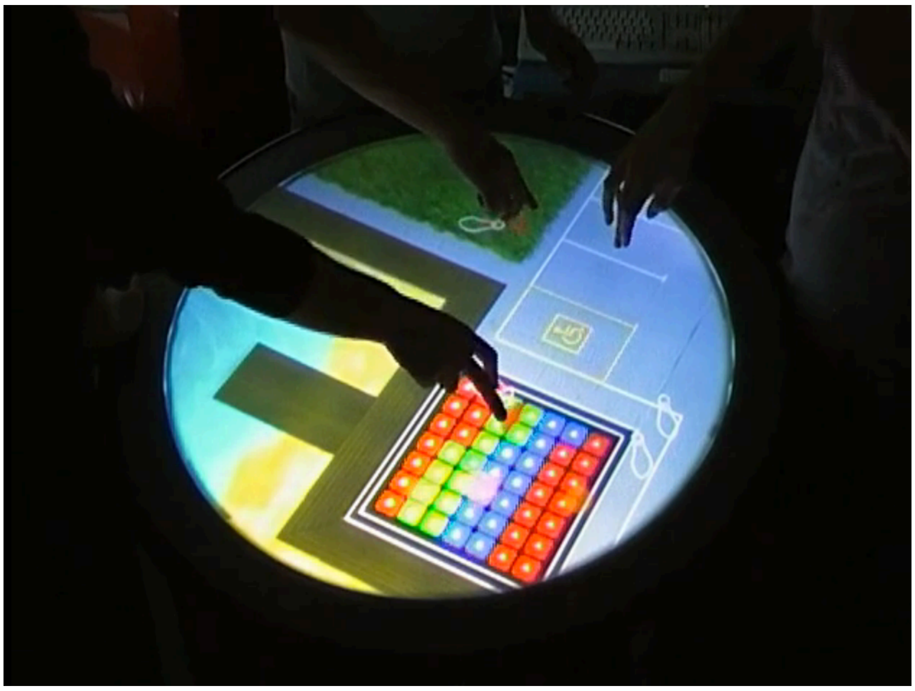


FIGURE 9: PROJECT WALK BY CIRO MONTENEGRO AND RICARDO VALVERDE (2010).

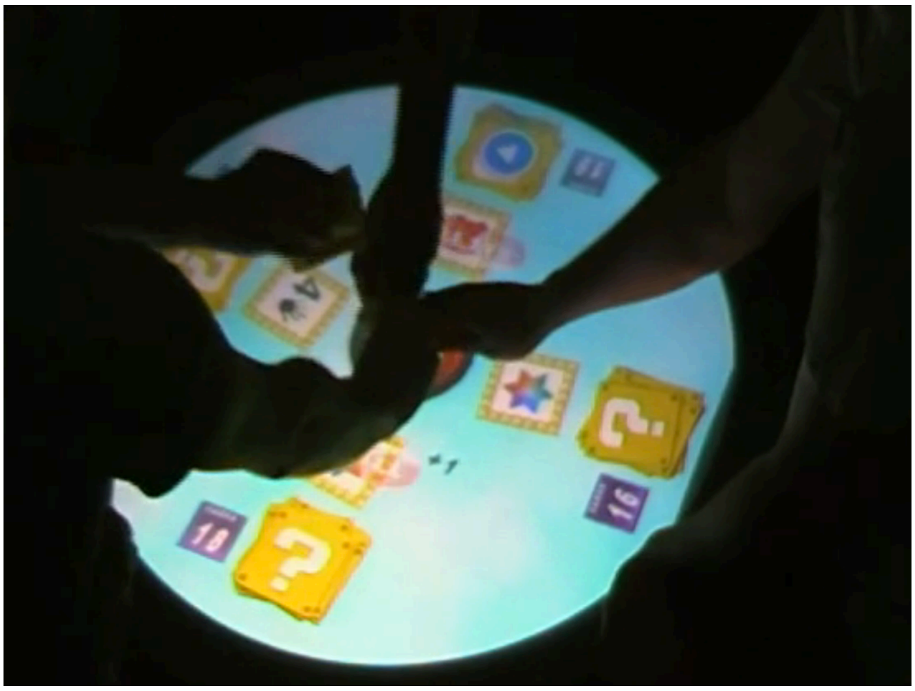


FIGURE 10: SMASHTABLE BY ALEJANDRO SANCHEZ (2010).

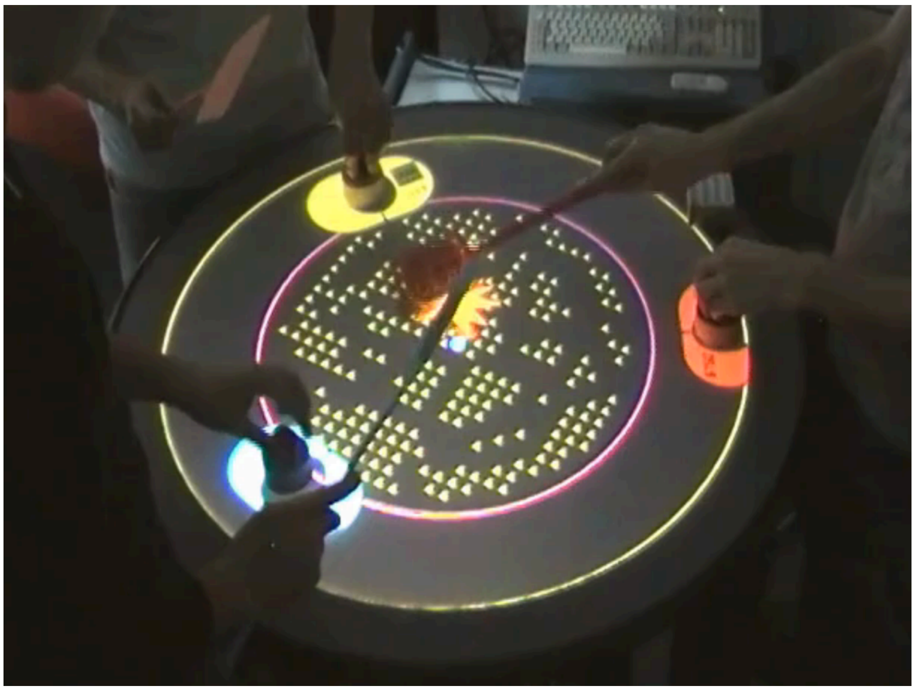


FIGURE 11: 80-TABLE BY DAVID PANYUELA, MARC RAMIREZ, MIQUEL CORNUDELLA AND PIERO SACCO (2010).

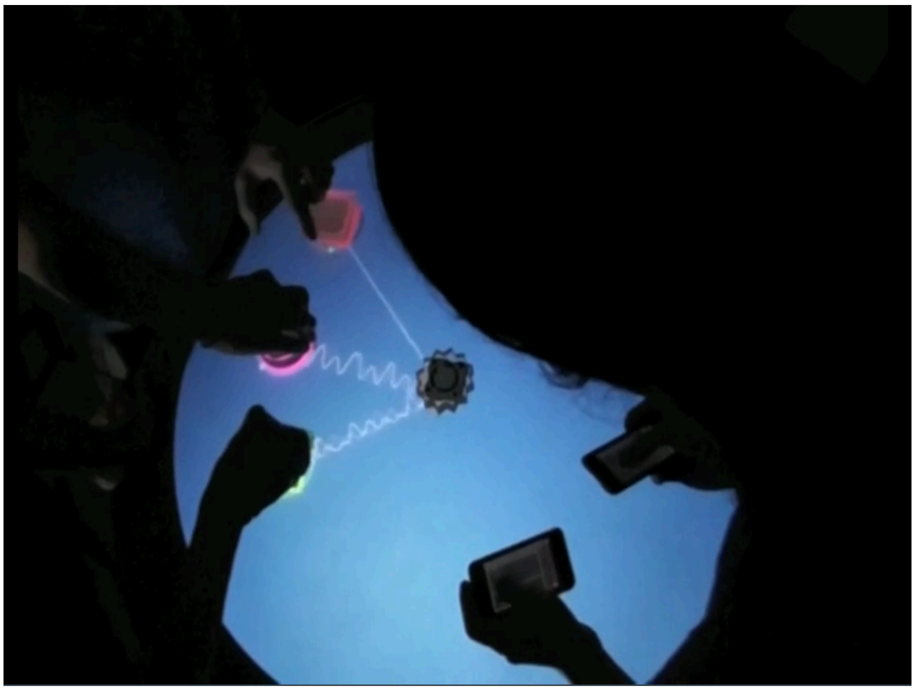


FIGURE 12: "OUR LITTLE CHOIR" BY ALEXANDROS KATSAPRAKAKIS (2011).

Project	3D component
Project walk	Walking hands beyond the screen implies a complex gesture involving the whole hand. While one finger is touching the surface, the other is performing on-the-air interaction although it is not tracked by the system until it touches the surface again.
Smashtable	There is an implicit on-the-air gesture part induced by the game dynamics when the users try to trick each other in the air, thus preventing the opponent from winning the game.
80's Table	Similar to Smashtable. There are the same implicit in-the-air gestures, but the blocking gesture is added whereby the user puts hand between the table and the flyswatter to prevent the opponent from hitting their avatar.
Our little choir	This application employs an external display combined with the use of accelerometers and gyroscopes (movement) to interact with a 2D tangible tabletop interface.

TABLE 2: DESCRIPTION OF THE 3D COMPONENT OF THE PRODUCED APPLICATIONS.

These applications demonstrate the potential of 3D tangible tabletop interaction; even with a common tabletop it is possible to expand the interaction to the third dimension by using tangibles with certain affordances or external connected devices that enrich the interaction [Table 2]. In all of these examples the tabletop used is a Reactable tabletop running reacTIVision and the aforementioned frameworks. This table is not able to track elements above the table, only fingers and tangibles on the surface, but all of these applications use interaction in the air without being detected by the system (at least until an element hits the surface). Expanding the interaction beyond the surface by tracking in-the-air elements and showing some feedback would significantly increase the degrees of interaction and produce new paradigms on in-the-air continuous interaction. In fact, these applications can be classified in the 3D tabletop taxonomy introduced by Grossman as shown in the next section.

2.3 WHAT IS 3D TABLETOP INTERACTION

According to Grossman (Grossman & Wigdor, 2007), any system that presents a 3D virtual environment on or above an horizontal surface can be considered a 3D tabletop interface. This definition is wide enough to include 2D tabletops with special “tangible” affordances in the 3D space such as URP, whereby the wireframe representations of buildings placed on the table generate virtual shadows on the surface simulating the presence of an above the surface elements, to including full virtual reality environments with augmented reality tools such as head mounted displays where even the horizontal surface can be virtual.

The first tabletop interface that was considered a 3D tabletop interface, was the Responsive Workbench in the early nineties (W. Krueger & Froehlich, 1994); a table-based stereoscopic surface where the user interacts by means of data gloves while wearing stereoscopic glasses [Figure 15, left]. Following the same structure but using a large semi-vertical display, ImmersaDesk (Czernuszenko et al., 1997) was conceived a virtual reality display whereby users interact with the data by means of a remote controller[Figure 13].

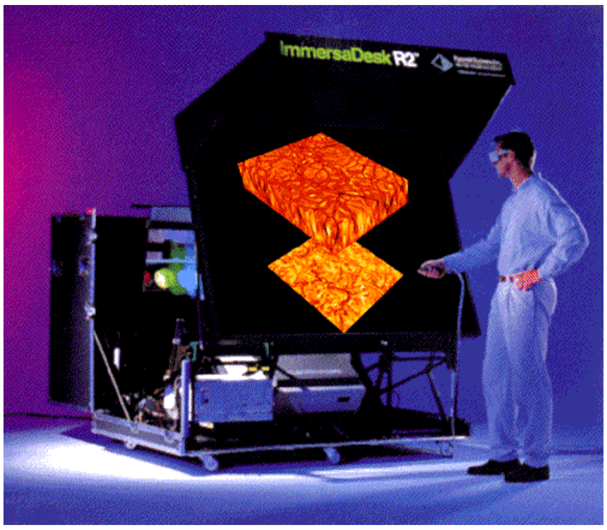


FIGURE 13: IMMERSADESK[14].

[14] http://www.hq.nasa.gov/hpcc/insights/vol8/images/turb2LG.gif

What the two aforementioned 3D tabletop devices had in common was the non-existence of any tangible device to interact with the interface. The first device to mix tangibles with a 3D tabletop interaction was the Metadesk (Ullmer & Ishii, 1997) consisting of an horizontal projected surface with some physical objects that were used as tools. These were: an active lens, built with an arm-mounted flat-panel screen, a passive lens, a transparent surface through which the desk projected and some office objects used as "tangible tools". The objective of Metadesk was to explore the use of tangible elements to drive elements of computer interaction by using real physical entities that can be touched and grasped. Whole-hand and hand-shape interaction above the display have also been a concurrent topic of on-the-air interaction whereby many hand poses are defined and reinterpreted across time with the same purpose: to interact with virtual data represented on a flat horizontal screen (Carreira & Peixoto, 2007; Epps et al., 2006; Hinckley, Pausch, Proffitt, & Kassell, 1998; Huppmann et al., 2012; Kim, Izadi, Dostal, & Rhemann, 2014; M. W. Krueger et al., 1985a; Marquardt et al., 2010; Pyryeskin, Hancock, & Hoey, 2012; Sutcliffe, Ivkovic, & Flatla, 2013; Wu & Balakrishnan, 2003).

With the arrival of new technology and tracking techniques, 3D tabletop devices have started to merge input and output methods thus creating direct 3D data manipulation systems. This is the case of Z-touch (Takeoka, 2010), a tabletop that can track fingers above the surface by using a complex grid of infrared emitters. The tabletop, introduced in "the continuous interaction space" (Marquardt et al., 2011), which uses a tactile surface mixed with a motion capture system for tracking objects, hands and fingers beyond the surface or the Holodesk (Hilliges et al., 2012). The Holodesk is a desk that combines a Kinect camera for tracking user gestures, a front camera for user head tracking and a half silvered mirror where 3D models are projected and perceived by the user as holograms. In HoloDesk, the systems know at all times the positions of the hands of the user, the volumetric data of any used object and the position of the user's head in order to show holograms that can be touched and manipulated.

2D tabletop devices also have 3D tabletop applications that fit in the Grossman's definition. This is the case of "Sticky tools" (Hancock, Cate, & Carpendale, 2009).

That draws a virtual sandbox expanded below the surface where different virtual objects can be manipulated by performing 2D gestures on the surface or by augmenting 2D tangibles such as the URP, Lumino or the interface used at "the tangible 3D tabletop", whereby tangibles are augmented by mapping real-time textures on them (Baudisch, Becker, & Rudeck, 2010; Dalsgaard & Halskov, 2012; Underkoffler & Ishii, 1999)

Many different kinds of 3D tabletop interfaces exist. Some show volumetric data above the screen, others underneath, others only track in-the-air gestures, etc. A 3D tabletop taxonomy introduced by Grossman (Grossman & Wigdor, 2007) organizes these tabletops, categorizing them into three main areas whereby they are distinguished in terms of Display properties, Input properties and Physical properties [Figure 14].

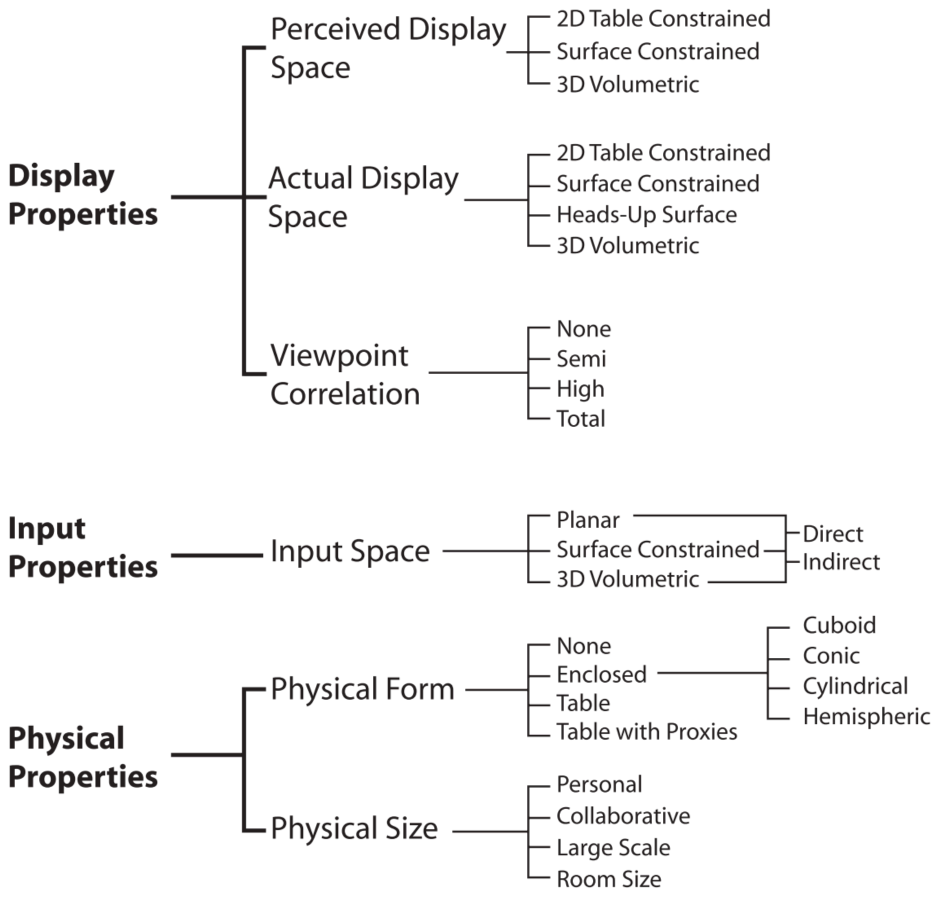


FIGURE 14: TAXONOMY OF INTERACTIVE 3D TABLETOPS INTRODUCED BY GROSSMAN[15].

[15] Schema extracted from (Grossman & Wigdor, 2007).

As seen in Figure 14, the first level of the taxonomy classifies the tabletops according to the display properties, the input properties and the physical properties of the tabletop. To illustrate this taxonomy, some of the previously mentioned 3D tabletop applications have been grouped and are presented below in the distinct levels of the taxonomy:

The tabletops to be classified are: 1) Project walk; 2) Smashtable; 3) 80's table; 4) URP; 5) Sticky tools; 6) Z-touch; 7) My little choir; 8) Responsive workbench; 9) "The continuous interaction space" Marquardt's table; 10) Lumino; 11) Immersadesk; 12) Holodesk.

- Display properties
  - Perceived display space
    - 2D table constrained: traditional 2D tabletop interfaces [1, 2, 3, 4, 5, 6].
    - Surface constrained: horizontal augmented displays or surfaces with augmented tangibles [7, 8, 9, 10, 12].
    - 3D volumetric: AR visuals based [11];
  - Actual display space
    - 2D table constrained: only tabletop based displays [1, 2, 3, 4, 5, 6, 8, 9, 11].
    - Surface constrained: the tabletops that can sow results on the tangibles [7, 10].
    - Heads-up display: tabletops that projects images into a middle layer or head-mounted displays [12].
  - Viewpoint correlation
    - None: traditional tabletops [1, 2, 3, 4, 5, 6, 8, 10]
    - Semi: external displays [7, 9].
    - High: head tracking [12].
- Input properties
  - Direct 2D: direct touching on 2D surfaces [1, 2, 3, 5, 6, 7, 9].
  - Indirect 2D: by using an external device such as a mouse interacting in a reduced space for reaching areas of large interfaces

        or above the surface interaction for the manipulating of 2D objects [6, 9].
    - Direct surface constrained: interacting above the surface by applying gestures on tangibles [4, 7, 9, 10].
    - Direct 3D touching: grabbing objects in 3D space. [8, 12].
    - Indirect 3D: interacting with large 3D space by using external tools or dislocated interaction. [11].
- Physical properties
    - Physical form
        - None: the entire environment is virtual [11].
        - Table: planar table surface [1, 2, 3, 4, 5, 6, 8, 12].
        - Table with proxies: planar table surface spatially augmented [7, 9, 10].
    - Physical size
        - Personal [5, 6, 12].
        - Collaborative [1, 2, 3, 4, 7, 10].
        - Large scale [8, 9, 11].

Physical properties are linked to the physical constrains of the interface's device–making. Therefore, in terms of these properties, the tabletops can only be classified into one category. This is not the case in the classification according to display and input properties. The applications [6,9,7] can be classified into two or more sub-categories under the Input properties because they may comprise of several distinct types of interactions: 6 and 9 can sense fingers on the surface and above the surface thus making them eligible for the direct 2D input (direct touching on the surface) and indirect input (performing gestures in the air) sub-categories, while 7 and 9 can also sense fingers on the surface and, simultaneously, external displays may be used for direct surface input.

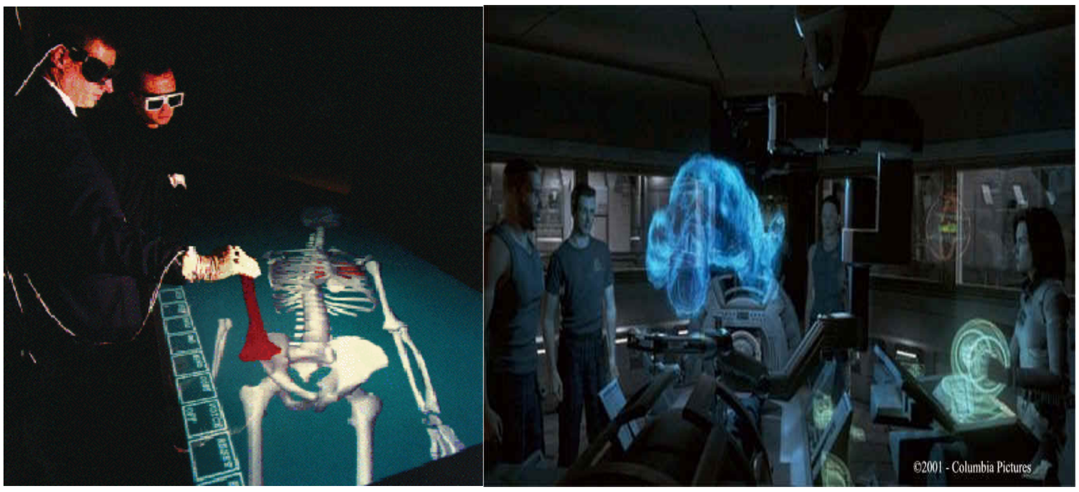


FIGURE 15 KRUEGER'S RESPONSIVE DESK (LEFT); FINAL FANTASY MOVIE CAPTURE OF VIRTUAL INSTRUMENTS [16](RIGHT)

Grossman's Taxonomy shows that 3D tabletop interfaces embrace many computer science disciplines: Virtual reality, volumetric displays, tabletop interaction, tangible interaction, etc. In this dissertation we propose to expand the current tangible and tabletop interfaces by enriching its gestures, which entails adding new degrees of freedom to the tangible objects and capturing in-the-air and continuous hand gestures. Expanding tangible tabletop interaction to the third dimension, does not need to be related to visualizing and manipulating 3D content, but can consist of an in–the-air interaction in a continuous space, visualizing graphics or feedback above the screen and enriching actual 2D gestures.

2.3.1 Interacting in the air

In-the-air interaction is not a new concept. It originates from environments of virtual realities with first head mounted displays (Sutherland, 1968) and from the development of Data Gloves in the 80's (Quam, 1990; Zimmerman, Lanier, Blanchard, Bryson, & Harvill, 1987). Myron Krueger, an engineer and media artist, and a passionate about unencumbered rich gestural interaction, developed camera-based installations to track the user's full-body interaction as well as hand and finger movement (M. W. Krueger, 1991). First with VideoPlace (M. W. Krueger, Gionfriddo, & Hinrichsen, 1985b), a full-body tracking system

[16] 2001 © Columbia pictures.

whereby users interact with a big screen, and, lately, with VideoDesk, Krueger showed the use of two-finger and two-hand gestures to redefine the shapes objects (scaling or translating). It was not until 2002, with the film "Minority Report"[17], when 3D in-the-air gestures reached a mass media impact. In the film, the main character manipulates a virtual wall by using hand gestures in the air. This technology, made reality in 2010 by John Underkoffler[18] and shown in a TED talk, revealed the weak point of this kind of interfaces before being implemented: in the movie, the main character gets distracted and everything he had been doing with the wall gets ruined.

The other weak point of interacting in the air is the fatigue caused by holding one's hands in the air during the entire work session. This issue, pointed out in the 90's (Baudel & Beaudouin-Lafon, 1993) and also reported in multi-touch environments (Yee, 2009), demonstrated that full-hand gestures and large-scale gestures should be avoided and replaced by quick and local gestures enabling a comfortable arm pose. In [Figure 15] there is a comparison of two medical applications for full-body diagnostics. The one in the left (responsive desk) is a real world product, while the one in the right is from an animation movie (virtual body representation). These examples show the two different approaches: the former directly covers the whole virtual body, thus forcing the user to keep an uncomfortable position in the air for a certain period of time, while in the latter example, virtual body representation is only used to view the general results of the surgery and the user is able to manipulate in a more comfortable and reduced virtual interface.

2.3.2 In the air tabletop gesture affordances

In the air tabletop interaction is often limited within two distinctive areas. These two interaction spaces are use to be separated and it is very hard two find examples that uses gestures involving these two areas:

- Surface area, includes touch and tangible interaction by directly reaching parts of the display surface with hands or tangibles. The gestures

[17] http://www.imdb.com/title/tt0181689/

[18] http://www.ted.com/talks/john_underkoffler_drive_3d_data_with_a_gesture

contained into this interaction space are mainly used for selecting, moving, scaling, rotating and grabbing virtual objects as pointed in the following projects:
  - 2D gestures for controlling 3D representations (Hancock et al., 2009).
  - Stacking tangibles (Baudisch et al., 2010).
  - Hand positioning on the surface (Marquardt et al., 2010).
- Above the surface area, incudes free hand gesture recognition and mapping gestures to particular actions such as:
  - Hand shape (Epps et al., 2006).
  - Hand orientation (Carreira & Peixoto, 2007)

Reducing the interaction space only into these two distinct areas ignore the interaction space that exist between them and where natural interaction keeps it meaning. On real tables, users can perform actions on the surface such as writing, moving, rotating but it is not possible to perform actions only above the table keeping a relation to the table's surface, the traditional table's actions use to start on the table and continuously move to above it or vice versa.

Some early approaches explore these gestures, this is the case of the air pinching gesture explored by (Hilliges et al., 2012). Air pinching consists on picking up a virtual object represented on the surface and left it at the desired position without touching the surface. Another gesture into this category is the introduced by (Wilson & Benko, 2010) named object pickup where objects where picked up with two hands directly from the surface. Using this continuous space but instead of hands it uses external displays is the one presented in (Subramanian, Aliakseyeu, & Lucero, 2006) where the space above the table is discretized and could be explored by moving the external display up and down.

The work done by Marquardt (Marquardt et al., 2011) goes an step further and defines this continuous interaction space as the space composed of the direct touch surface and the space above. For explaining this interaction space, he introduces some gestures involving hands, finger, objects or a combination of them. In the following lines some of them are introduced:

- *Extended continuous gestures to avoid occlusion*: Consist on direct touching the surface and continue the gesture in the space above it to avoid the occlusion of the digital content.
- *Lifting gestures to reveal objects*: Consist on lifting virtual objects by touching them and raising the hand to show the virtual content hidden behind them.
- *Lifting to adjust scale precision*: allows lift the hand when manipulating an input widget like a slider to reaching a wider slider range.
- *Interaction with discrete layers*: Summarizes the discretized interaction space introduced by Subramanian.
- Magic lenses and viewports: This gesture the system is aware of the position and pose of a smartphone and uses it as magic lens (Bier, Stone, & Pier, 1993).
- 6-DOF Manipulation: by picking-up a digital 3D object, the user is able to move and rotate it in the air as it was hold by the user's hand (limited by the movement od the wrist join).
- Feedback of possible actions by hovering: Consists on tracking the user's finger above the surface and show some reactions with surface's virtual objects like GUI buttons that can glow when the finger is hover.

These gestures are a good start point for building and developing our expanded tabletop surface and keep them as an interaction reference for our future developments. In Chapter 5 a classification of these gestures and the ones introduced by our developed interface will be analysed and classified.

2.3.3 Expanding tabletop interaction to the 3rd dimension

At the very beginning of this chapter we posed the question of the possible advantage of expanding tabletop interaction beyond the surface. Afterwards, we presented TUI's and showed that tangible interfaces opened a two-way communication channel for the direct manipulation of data. Furthermore, we also introduced Tangible tabletop interfaces as multipurpose interfaces but with the surface restriction of a 2D interaction. We discussed how 3D tabletop interfaces break the limitation of 2D surface computing and many different and multimodal solutions were introduced and classified. We also mentioned our background on tabletop interfaces, introducing three frameworks for coding tabletop interfaces and some resulting applications that can be considered as 3D tabletop applications. Consequently, we now express our strong belief that expanding tangible tabletop interaction beyond the display will enrich the interaction metaphors and increase the communication bandwidth between the user and the computer.

After comparing all the aforementioned 3D tangible interfaces, except Metadesk and the table built for "the continuous interaction space" Marquardt's tabletop, these interfaces can be classified in only one area of the Input properties in Grossman's taxonomy, and very few combine hand gestures with tangible interaction. It is also difficult to find examples where the transition of surface interaction to beyond-the-surface interaction can be done through fluid and continuous interaction. Marquardt (Marquardt et al., 2011) defined a continuous interaction space as the unification of in-the-air gestures and direct-touch on the surface. He defended this definition by showing some in-the-air gestures that end on the surface and vice versa. By combining finger, hand and tangible interaction, this continuous interaction space could be more complex than pointed out by Marquardt and some "in-the-middle" categories could be applied.

These categories would be intermediates between continuous interaction and pure 2D or 3D interactions. They could be defined as "two-state continuous interaction" where the meaning of the action changes depending on whether it is performed above the surface or on the surface, which can be seen as configuring a tool versus using the tool. Magic lenses (Bier et al., 1993) and external displays could also take an important interaction role in 3D tangible tabletops. These movable displays were mainly used for showing hidden data on tabletops (Izadi et al., 2008), build augmented content changing tangibles (Weiss, Hollan, & Borchers, 2010), video jigsaw puzzles such as the ones shown for the surface 1.0, etc. Moving and tracking these external displays above the screen can introduce endless functions:

- Visualize different layers of a map: satellite, political boundaries, streets, terrain, …

- To be used as volume slicing displays (Cassinelli & Ishikawa, 2009) in order to view cuts of a 3D model.
- For exploring large collections of databases sorted into categories whereby each category corresponds to a z-value.
- For multiple browsing and interaction purposes.

We propose to build a multi-purpose tangible tabletop that would show visual feedback on the surface, above the surface (on hands and objects) or both, and, simultaneously, implement different inputs by reducing the hardware complexity by an all-in-one device able to track:

- Fingers on the surface
- Tangibles on the surface
- Hands above the surface
- 6 DoF tangibles on and above the surface
- External or portable displays

In brief, we look for an expanded tangible tabletop surface, that does not necessarily require 3D visualization, but can be used to expand the bandwidth between users and machines and provide new gestures and interaction metaphors with new tracked data [Table 3].

	Traditional tabletop interfaces	3D tabletop interfaces
Fingers	Touches	Touches, hover-touches and pointing tool.
	X, Y	X, Y, Z, direction, hand ID
Tangibles	Access to surface data, fixed tool.	Access to surface data, continuous modification of the surface data, configurable tool.
	X, Y, angle, ID	X, Y, Z, yaw, pith, roll, ID
Transparent tangibles	Magic lenses	Magic lenses, 3D data browsers, volume slicers, …
	X, Y, angle, ID	X, Y, Z, yaw, pith, roll, ID
Hands	Blob interaction	Hand gestures in the air, pointing tool
	Blob (X, Y) points	Blob (X, Y) points, fingers, wrist, palm, opened/closed, pinch, etc.

TABLE 3: TABLETOP INTERFACE'S DEGREE OF FREEDOM VS. 3D TABLETOPS INTERFACES.

However, before further discussing the framework definition of gestures and tangible interaction, we need to build a hardware device that is able to satisfy our needs. This will be addressed in the following chapter.

Chapter 3 EXPANDING TABLETOP'S HARDWARE TO THE 3RD DIMENSION.

In the previous chapter we introduced our aim to extend tabletop interaction above the surface and the need to build a device that would help reach this objective. Creating a tabletop able to detect surface and above–the-surface interaction implies the understanding of the technology concerned with traditional tangible tabletop interaction and in-the-air tracking technologies. Before introducing the technology related with 3D tangible tabletops and to better understand the technology underlying traditional tabletops, recent 2D tabletops technologies are introduced. The existing technological approaches will be summarized and we will establish which of those technologies have the potential to be ported to the 3rd Dimension. After the traditional tabletops review, some 3D tabletop devices will be presented, among these, we will introduce the SecondLight, a tabletop given to us by Microsoft Research and on which we will introduce some modifications to satisfy our aim.

3.1 TRADITIONAL TABLETOP HARDWARE OVERVIEW

A tabletop by default has two communication channels: the input surface and the visual feedback on this surface. Output graphics on the surface can be done by using an embedded screen or a projection surface and a projector.

The use of a projector often implies large volume prototypes to feed everything into the table in the case of the rear-projected devices or to build specialized ceiling structures for the over-projection systems. The advantage of projector-based surfaces in front of embedded screens is that they leave sufficient space for fitting any tracking method. However, screen-based surfaces often offer better resolutions than projector-based tabletops.

Tabletop input-sensing technologies can be separated into three groups: electronic sensing using resistors or electromagnetic devices, infrared emitter-receiver-based tabletops and camera-based tabletops. Akin to visual feedback technologies, each of these groups has its advantages and disadvantages. While camera-based tabletops need a controlled light environment the ones that use electronic-sensing are more resistant and reliable, but on counterpart, the first group can sense a large variety of inputs (markers, shapes, touches and hands) and the second-mentioned ones are designed for multi-touch inputs[19].

3.1.1 Electronic traking solutions

Tracking fingers by using electronics is made by measuring different parameters (resistive, capacitive, electromagnetic) on a thin layer of glass that is mounted on a screen for getting the visual feedback. In this subsection are introduced resistive and capacitive screens showing some variations of the last ones and how they can be modified for tracking objects on their surface.

Resistive touchscreens

The cheapest touchscreens, which were popularized in the late-nineties by PDA Devices, and the first finger and stylus-based touchscreens were resistive touchscreens. These kinds of touchscreens are based on two conductive panels separated by an insulating layer usually made of very small silicon dots [Figure 16]. The front panel is typically made of a flexible hard-coated membrane, while the back panel is often a glass substrate (Schöning & Brandl, 2008). The screen's controller drives current to one layer and measures the received current on the other. When the user touches the front panel, the two conductive layers are connected and, by measuring horizontally and vertically the received current, the controller can estimate the X and Y position.

[19] In some cases electronic-sensing devices can track objects or markers but it is not an extended practice.

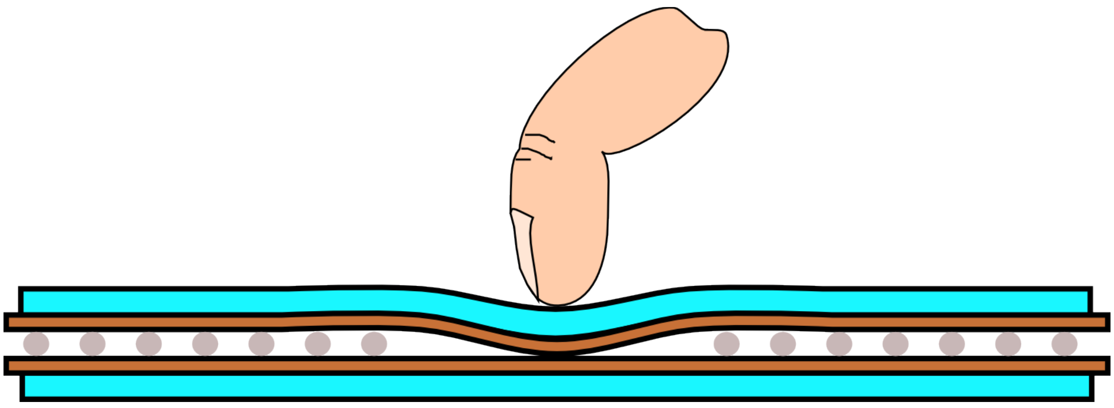


FIGURE 16: RESISTIVE TOUCHSCREEN LAYER.

Resistive screens are able to accept touch inputs from fingers, stylus pointers or any sharp object. However, given the constraints of its assembly, these screens can only accept one touch at a time. Although resistive layers are designed to be placed directly on a display, they provide low-clarity images[20] because the layers that enable the adequate functioning of the device are not fully transparent. In addition, it is not possible to cover them with a screen protector because this would have a direct impact on the performance of the resistive layer.

The advantage of resistive touchscreens is the simplicity of the hardware. By measuring resistance values on the X and Y sides of the screen, it is easy to recognize and locate a touch input. The biggest constrain of resistive screens is also due to the simplicity of the technology since they can only track one finger at a time. For this reason they were later replaced by other, more complex, multi-touch technology.

Capacitive touchscreens

In contrast to resistive screens, capacitive touchscreens can detect more than one touch simultaneously and allow fully transparent assemblies on any display without losing image brightness. Nowadays, the use of capacitive screens has been widely extended; we can find them in any smartphone, tablet or multi-touch computer.

[20]Resistive layers can offer about 75-85% of transparency.

Two different capacitive screen constructions exist: Surface capacitive touch surfaces and Projected capacitive touch surfaces. The most extended one is the projected capacitive touch surface because of its multi-touch tracking detection system, which uses a very thin grid of wires placed between two protective glass layers (Rekimoto, 2002). When the glass is touched, the screen's controller measures the electrical characteristics of the thin grid of wires, detecting the capacitance between the finger and the sensor grid to register a touch. Capacitive screens are not interfered by non-conductive solid objects, thus allowing the use of protective glasses or acrylic covers at the top in order to protect the entire screen. The most common capacitive screens can detect between 10 and 20 points. Although in theory they can detect more points, this does not occur because of a firmware restriction[21] or the bad design of the controller.

There are some projects that uses capacitive touchscreens for detecting more than touches, this is the case of diamond touch for user identification and sensetable for object tracking:

Diamond touch, produced by MERL labs (Dietz & Leigh, 2001), is based on a projected capacitive surface with a particular characteristic: This table can identify which user is touching the table by connecting the user's chair to different wave generators and marking them with a distinctive electronic pattern [Figure 17, right]. When a touch is produced on the table, the signal travels from the user's chair, through their skin and to the surface that can detect both the position of the touch as well as the user identifier.

Following the idea of identifying touches, is **Sensetable** (Patten et al., 2001). It uses a grid of wires and special tangible objects filled with electronics, for detecting the position (X, Y) and identification of these tangibles. Each tangible is configured with a different resonance factor and is identified by the system as an input with a unique ID. In addition, some tangibles are provided with a dial or knob that, when spun, their internal parameter is changed, allowing knob

[21] There are some companies that offer (paid) firmware upgrades for detecting more than 60 points with the same capacitive hardware

interaction on each object [Figure 17, left]. However, Sensetable, unlike capacitive screens, was not designed for finger detection and it can only detect active objects.

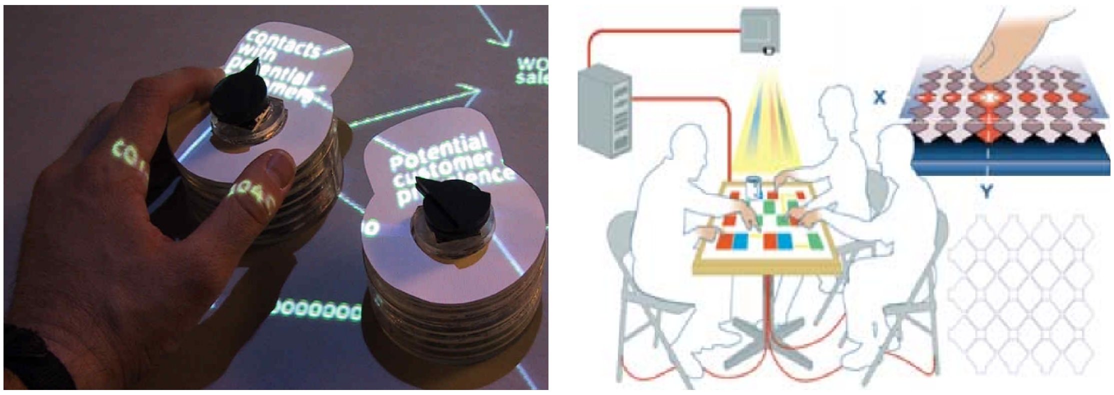


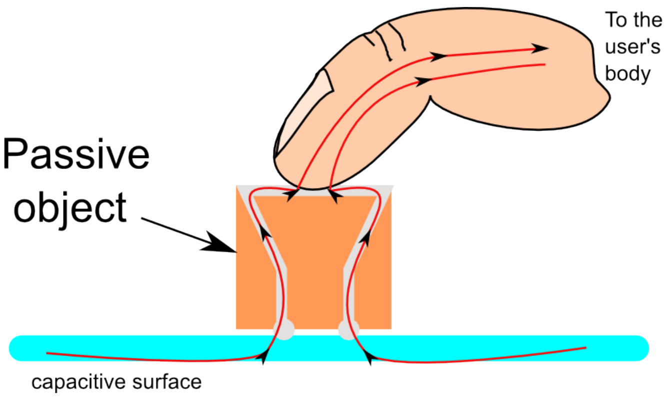
FIGURE 17: SENSETABLE TANGIBLES[22] (LEFT); DIAMOND TOUCH CAPACITIVE DISPLAY [23](RIGHT).

Object tracking on capacitive touchscreens

Tangible interaction on capacitive surfaces can be done, but it is not a very widespread practice since the resulting set of objects is reduced to only a few of thm or these require the use of expensive electronics fitted inside them. Tangible objects on capacitive surfaces can be divided in two groups:

- **Passive objects** connect the user's hand capacitance to a series of dots that the tangible has at the side of the screen [Figure 18, top]. While the user is touching the object, it is detected. However, when the object is left alone on the surface, it is no longer detected. Although there is this "always touching" constraint, passive objects are the only commercialized objects for capacitive screens [Figure 18, bottom] because they works on any capacitive device without having to previously modify the touch firmware.
- **Active objects**, similar to those used in the Senstable, contain a circuit board inside that changes the capacitance of the screen. These can be detected by the surface without having to be continuously touching them and, in some cases, they can be identified with different IDs by modifying its capacitance (Yu et al., 2011).

[22] Picture from: http://tangible.media.mit.edu/project/sensetable/
[23] Image extracted from (Dietz & Leigh, 2001)



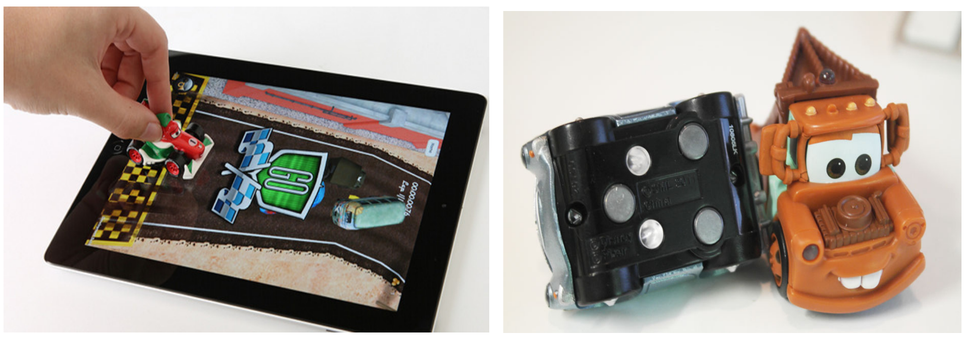


FIGURE 18: CAPACITIVE FLOW ON PASSIVE OBJECTS AND PASSIVE OBJECTS ON IPAD FROM APPMATES DISNEY CARS GAME[24]

Electronic tracking devices, and concretely projected capacitive touch surfaces, are the most widespread technology for multi-touch gestures. Nowadays this capacitive technology can be found into the most common popular market such as into smartphones or tablets. These devices use to detect up to ten fingers simultaneously, however as commented before it is a hardware restriction and can be found some industrial solutions that can increase that number up to sixty touches[25]. The construction process of these screens is not easy and cannot be produced "outside a specialized centre" because they have to be built using complex and expensive production processes in order to make the sensor thin and transparent.

3.1.2 Infrared emitter-receiver based tabletops

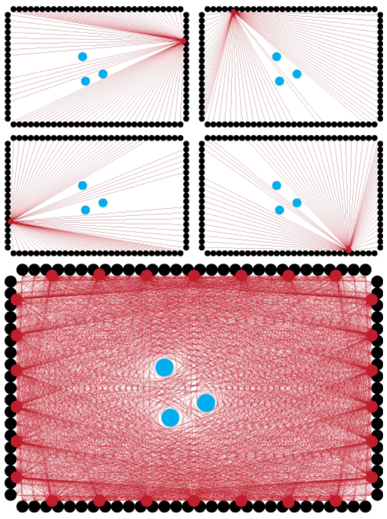
Besides pure electronic tracking via resistive or capacitive layers on the top of a screen, optical-based technology exists for multi-touch and tangible interaction satisfying the needs of most tangible tabletop applications. The main

[24] Images from http://www.gadgetguy.com.au
[25]http://solutions.3m.com/wps/portal/3M/en_US/Electronics_NA/Electronics/Products/Touch_Systems/~/Multi-Touch-Displays and http://www.zytronic.co.uk/ among others.

characteristic of this technology is the use of infrared emitters and receivers to detect touches and objects. There are not many products on the market that use this technology, but it is possible to find tabletops with IR emitters and receivers around the screen's frame (IR-Frames) and tabletops which screen is provided by an array of IR sensors behind them.

IR-Frames

IR-Frames are structured on an array of well-aligned emitters and receivers that are placed at the top of the screen's edges. When an object crosses this frame, the system can interpolate its X and Y coordinates by measuring the received IR-light. The main advantage of these frames is the scalability of the system; they can be produced for large screens whereby the effectiveness of other systems such as capacitive sensors is diminished due to electrical interferences. The greatest problem of these systems is the occlusion of the sensor frame when using multiple fingers or tangibles, which considerably reduces the supported touches by blinding the sensors. **Zero-touch technology** (Moeller & Kerne, 2012) goes a step further. It combines a large array of emitter/receiver diodes disposed across a rectangular frame, whereby each diode separately scans the other diodes thus creating a dense mesh that decreases the occlusion problems and detects finger intersections as well as the shapes of the objects [Figure 19].



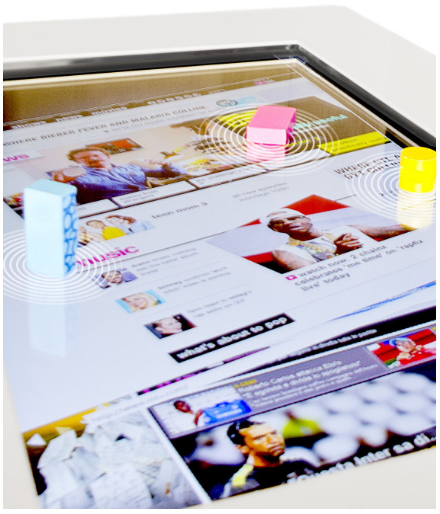


FIGURE 19: ZERO TOUCH SCAN MESH[26] (LEFT); TOUCH WINDOW: MULTI-TOUCH SENSOR DETECTING TOKEN'S SHAPES[27] (RIGHT).

By using dense infrared networks, Zero-touch frames can detect objects and determine the shape and size, which allows the use of a large variety of tangibles that have different physical characteristics [Figure 19, right].

IR-Emitters and receivers behind the screen

These devices are not as extended as IR-frames. An example of this technology is Microsoft Research's **ThinSight** (Hodges, Izadi, Butler, Rrustemi, & Buxton, 2007), which consists on placing an array of emitter and receiver diodes behind a disassembled TFT screen. The objective of ThinSight was to imitate what the sensor of a camera detects if the camera CCD were to point at the screen from behind filling the entry surface.

By adding dense sensitive IR-pixels matrices (IR emitter and receivers) behind a TFT screen and computing the captured data from each pixel, a detailed picture of what is placed on the table is obtained. Based on the idea of ThinSight, **Microsoft PixelSense** [Figure 20, left] was developed, becoming the second version of the first commercial tangible tabletop interface from Microsoft: the surface 1.0. On the screen of PixelSense, the pixels not only emit the red, green and blue colours, but they contain an infrared sensor for receiving the reflected

[26] Image from http://ecologylab.net/research/zerotouch/
[27] Picture extracted from http://www.touchwindow.it/en/overlay-multi-touch.php

IR-light thus reconstructing a scan of the objects touching the surface. Behind the screen's pixel array, PixelSense contains infrared emitters in addition to containing only a backlight for lighting graphics.

A large advantage of PixelSense is its capacity for "scanning" everything that is placed on the table thus providing a detailed frame of what the screen "sees". This puts PixelSense at an advantage when compared to its competitors since it can detect: unlimited fingers, blobs, fiducial markers and shapes (Figure 20, right).

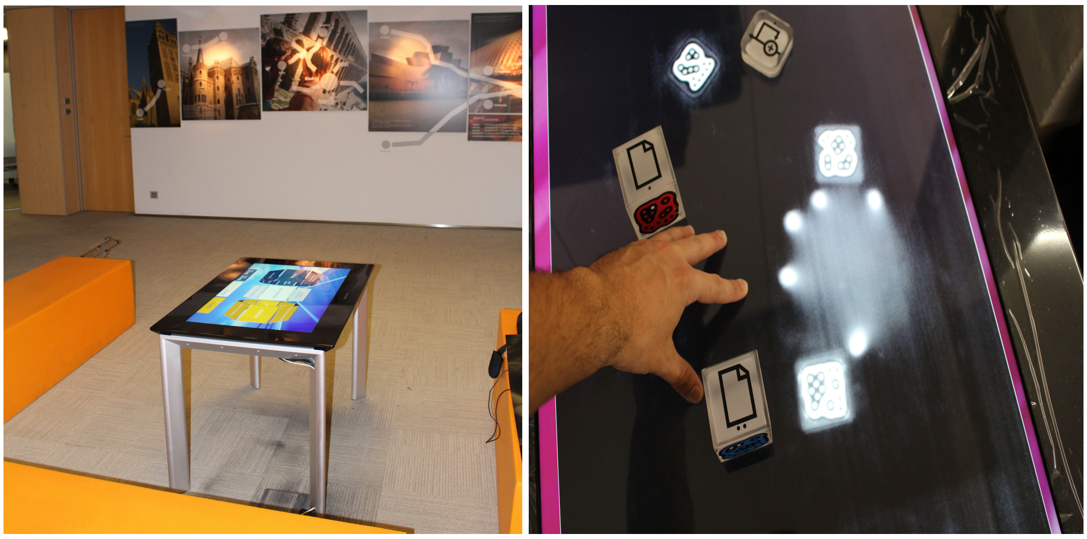


FIGURE 20: PIXELSENSE TABLETOP DEVICE (LEFT); PIXELSENSE SCANNING HANDS AND REACTABLE'S FIDUCIALS (RIGHT)

IR-emitter/receiver solutions, specifically PixelSense, generally function better than electronic sensing devices but only when certain conditions are met:

- No direct sunlight: The sun is the biggest uncontrolled IR emitter, which causes interferences to the sensors.
- Controlled light sources: Nowadays this is not such a large problem because lighting technology is moving to LED bulbs. However the use of an incandescent light source acts akin to the sunlight and therefore creates IR interferences.
- Large objects on IR-frame-based sensors decrease the sensor performance.

- PixelSense has to be accurately calibrated such that it does not detect hovering fingers as touches, which may occur given that it can perceive objects up to few centimetres above the screen.

3.1.3 Camera based Tabletops

Optical tracking camera-based tabletops are the most common techniques for "do it yourself" tangible tabletop surfaces. The tabletops that use this technique tend to be rear-projected surfaces with a camera underneath that see what is placed on their translucent surface [Figure 21]. The camera used on this kind of tabletops works in the infrared spectrum and is always accompanied by an infrared light source.

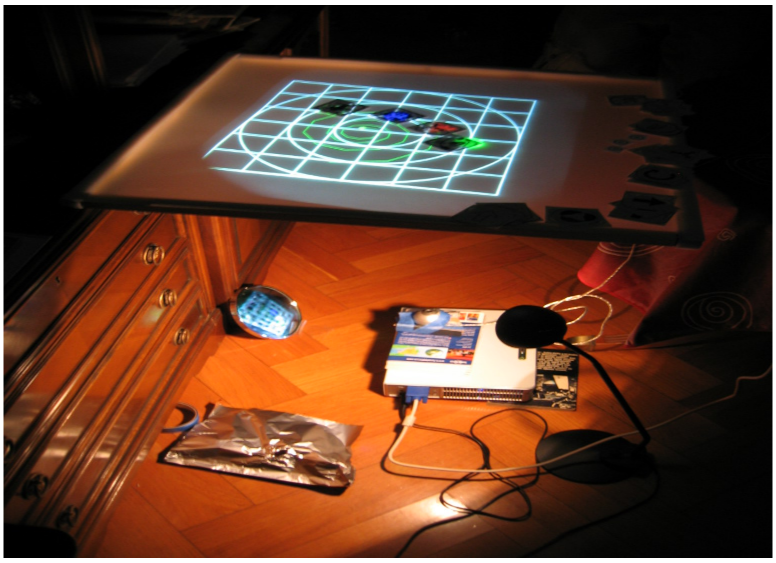


FIGURE 21: CONSTRUCTION OF A "DIY" OPTICAL TABLETOP.

The position and orientation of the infrared light determines what kind of objects the camera will be able to track. Two lighting techniques exist: "Frustrated Total Internal Reflection" (FTIR) and "Diffuse Illumination" (DI).

FTIR

Introduced by Hann (J. Y. Han, 2006; J. Han, 2005) in 2005, FTIR lighting is based on a optical total internal reflection within an interactive surface. Three layers with certain optical characteristics compose this surface: The top layer acts as a diffuser, which is made of a translucent material and is the projection support, the middle layer is made of rubber silicone and is used to change the light properties of the third layer (acrylic), which it is the medium where the infrared light is projected [Figure 22]. When a weight (finger, hand, object) is placed on the FTIR surface, the optical properties (width) of the middle layer changes and the light inside the acrylic escapes from the object thus allowing the infrared camera to track the position of the escaped IR light. Given the properties and the construction process of FTIR, cameras using this lighting method can only detect fingers (touches) and blobs on the surface. As a consequence, the detection of fiducial markers or any kind of object becomes impossible without sufficient weight to modify the baseline state of the middle layer.

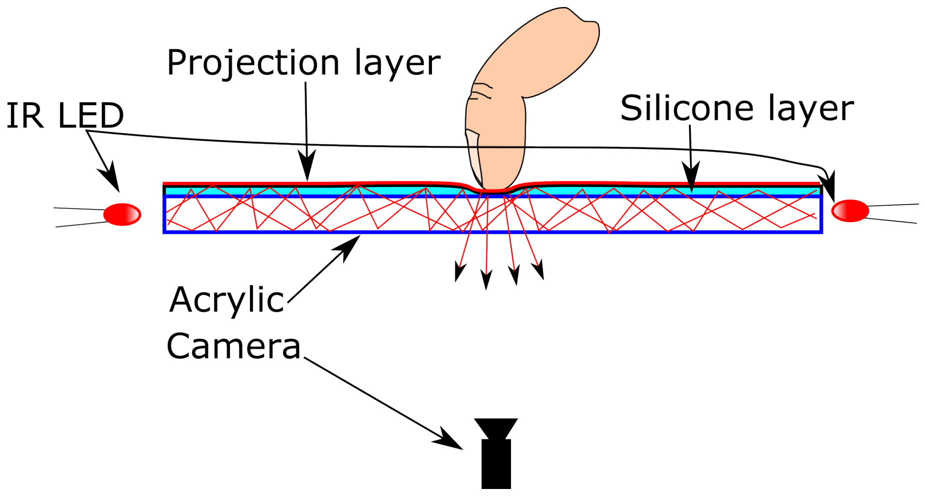


FIGURE 22: FTIR CONSTRUCTION LAYER SCHEMA.

Diffuse illumination

The hardware used in DI systems is similar to that used in FTIR. Both techniques use an infrared camera that tracks what is happening on the surface. The only difference is the IR light source. In Diffuse Illumination, the IR-lamps are placed outside the Acrylic surface, lighting it from underneath [Figure 23]. The significant advantage of this lighting technique is that all the objects are lighted equally, allowing the detection of fiducial markers, fingers, blobs and other objects on the surface. Upon comparing DI with FTIR, one can assume that DI is better and cheaper (less material needs to be used), but the reality is different: DI systems require a precise and concrete amount of Infrared light obtainable via tedious processes of camera adjustments (diaphragm, shutter, etc.), while the infrared light on FTIR surfaces is driven through the acrylic material, which is a better medium to control the light.

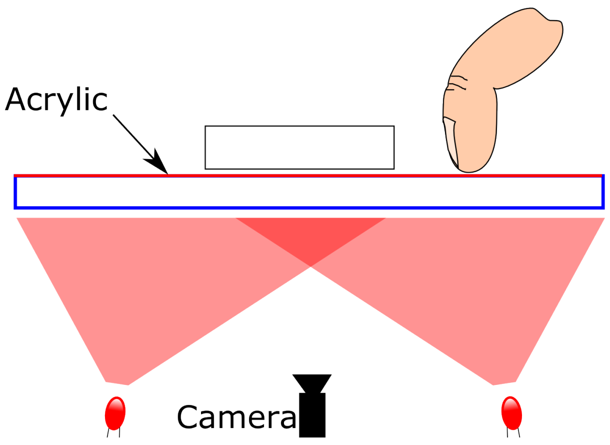


FIGURE 23: DI SURFACE SCHEMA

Rear-projected surfaces, such as camera-based surfaces, can be shaped in any way and the sensing area no longer has to be restricted to a square or a rectangular shape. This is the case of the Reactable (Sergi Jordà et al., 2005), a round shaped table that uses the DI lighting method to detect Tangibles and fingers. As it uses a camera to record what is happening on the surface, Reactable is a tall table (around 85 centimetres in height) that, therefore, permits the camera lens to encompass the entire surface. This is not the case of Microsoft surface 1.0, a rear-projected surface that uses DI but, instead of using a single camera, it employs an array of cameras to reduce its height.

Akin to IR-emitter/receiver-based solutions, camera based tabletops are affected by any uncontrolled source of light (sun, incandescent light, candles, etc.) but in the case of camera-based tabletops this interference is more significant. When a camera gets blinded because of a large amount of IR light, the tracking of the elements of the surface is impossible. However, in IR-receiver-emitter devices, blinding an IR receiver or a group of them, is not as decisive; the interface will work but some portions of the surface could stop detecting fingers and objects for a while.

3.1.4 Tangible Tabletop Technology briefing

In [Table 4], a comparison of the aforementioned tangible tabletop technologies is shown. In this table, the kinds of technology that can track fingers, tangibles and blobs are presented, along with the pros and cons of each type of technology.

Tabletop/technology	**Tracked elements**	**Pros**	**Cons**
Resistive	Single touch	Cheap. No interferences.	Poor image results because it is not fully transparent.
Capacitive	Multi-touch (Tangibles)[28]	No interferences (with normal screen's sizes). 100% transparent. Fast response.	Expensive. Passive objects need to be touched at all times. Reduced set of tangibles.
IR Frames	Multi-touch Tangibles [29] Blobs	Works well on large screens. Easy to attach to any screen.	Occlusion with large blocking objects. Reduced subset of "tangibles" composed of different object sizes and shapes. Affected by incandescent light (sun, light bulbs…).
IR-sensors behind the screen	Multi-touch Tangibles Blobs Markers	Thin form-factor. Same results as camera-based solutions but without tedious calibration processes.	Affected by incandescent light (sun, light bulbs…).
FTIR camera based	Multi-touch Blobs	Does not require tedious light source calibration processes. Difficult to get false positive touches.	Unless it uses a camera it does not track markers. Affected by incandescent light (sun, light bulbs…). Detection speed linked with the camera frame rate.
DI camera based	Multi-touch Blobs Shapes Markers	Cheap and easy to build. With the help of computer vision it can track anything visible by camera. All types of tabletop shape are adequate.	Tedious light calibration process. Affected by incandescent light (sun, light bulbs…). Detection speed

[28] A reduced subset of them based on passive objects that have to be always in contact with the user's hand.
[29] Based on shape recognition.

			linked to the camera frame rate.

TABLE 4: TANGIBLE TABLETOPS SURFACES TECHNOLOGY COMPARISON

Our objective for the new expanded tabletop interface is to track objects and hands beyond the surface, while maintaining the common surface interaction. By checking Table 4, we can discard: resistive layers, IR-Frames and FTIR camera-based tabletops, because these cannot track markers or tangibles. Some of these presented technologies can be used for sensing objects not only on the surface but also above it up to few centimetres far from the surface. Before moving to 3D tabletops their capabilities should be mentioned:

- Capacitive screens can track fingers (or hands) beyond the screen: up to 5 cm before being unstable. Taking advantage of this feature exists commercial capacitive tabletop surfaces that can obtain the finger direction based on the hand position estimation. This capability is not so common among recent devices because the programmer should have to access to the controller low level data, which is restricted by hardware developers.
- IR-receiver matrices behind the screen: as commented before, the large array of IR sensors acts as a "surface" camera sensor pointing beyond the surface. The problem in this case however, is that this "pseudo-camera" does not have optics or lenses to focus objects beyond the display: they become blurred at a few cm above the surface. Again, like in capacitive screens, this feature is used to calculate touches' orientation.
- Camera-based DI: DI has a camera pointing towards the surface and lightens (with an IR-light) anything that is placed on it. This technology solution would be the best candidate to track objects above the screen (in contrast to IR matrix based screens; the camera of the DI can be focused) but has the disadvantage that the surface is made of a diffused material through which the camera cannot see objects.

3.2 Tangible 3D Tabletop recent solutions

This section makes an overview of some existing 3D tabletop systems that can track hands, fingers or tangibles beyond the screen or are able to produce visual feedback above the surface. The presented tabletops and tracking or feedback methods are divided in the following categories:

- Tabletops with feedback beyond the screen.
- Tracking methods beyond the display.
- SecondLight[30]

3.2.1 Tabletops with feedback beyond the screen

Providing feedback "on-the-air" or beyond the screen it is not an easy task because the user has to perceive shapes, tactile stimulus, or visualize images onto a non-persistent medium such as the air. The most extended practice in this area is the use of augmented tangibles. These tabletops use objects with certain characteristics for modifying their colour, images or shapes.

Lumino (Baudisch et al., 2010) is a DI rear-projected tabletop that uses special tangibles called Luminos. These tangibles are made of different structures of translucent fiberglass and opaque patterns that can be stacked on the table's surface [Figure 24]. Thanks to the translucent fibreglass, tangibles can transport the projection from the surface to the tangible placed at the top of the stack and vice versa. In this manner, a unique pattern can be generated from the top downwards containing the information of all the staked tangibles. This solution works well for marker composition through the process of stacking tangibles. However, when the stack becomes bigger, there is less space to represent the visual feedback.

[30] This is a 3D tabletop surface and not a category, but we keep it separated because of its special affordances.

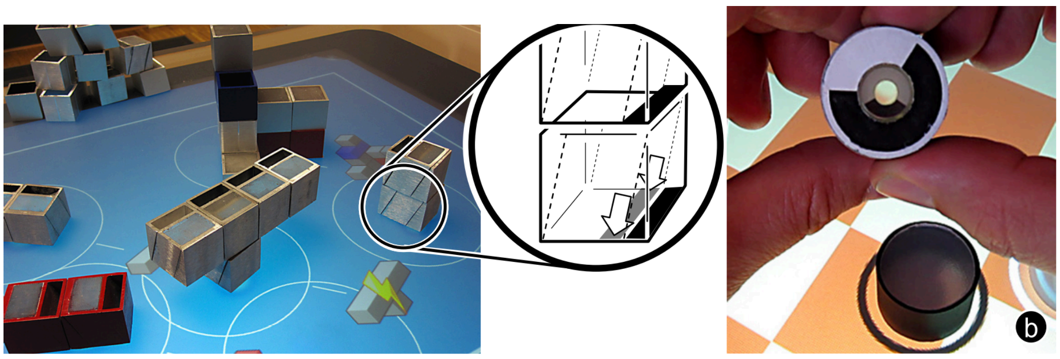


FIGURE 24: LUMINOUS TANGIBLES[31]

Another approach for representing images on tangibles is that proposed in (Dalsgaard & Halskov, 2012), whereby the authors combine **tabletop interaction and 3D projection mapping** by using 3 projectors: one underneath the table and the other two in the left and right sides above the table. The tangibles used are made of a white projection support material and adopt different volumes and three-dimensional shapes. At the bottom, there is a marker that identifies each tangible and extracts the position and rotation angle on the surface. With this information, the system can project data covering the entire tangible similar to a 3D mapping of a real-time generated texture. This method of 3D mapping onto tangibles adds some advantages comparing to the Lumino. It increases the resolution of the images on the tangibles and accepts tangibles of all shapes. On counterpart, using projectors above the surface to augment the tangibles may cause some tangibles to occlude themselves and produce black areas on some parts of the tangible objects.

The project **Back to the sandbox** (Beckhaus, Schröder-Kroll, & Berghoff, 2008) introduces a new paradigm on tangible interaction. Instead of using solid shapes, this tabletop uses sand that can adopt any shape as manipulated by the user [Figure 25, left]. A camera tracks the terrain accidents on the sand, while a projector generates the visual representations of the data onto the sand according to the pattern created by the user. **Physical telepresence** (Leithinger, Follmer, Olwal, & Ishii, 2014) follows a similar line. However, instead of sand, this table uses an array of pneumatic-driven tokens that can adopt shapes

[31] Pictures extracted from (Baudisch et al., 2010)

without being manipulated by the user [Figure 25, right]. This permits the surface to adopt any shape according to the running application.

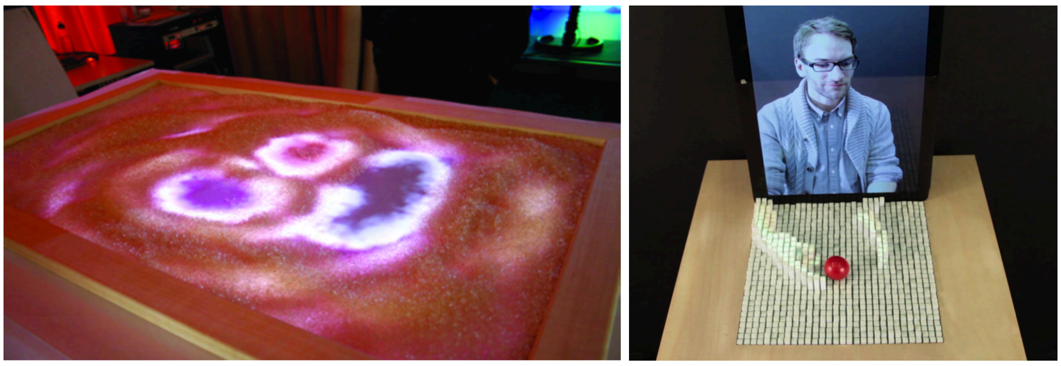


FIGURE 25: BACK TO THE SANDBOX[32] (LEFT); PHYSICAL TELEPRESENCE[33] (RIGHT).

Augmented tangibles did not introduce any change on tracking gestures in the air, but they represent another paradigm of changeable interfaces whereby the object itself has 3D components and can provide haptic feedback, which is very hard to reproduce in "in-the-air" interaction.

3.2.2 TRACKING METHODS BEYOND THE DISPLAY

Tangible and tabletop 3D interfaces use different approaches to track objects beyond the surface. They can be based on a camera (or a depth camera) solution or based on electronic tracking. These methods can track different objects or fingers beyond the screen in a 3D space but while the input can be 3D, their output will still be 2D because they cannot provide any "in-the-air" feedback.

Camera-based 3D tracking

Most of the camera-based 3D tabletop devices use depth cameras such as Kinect, stereoscopic systems of cameras like LeapMotion or single-cameras tracking markers with special 3D affordances. Camera-based solutions are the most immediate and inexpensive technologies. Yet, they are hard to use because they require an entire room to adjunct the side or ceiling cameras, control the room's light for not interfere with the frame acquisition and implies large calibration processes.

[32] Picture extracted from (Beckhaus et al., 2008)
[33] Picture extracted from (Leithinger et al., 2014)

Created by Prime Sense and released in the mass market by Microsoft with its Xbox console in 2010, **Kinect** was introduced as a non-invasive full-body motion tracker. This device is a set of two cameras, an infrared projector and an array of microphones [Figure 26,left]. It uses the structured light method for generating depth maps. This method consists of projecting a complex grid of IR-dots to form a known pattern and reading them using the infrared camera. This is achieved by computing the deformation of the grid from which the sensor can infer the depth position of each dot. Kinect, akin to all IR-camera based solutions, is vulnerable to direct sunlight. Furthermore, the optics of the camera are built to track elements placed between 1 to 5 meters far from the camera, making it very hard to properly track hands with the fingers at close distances. Although other structured light solutions exist, Kinect is still the most popular depth camera because it is easy to access and is cheaper.

Designed for hand tracking in front of a screen, **Leap motion**[34] is a tiny stereoscopic camera device for hand and finger interaction in the air [Figure 26,right]. It has been designed for being used in front of a screen and converting it on a vertical surface that supports multi-tactile and hand air-gestures. Leap motion has a considerable high frame rate, between 120-130 FPS, due to its internal processor and USB 3.0, which is more than twice the speed of any camera-based solution working over a USB 2.0 or Firewire.

[34] https://www.leapmotion.com/

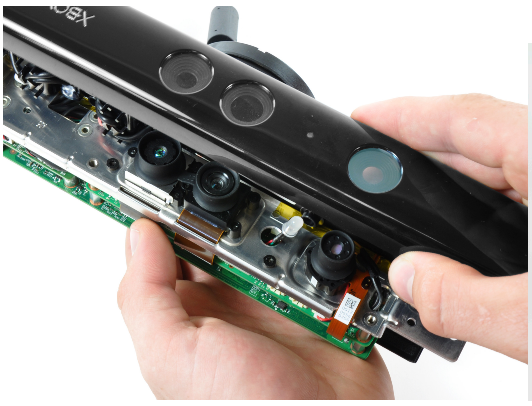
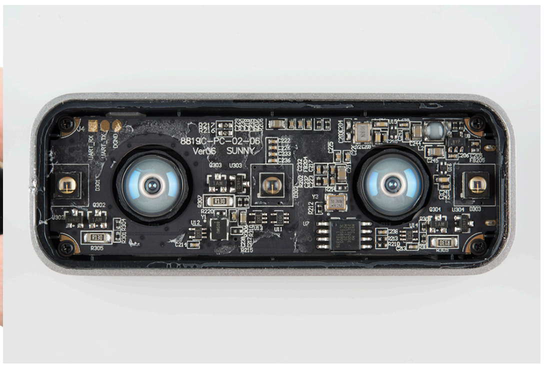



FIGURE 26: DETAILS OF A KINECT 1.0 DISSASEMBLED UNIT[35] SHOWING THE IR EMMITER, RGB CAMERA AND IR CAMERA (LEFT); DETAILS OF A LEAP MOTION DISSASEMBLED UNIT SHOWING THE TWO CAMERAS FOR STEREOSCOPIC VISION AND THRE IR-LEDS FOR LIGHTING[36] (RIGHT)

Normal cameras are also used in 3D tangible interaction; they are mostly used for in-the-air hand and marker detection and cannot detect depth data (Z-coordinates) by themselves. In (Epps et al., 2006), a single ceiling-mounted camera is used for hand shape recognition and manual interaction on any surface: physical or virtual (screen) desktop. Following a similar setup, but using a multi-tactile screen, (Carreira & Peixoto, 2007) extracted the hand contour and defined a set of hand postures for widget manipulation and orientation.

It is not possible to use a single camera for tracking objects in a 3D space unless the objects to be tracked-down are known and can be pose-estimated. On the other hand, Kinect-based tabletops are the most common among 3D tabletops (De Araújo et al., 2013, 2012; Hilliges et al., 2012; Huppmann et al., 2012; Leithinger et al., 2014; Sutcliffe et al., 2013). Kinect camera does not need special markers or known objects because it can extract the 3D data from any tangible or body without markers.

Using a different approach with motion capture methods [Figure 27,left], (Marquardt et al., 2011) explored **the continuous interaction space** by using sphere markers for a 3D motion tracking tabletop interface. The problem of this setup is that any element used has to be properly filled with these IR-reflective spheres, including the user's hand and fingers. Another project that uses markers

[35] Picture extracted from: http://www.embedded-vision.com
[36] Picture extracted from: https://learn.sparkfun.com/tutorials/leap-motion-teardown

placed on body parts, although the tabletop integrated camera rather than motion capture methods is **"what causes that touch"** (Marquardt et al., 2010). The authors used a Microsoft surface 1.0 for tabletop interaction and a glove filled with the surface's markers for hand pose estimation. Despite that this setup cannot track hands in the air, it can detect with which part of the hand the user is touching the surface [Figure 27, right].

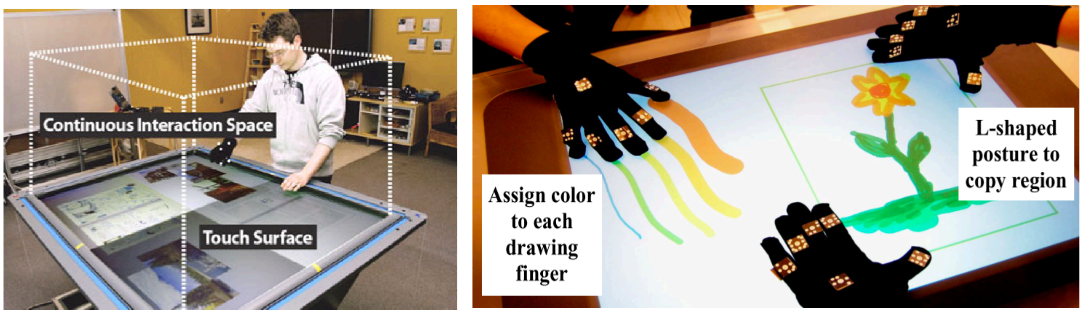

Continuous Interaction Space
Touch Surface
Assign color to each drawing finger
L-shaped posture to copy region

FIGURE 27: THE CONTINUOUS INTERACTION SPACE USING MOTION CAPTURE[37] (LEFT); "WHAT CAUSES THAT TOUC?" USING MARKERS ABOVE THE MS. SURFACE 1.0[38] (RIGHT)

Although all these methods are valid and have been proven to work as a 3D tabletop interface, they all require particular components such as gloves, invasive tracking devices or an external camera device placed at about one meter distance from the tabletop, which hinders the setup process and, because of the camera, blocks one side of the interaction surface.

Electronic 3D tracking

Electronic trackers for 3D tangible devices do not differ excessively from those used in 2D interfaces. They are based on the same principle of capacitive displays, IR frames or springs and threads connected to variable resistors that at the end results on a resistive linear value.

Z-Touch (Takeoka, 2010) uses 8 layers of the aforementioned IR-frames stacked one above the other on the surface [Figure 28,left]. The main difference to the IR-frames for traditional surface interaction is that Z-Touch uses an infrared camera instead of filling the frame with IR-sensors. Each IR-layer is triggered from top to bottom and a camera under the surface collects the frames

[37] Picture extracted from (Marquardt et al., 2011)
[38] Picture extracted from (Marquardt et al., 2010)

each time an IR-frame is triggered. For each triggered IR-layer, the system processes a 2D position map of any object inside the frame. When vertical aligning eight of these frames, the system only has to reconstruct the 2D “slices” to build a 3D model (depth map [Figure 28,right]) of the objects that are touching or are placed above the surface.

The use of stacked IR frames produces a low-range distance tracker, forcing the user to make in-the-air gestures up to where the IR-frame is placed at the top of the stack. Another issue of this system is that they cannot track any kind of markers above the surface; in fact, these IR-frames could be seen as different FTIR layers where the light is projected through the air instead of through the surface.

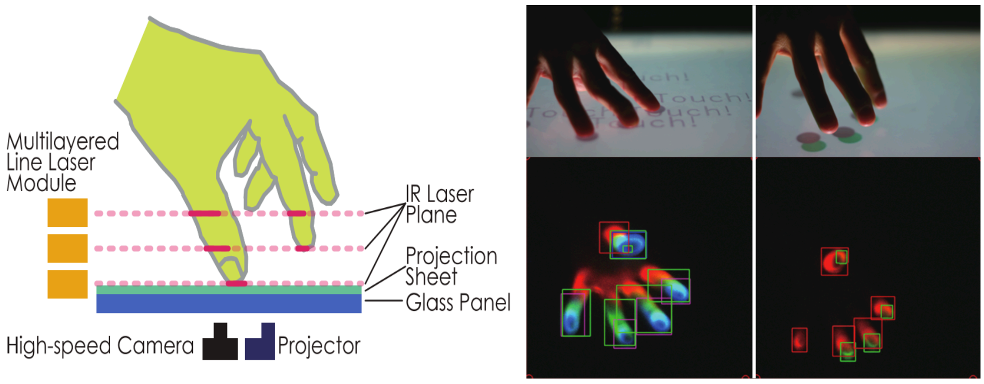


FIGURE 28: Z-TOUCH: DIAGRAM OF THE IR LAYERS (LEFT); DEPTH MAP OF HAND GESTURES (RIGHT)[39].

Mock-up builder (De Araújo et al., 2013) is a semi-immersive environment for conceptual designing that allows virtual mock-ups to be created via the use of 3D gestures. This tabletop uses a stereoscopic multi-touch display, a Kinect camera (introduced in the next sub-section) for head, body and arms tracking, an IR-stereo Emitter for head tracking and two Gametracks for finger detection [Figure 29,left]. A Gametrack is a complement of a golfing game for the Playstation2; it uses a nylon tether (red line in [Figure 29, right]) that is spring-tensioned. A geared turn-potentiometer senses the distance that the tether is pulled out of the Gametrack. The tether is threaded through a 2-axis analogue joystick. Combining one radial measurement and two angles, it is possible to

[39] Figure and picture extracted from (Takeoka, 2010)

obtain the absolute position of the end of the tether, anywhere within a conical region below the Gametrack.

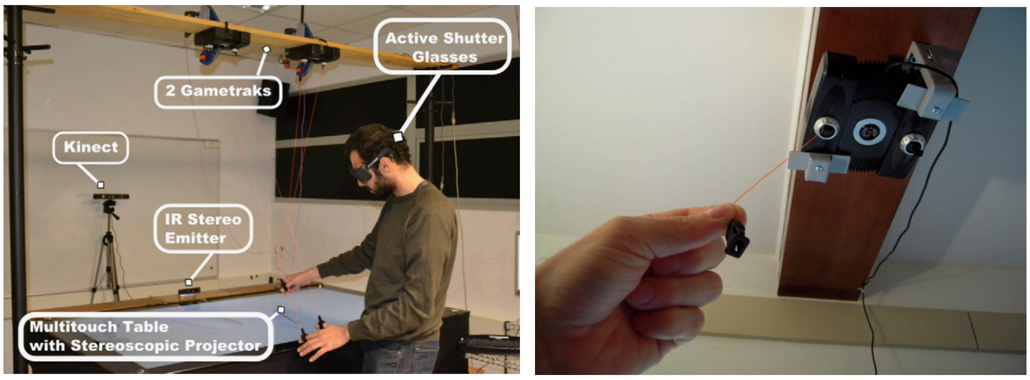


FIGURE 29: MOCK-UP BUILDER[40] (LEFT); A GAMETRACK DEVICE (RIGHT).

Mock-up builder uses four 3D tracking devices for only hand and finger interaction, including redundant ones such as Kinect and Gametracks. It also uses invasive devices attached to the user such as the ends of the Gametrack's tethers attached to the user's finger and the shuttered glasses for the stereoscopic display. Invasive devices and external cameras such as Kinect, combined with a multi-touch device, can function well in lab conditions but as more technology is added to an interface, the calibration process becomes more complicated. In addition, crossing arms in the air with the Gametrack as well as multiuser interaction is impossible because the physical limitation of the Gametracks.

3.2.3 SecondLight

SecondLight (SL) is a tabletop interface from Microsoft Research (MSR) developed by the group lead by Shahram Izadi (Izadi et al., 2008). This table has two cameras and two projectors underneath that point to a translucent element placed at the table's surface, similar to many other rear-projected tables. What makes this table different from the others is its surface: the SL's surface is made of a special material called polymer-stabilized cholesteric-liquid crystal (PSCT-LC) that is switched from the scattering focal conic texture to the transparent homeotropic texture when a sufficiently high voltage is applied and remains in the homeotropic texture when the applied voltage is removed (Li, 2012). Due to

[40] Picture extracted from (De Araújo et al., 2013)

this special liquid crystal, SL can switch its surface from a completely transparent crystal (clear states) to a diffused one (diffuse states) by driving current through it (Figure 30).

SecondLight (SL) affordances are rather different from those of a conventional tangible tabletop surface, since its camera can see objects both on and above the surface, while its projector can display images on the surface as well as on any object above it.

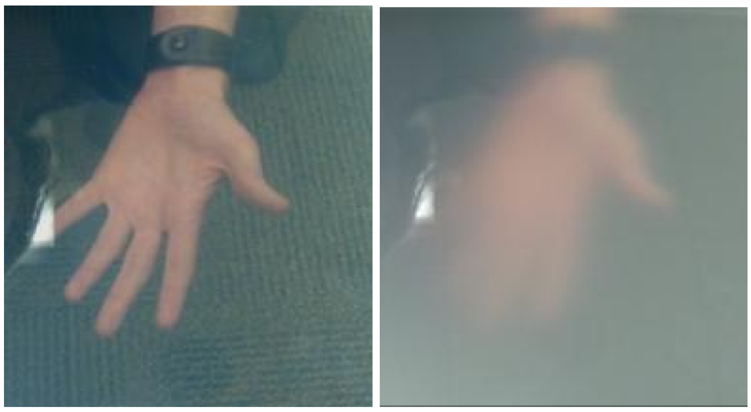


FIGURE 30: PSCT-LC STATES: CLEAR (LEFT) AND DIFFUSE (RIGHT)

SL's first iteration uses two projectors, each with a FLC shutter and two IR-cameras pointing the surface from below (Izadi et al., 2008). FLC shutters are synchronized with the flickering surface therefore, while one projector projects images on the surface, the other can project images on the objects above the surface. The cameras are adjusted at different focal lengths, one pointing towards the surface and the other above the surface. These cameras work under the infrared spectrum using the light emitted by a FTIR lighted surface.

SecondLight demos

SecondLight has a strong potential in tracking and visualizing in a 3D space. The demos presented by Microsoft are very simple but effective in showing the potential of this technology.

As it is based on a common rear-projected tabletop interface, on-surface multi-touch interaction applications are the most immediate to be used, it can detect multiple touch inputs and is able to recognise shapes "in the air". SecondLight allows the combination of traditional surface interaction with more advanced features that extend interaction beyond the surface.

Its capacity to project beyond the surface was the most striking demo of this tabletop device. By using translucent sheets of diffuse films, it is possible to project one image on the table's surface and another completely different image on the translucent film. To show this feature MRS built a gesture-based (pinch, drag, rotate and zoom) photo application for the SecondLight that showed some images on the surface and, by placing a translucent sheet on these images, complementary information were shown on the external sheets as they where magic lenses (Bier et al., 1993) (i.e. a Wireframe view for 3d model (Figure 31, left), constellations on a sky photo or a Wikipedia article on an image).

MSR also explored the capacity of this tabletop by tracking two kinds of external movable surfaces: one made of translucent material with a pair of IR-reflective stickers placed on each side, and the second one made of acrylic material with a built-in IR-LED and a battery for the FTIR illumination of the movable surface. While the external surface with IR-reflective stickers was detected and projected in the air, it could be blended together, generating deformations on the projected images. On the FTIR lighted external surface, having the property of refract IR-light when a finger was placed on it, this touch event was used by the system as an in-the-air touch on a movable tactile screen (Figure 31, right).

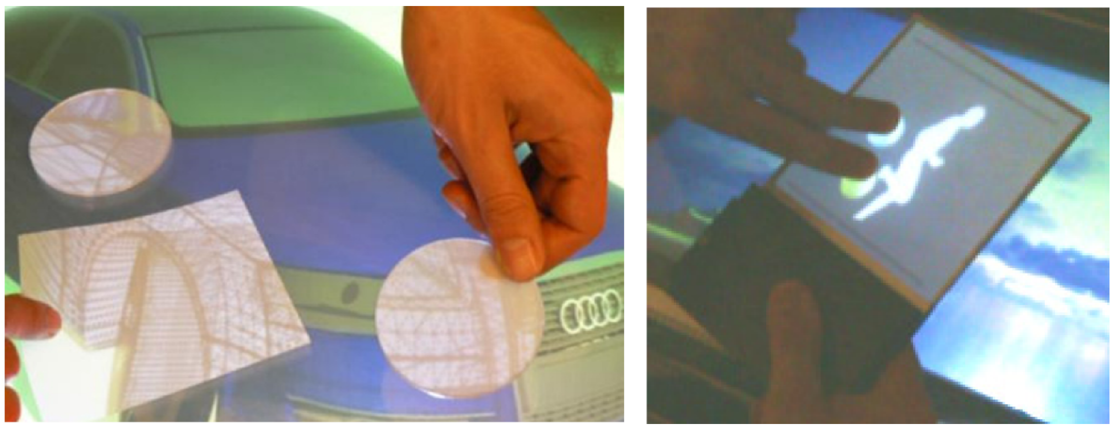


FIGURE 31 SECONDLIGHT ABOVE THE SURFACE DEMOS.

3.2.4 3D TABLETOP TECHNOLOGY BRIEFING

Table 5, summarizes all the aforementioned tracking methods and points out their advantages and disadvantages. Notice that some are types of technology while others are complete tabletop systems.

Device	Function	Pros/cons
Projected tangibles	Visual feedback beyond the screen	+ Visual and physical feedback due to the projection and the physical structure of the tangible. - Does not track movements beyond the screen.
Stereoscopic glasses or devices	Visual feedback beyond the screen	+ Visual feedback beyond the screen. - Loses any physical feedback, which interferes with the direct manipulation concept.
Shape changer tabletop surfaces	Can adopt any real-time shape on the surface by using malleable elements or actuated tokens	+ Visual and tactile feedback beyond the screen. - Very limited possibilities as a multi-purpose device.
Stack of IR-frames	Tracking objects and user input beyond the surface	+ Easy to be adapted to any projected tabletop surface. - Short distance-tracked elements. - Cannot track markers.
Gametrack	Object tracking with relative tracking from the device.	+ Easy to track direction and object accelerations, faster than any camera based solution. - Attached tether to any tracked object. - Only two elements per device. - Does not provide a rotation angle.
Motion capture	Tracking objects and hands in the 3D space with pose estimation.	+ Tracks "in-the-air" rotation angles. - Any tracked object has to be attached to a special volumetric marker. - External camera ceiling structure, making transportation of the device.very hard.
Kinect	Tracking objects using a depth camera	+ Does not need any special marker attached in order to track objects. + By using post-processing techniques (Newcombe et al., 2011) it is easy to estimate the object's position and orientation. - It cannot track elements that are closer than 1 meter because the camera lenses and IR-projection. - In the recent official version for developers, it is not possible to track hands or fingers. - It needs to be placed at a table's side or

		on the ceiling.
Leap Motion	Stereoscopic all-in-one device for hand and finger tracking	+ Fast tracking with a camera frame rate of 130 fps. - It cannot detect objects or markers.
SecondLight	Surface and above the surface tracking with a double projection system on the surface and beyond it	+ All-in-one tabletop device + Surface[41] interaction ready + Projects beyond the surface + Tracks movable displays above the surface + Tracking fingers on movable displays. - Cannot track markers - Cannot track hands or fingers beyond the surface - Cannot distinguish between two external displays.

TABLE 5: 3D TABLETOP TRACKING AND FEEDBACK METHODS.

When comparing all the aforementioned 3D tangible tabletop devices and considering the tabletop that we were searching for to expand tangible tabletops to the third dimension, the one that met most of our requirements was SecondLight because it can provide visual feedback above the display and has the potential to see beyond the surface. However, although SecondLight seems a good candidate, some modifications and additions should be added in order to reach our objectives of:

- Tracking hands and fingers beyond the surface.
- Tracking 6DoF markers on and above the surface.
- Identifying two or more different external surfaces.

As mentioned in the Motivation part, Microsoft Research in Cambridge has funded this thesis. It is due to this collaboration that we were able to access a new SecondLight unit. After some modifications, the three aforementioned points could be integrated into the SecondLight capacities. However, before the SecondLight arrived to our lab, others alternatives were explored in an attempt to extend traditional tangible tabletop surfaces.

[41] Only fingers, shapes and contours, marker detection need the use of DI illumination.

3.3 Prototyping a 3D tabletop interface

Before getting access to our SL unit, we explore third dimension on a traditional tabletop interface by developing and using external objects with special affordances. Aside of the aforementioned TableGestures applications suggesting a 3D tabletop interaction we develop an accelerometer-powered tangible that can track “on-the-air” gestures.

3.3.1 Tangible 3D tetris and the Active Cube

Tangible 3D Tetris or Tangible Blockout was part of a tabletop interaction input exploration to be applied to control games developed on the same interface as Reactable, a round shaped tabletop designed for 2D interaction. This project uses a Plexiglas cube with different markers placed on each face such that the system can detect which face of the cube is placed on the table. The application starts with the representation of a virtual space drawn in the middle of the table as a cubic box that is expanded below the surface. Near the virtual space, a three-dimensional representation of a Tetris figure is also shown that suggests the next 3D volume candidate to be dropped inside the cubic box [Figure 32]. When the user places the tangible in any place on the table that is not the virtual cubic area, the suggested candidate is reoriented towards the cube’s position. The aim of this game, like in the traditional Tetris, is to resist as long as possible without flooding the cubic virtual space with the different volumetric figures, which disappear when a row (volumetric) is fully completed. The user can only interact with the Plexiglas cube by rotating it in the air and placing it on the desired column for dropping it inside the cubic volume [Figure 32].

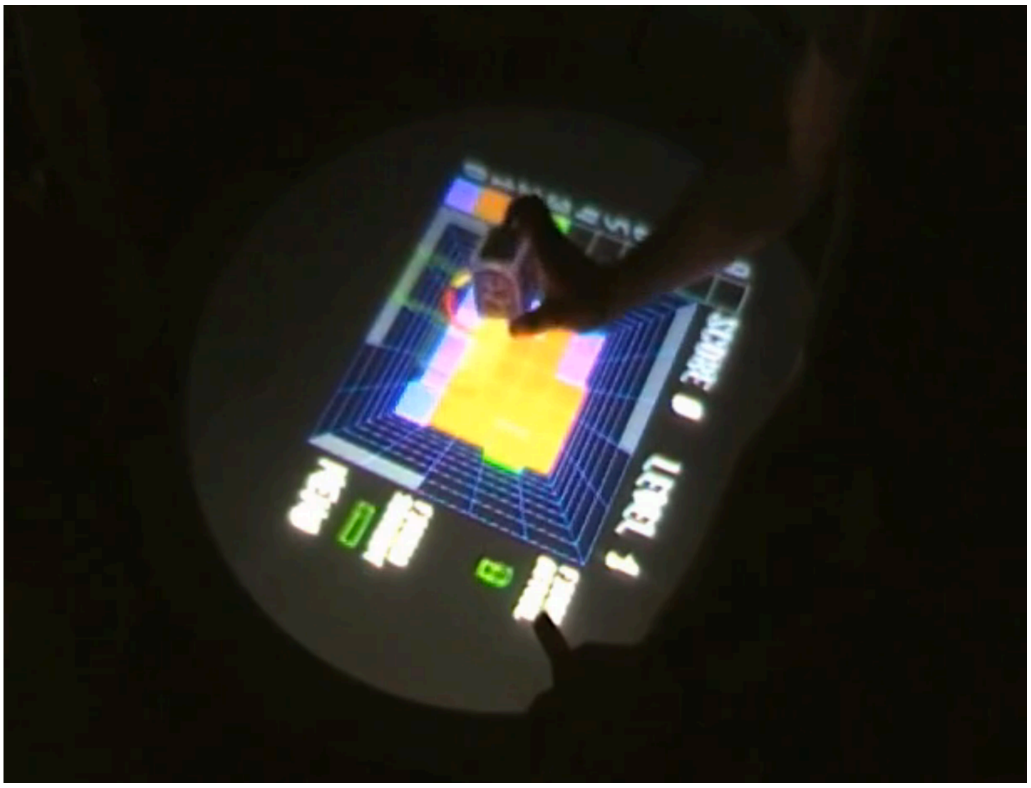


FIGURE 32: TANGIBLE BLOCKOUT BY ROGER LOPEZ GARCIA.

In this application, the user can only perceive the rotation of the virtual volume by the physical characteristics of the cube and by a written "UP" on one of the markers' side. When the Plexiglas cube is "in the air", the system is not able to detect the cube's position and orientation unless it is placed on the table, for that reason, any cube orientation's visual feedback is reported or shown on the screen. While searching for a way to tell the system what the position of the cube is when it is in the air, the Active Cube was developed.

Active Cube is a Plexiglas Cube with a dot-led matrix screen at one side that can show any pattern depending on the context of the application and a fiducial marker on the opposite side for surface interaction. It is filled with an Arduino[42] Nano powered with a battery, a Bluetooth to a serial antenna and an accelerometer [Figure 33].

[42] www.arduino.cc

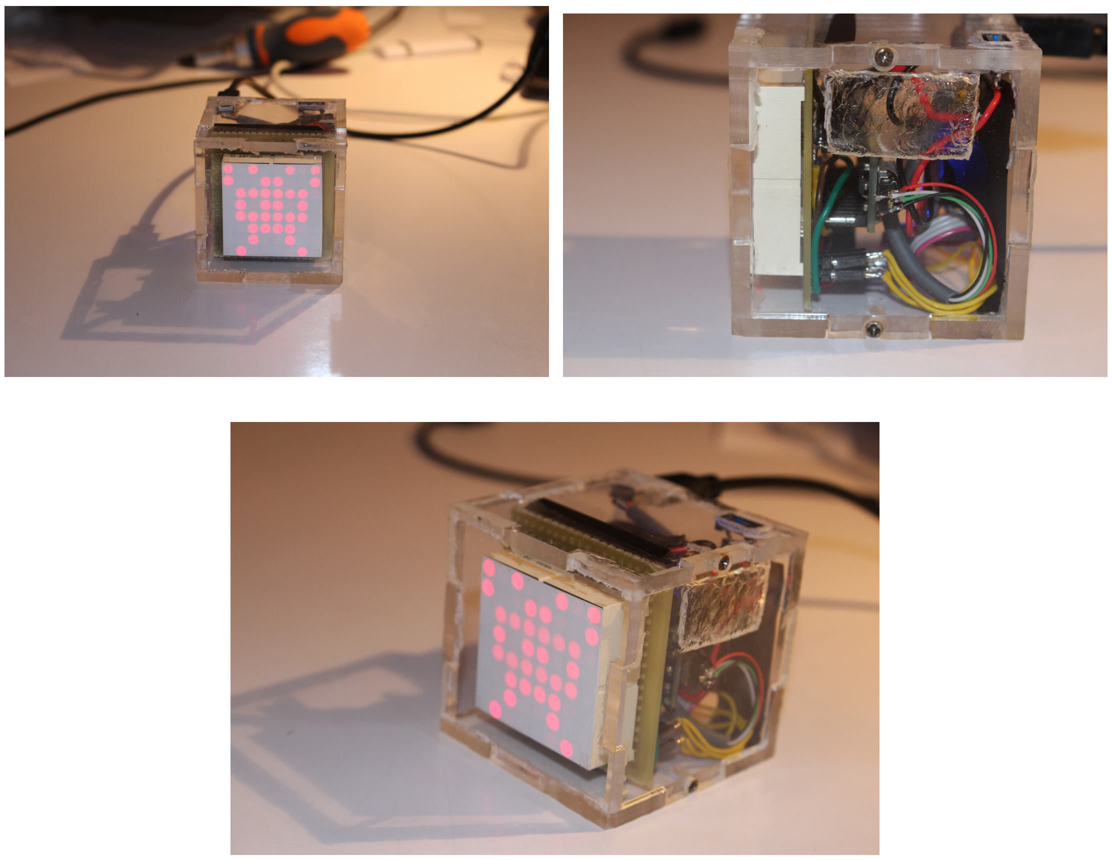


FIGURE 33 ACTIVE CUBE.

Due to the accelerometer, the cube knows, at all times, its orientation and in-the-air accelerometer-based gestures such as those that can be realized with a Wiimote (Schlömer et al., 2008). The data gathered from the cube is sent directly to the computer via Bluetooth to display real-time visual feedback.

The interaction with this cube in 3D Tetris was done in the following way: the graphics and interaction areas of the 3D Tetris were identical but, instead of using a Plexiglas cube with markers, the Active Cube was employed. When the user spun the cube in the air, the virtual representation of the volumetric Tetris figure spun in the same way, thus providing a real-time visual feedback of the 3D interaction on the surface. When the desired position was found, it could be selected by simulating a drop movement in the air. However, as the tabletop was not aware of the cube's absolute position, the user had to place it on the surface in order to select the desired position on the volumetric space to drop the Tetris volume.

This cube provides us with a system that can track tangible interaction on the surface and interaction in a second state above the surface without tracking position and height; only angles.

This two-state interaction was far from satisfying our need to expand tangible tabletop interaction beyond the display and was discarded as an option because it cannot track continuous interaction beyond the surface, neither on tangibles nor on hands and fingers.

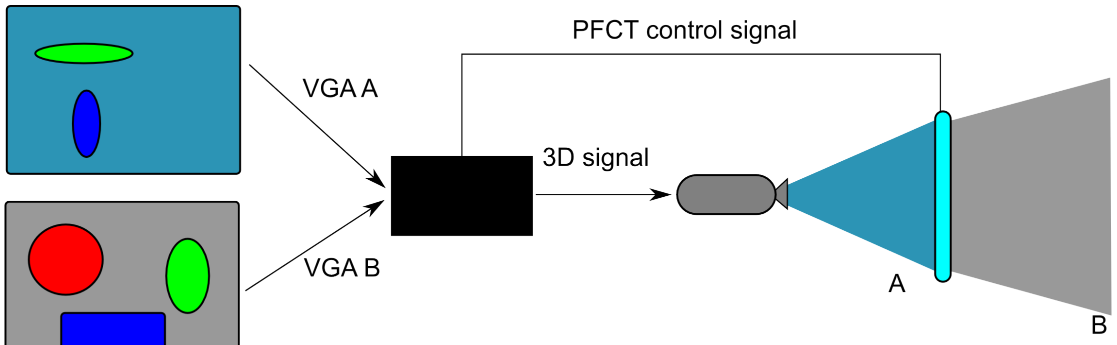
3.3.2 Modifying the SecondLight And physical constrains

The unit that we received differs from the first SecondLight (SL) prototype presented in (Izadi et al., 2008). Some modifications were applied at Microsoft Research before sending the prototype to our lab because of the premature degradation of some components or the elevated cost of its different parts, the most significant changes were applied because these issues:

- Active shutters mounted on two projectors at the first SL tabletop, after a continuous period of use, were losing their capacity of being transparent and the projected images were slowly disappearing.
- The first SL prototype used two firewire cameras implying a precise shutter control and increasing considerably the cost of the table.

For avoiding the issue with the active shutters, the active shutters where removed and instead of using two projectors, the delivered SL have one that is in charge of projecting images on the surface and beyond the surface. On the camera side, instead of using two cameras, one focusing on the surface and the other beyond the surface, now it comes with only one firewire camera pointing towards the surface and beyond, resulting in more complicated tracking processes as shown in (Chapter 4).

Projecting on and beyond the surface with only one projector is not an easy task. The projector should know what to project in clear and diffuse states. Instead of modifying the projector, the new SL solves this problem by using a commercial 3D projector[43]. In SL, a special graphics proxy card [Figure 34] has been built for the task of merging two different VGA signals from the computer and generating a "3D VGA" output for the projector. Syncing the switchable surface at 120Hz with the generated 3D video signal, the effect of having two different projections is accomplished: one on the surface and another above.



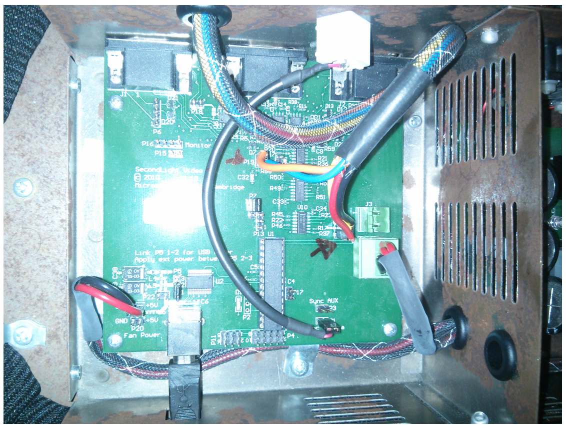


FIGURE 34: (TOP) SCHEMA OF THE SL WORKFLOW; (BOTTOM) DETAIL OF THE VGA PROXY INSIDE THE SECONDLIGHT.

Besides of the extracted camera and projector, the received SL unit was provided with only FTIR illumination for lighting elements on the surface and detect external built-in LED displays, which resulted in a tabletop that could track:

- Fingers on the surface.
- Markers on the surface.
- External or portable displays.

[43] Three-dimensional projectors can display two different frames, one after the other, sixty times per second and combined with shuttered head-mounted lenses the user can perceive two different images, one per each eye, perceiving 3D projections.

Hardware and physical constraints

The SL model that we have used in this thesis is equipped with an IEEE 1394 (FireWire) of 640x480 pixels-resolution black and white camera; fitted with an infrared filter in order to avoid the feedback that could be caused by tracking the images projected by the projector. It also has a built-in computer, a DELL workstation with an Intel Xeon 2.53Mhz processor and 4GB of RAM running Windows7. The overall hardware is fitted with the PFCT controller and VGA proxy card. It is placed in a metal case, at the bottom of the table, with its respective airing holes and fans [Figure 35, left].

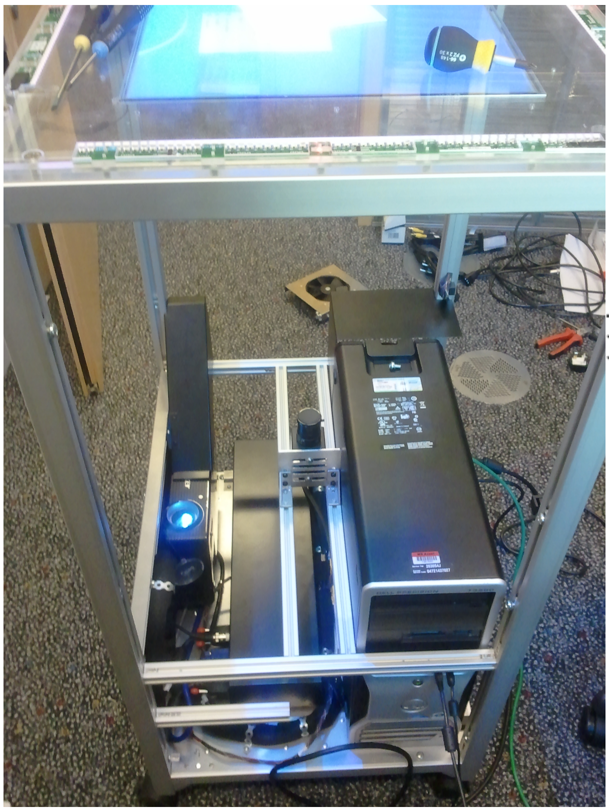
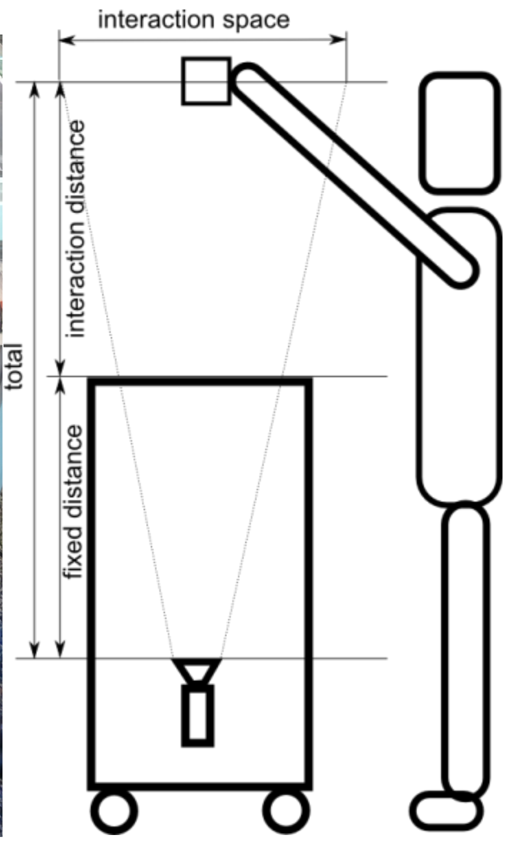



FIGURE 35 SECONDLIGHT HARDWARE PLACEMENT (LEFT) AND PHYSICAL CONSTRAINTS (RIGHT).

SL is 70 centimetres tall, with a surface that is not much bigger than a twenty-inch screen. It has the camera at the bottom, thereby the distance from the camera to any element placed on the surface is 70 centimetres, while if the element is above the surface, this distance is increased up to 170 centimetres [Figure 35, right].

Instead of modifying the tabletop physical sizes for getting a bigger interaction surface area, we decided to explore the interaction in the air by enhancing the Z-range of the tracked elements above the surface and leaving the physical modifications as a future work. **Our objective with this table** is to track different elements in order to distinguish whether they are on the surface or above as well as to track the continuous transition between the two states. The elements that we want to track, as described in the Motivation chapter, are:

- Fingers on the surface (SL has it by default)
- Tangibles on the surface (Theoretically SL cannot support it because FTIR)
- Hands above the surface (There is no IR-light beyond the surface for hand tracking)
- 6 DoF tangibles on and above the surface (same problem with FTIR and IR-light beyond the surface).

SL Modifications

As commented before in this chapter, FTIR illumination tabletops can track fingers and blobs on the surface, although the received SL unit with FTIR can also track markers on the surface, up to 3 cm, as well as fingers. This is possible because the material used for the surface where the IR-light is projected does not contain a malleable layer such as silicone but is made by Plexiglas EndLighten[44] material. EndLighten is an acrylic transparent material filled with small reflective crystals that do not influence the projection that passes through it but affect the way that the IR light crosses the surface. These crystals reflect the IR light to any side of the surface and act similar to a short-distance DI tabletop [Figure 36].

By using the EndLighten material, SL at clear surface states can track objects and fingers up to 3 centimetres distance before IR-Light becomes too weak, but, on the other hand it could project up to one meter distance without losing focus. To enhance the tracking distance, we added some extra lights around the table's

[44] http://www.plexiglas.net/product/plexiglas/en/products/solid-sheets/endlighten

surface pointing beyond it[45] to light elements located up to one meter distance from the surface, akin to the projection [Figure 37].

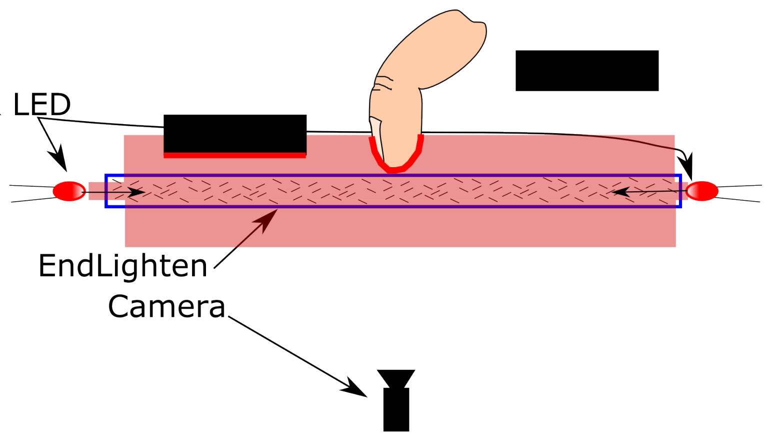


FIGURE 36: FTIR WITH ENDLIGHTEN MATERIAL.

Upon checking the frames above the surface, we noticed that they were blurred regardless of the distance beyond the surface. We tried to focus the camera, and change the position of the IR-light that points beyond the table but none of these changes produced the searched results. Finally, we changed the EndLighten surface, replacing it with an acrylic one, and, consequently, the camera was able to track well-defined elements beyond the surface. The EndLighten material reflexion properties affected the images tracked by the camera but not the projection emitted by the projector.

[45] By adding IR-LEDs below the surface, the camera cannot track elements because the IR-light bounces on the switchable screen.

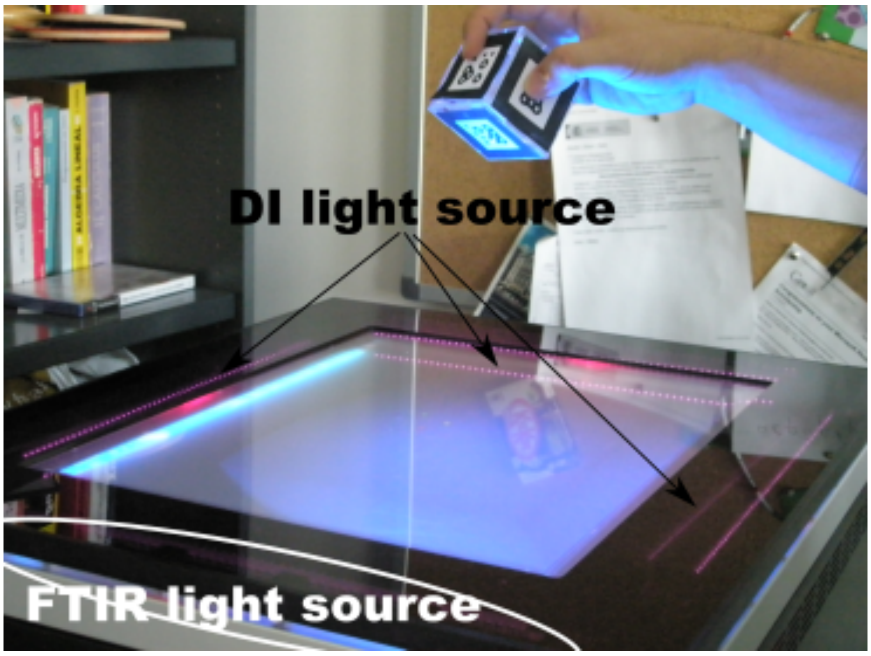


FIGURE 37: FTIR ILLUMINATION ON THE SURFACE AND DI ILLUMINATION ABOVE IT

With these modifications, our SL unit can see objects through the surface up to a 1 metre distance, being able to continuously track fingers, hands and markers from the surface to the air. This resulted in a table able to capture different data according to the switchable surface:

- In diffuse states [Figure 38, left], the camera only sees the shapes and markers of the objects directly placed on the table's surface and the projector can only project images on this same surface, thus behaving like a regular tabletop.
- In clear states [Figure 38, right], the surface is transparent, allowing the camera to see what happens above the surface (hands, arms, heads, objects, etc.), and also permitting the projector to project images on any object placed above the surface, even when there is no contact.

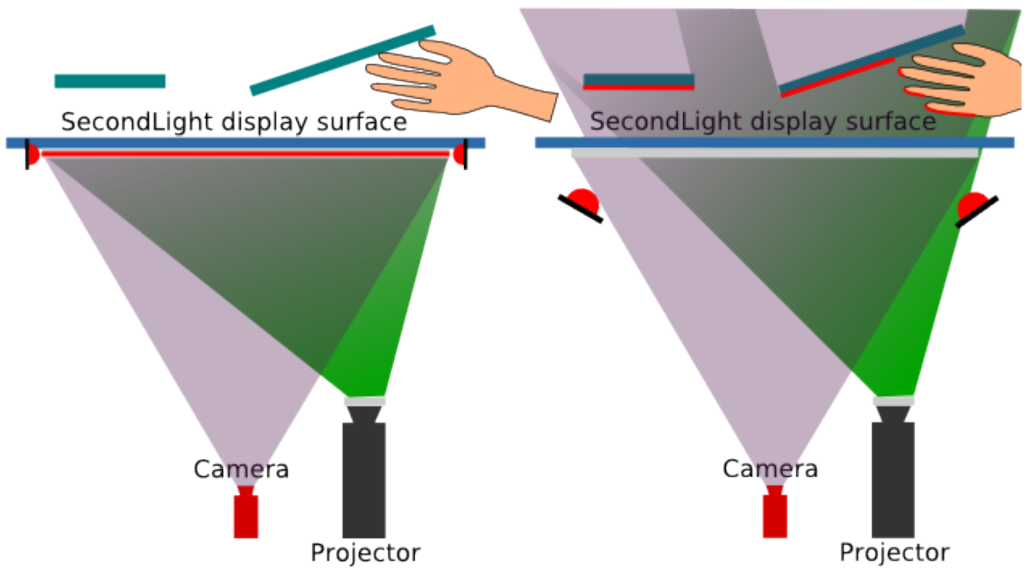


FIGURE 38: THE TWO STATES OF THE SWITCHABLE SURFACE. (DIFFUSE AND CLEAR).

Original SecondLight	**SecondLight build in MSR** (done at MSR Cambridge)	**Modified SecondLight** (done at the UPF)
Two projectors.	One projector.	FTIR with a Plexiglas surface.
Two projector shutters.	One camera.	DI pointing above the surface.
Two cameras.	FTIR with EndLighten.	
FTIR with EndLighten.		

TABLE 6: SL LIST OF MODIFICATIONS

The changes introduced in the SL at the Microsoft research lab and at the UPF lab as shown in [Table 6] produce a new SL tabletop device that is able to fulfil all our tracking needs as well as project visual feedback beyond the display. Classifying the new SL in the Grossman taxonomy will produce the following description of this new SecondLight unit:

- Display properties
    - Perceived display space
        - As SL does not use a stereoscopic vision system nor any volumetric display, this table is a **2D table constrained.** However, as it can project and track special tangibles such as external displays, it can also be classified as a **Surface Constrained** device.

  - Actual display space
    - SL is not a volumetric display nor does it use head mounted displays. It is a tabletop that projects on the surface and above it onto special tangibles. Therefore, it can be classified as both a **2D table constrained** and **surface constrained**.
  - Viewport correlation
    - Given that it is able to project beyond the display, SL is classified with a **semi** viewport correlation.
- Input properties → input space
  - SL has **direct 2D** touching because it can be used as a traditional tabletop. It can also use tangible interaction, where the tangibles can sense touches, thus making it a candidate to be classified as a **Direct surface constrained** as well.
- Physical properties
  - Physical form
    - Undoubtedly, SL is a **table with proxies** because of the use of a tabletop interface and the tracking of in-the-air tangibles and external screens.
  - Physical size
    - The interaction space of SL is reduced to up to a 20-inch screen, making the collaboration difficult between different users. It may thus be classified as a **personal** tabletop.

3.4 Hardware Conclusions

In this chapter we have reviewed the actual state of the art of the hardware used for both tangible surfaces and 3D tangible tabletop surfaces. Some categorizations have been made, dividing these tabletop surfaces according to electronic sensing, IR emitter-receivers or camera-based tabletops.

By the end of this chapter we have seen the Active Cube, a special tangible that augments traditional tabletop interfaces, such as the Reactable, for interacting in the air. We have presented the SL; a special tabletop interface from Microsoft research and the platform for all the following developments conducted in this thesis. This section has also presented the modifications in the unit that we have been working on and its physical and hardware constrains, which resulted on a table that can theoretically:

- Track fingers on the surface.
- Track markers on the surface.
- Track fingers and hands beyond the surface.
- Track 6Dof markers beyond the surface.
- Project graphics on the surface.
- Project graphics beyond the surface.

At this point we can say that these features are theoretically possible because the camera can see all these objects on and beyond the surface, but all of this SL hardware potential is useless if it is not accompanied by a computer vision system able to track all the aforementioned objects. The definition and development of the computer vision system for the SL is addressed in the following chapter.

Chapter 4 COMPUTER VISION SYSTEM FOR THE SECONDLIGHT

Camera-based tabletops are always accompanied by a computer vision system. Apart from leap motion or Kinect that have an internal processor inside the device for pre-processing the captured images, all image processing is typically done at the side of the computer. In this chapter we introduce SecondLight Vision system (SLVision), a computer vision application that we have specially made for SecondLight (SL). SLVision takes advantage of the new SL's capacity for tracking fingers, hands and markers on the surface and above it. In this chapter a new set of tags, SLFiducials is introduced as part of SLVision. SLFiducials are special 6DoF markers designed for fulfilling our need to track objects and portable screens beyond the surface.

4.1 INTERACTION SPACE AND PROBLEM STATEMENT

Having been designed for tracking blobs, touches and fiducials on a plane, most of the camera-based solutions for tabletops are explicitly 2D. Since these fiducials are meant to be used on a flat surface, available tracking systems[46] only report the objects' X and Y positions with the Z angle (yaw). Our fiducial tracking system, on the other hand, would have to be able to determine, for each marked object, its X, Y and Z positions with their respective angles Yaw, Pitch and Roll, as well as finger and hand interaction. Furthermore SL structure and interaction modes impose some additional constraints concerning the shape, size and design of these new fiducials.

[46] ReacTIVision, PixelSense markers, Nui group's Community core vision.

SL is a high table; it is 70 centimetres high with the camera placed at the bottom of it. Due to its switchable display, we need to be able to track objects beyond the table surface, at an interaction distance that will be roughly determined by the user height plus the distance that she could reach with her hands [Section 3.3.2]. This requires tracking markers without losing information at about 170 centimetres (70+100cm) with a 640x480pixels camera. Based on this distance and the camera resolution the minimal size of the new fiducials will be determined. However, it is also important to find the right balance between the new fiducials size and the SecondLight interaction space, especially considering the table's limited surface dimensions (i.e. about 20 inch screen).

According to these physical constraints, the new fiducial markers will have to meet the following requisites:

- The tracking system has to be able to detect markers from a distance of 1.7 meters.
- Markers ought to be as small as possible in order to take advantage of the reduced interaction space.
- Markers ought not to lose performance under severe tilt conditions (roll and pitch).
- The system has to be robust and fast enough for smooth real-time interaction.

There is only one camera in the SL, which captures 2D images. In this baseline structure, it is therefore impossible to access depth data of hands and fingers without introducing a hardware modification. Kinect cameras project an array of IR-dots for calculating the depth data as pointed in [Section 3.2.2]. Placing a Kinect under the SL's surface, the array of IR-dots will collide with the SL's switchable display thus disabling the depth camera contained within this device. Another depth tracking device based on a stereoscopic camera, Leap Motion, was released for pre-order in 22th July 2013. For this reason, it was too late to test it in depth with the SL. However, given that it would be placed under the table's surface, as in the case of the Kinect depth camera, Leap Motion would not work because it cannot be synchronized with the clear states of the SL surface.

As previously commented, it is impossible to add a depth camera inside the SL surface. Therefore, the only method for tracking objects beyond the surface and detect their 3Dposition is by using 6 DoF fiducial markers.

4.2 Related work on fiducial tracking

There is a plethora of Fiducial-based tracking systems for 2D surfaces and for AR installations (i.e. with 6Dof). These can be classified according to their identification and location methods, their shape, colour, range and the size of the markers as it has been pointed by (Owen, Xiao, & Middlin, 2002).

Before introducing SLFiducials, our markers that were especially designed for the SL, an overview of the current markers and algorithms has to be done in order to better understand the design of the new fiducials

4.2.1 6DoF AR markers

The most widespread 6Dof fiducials are the ones used for augmented reality, such as those used by ARToolKit[47], which were developed in 1999 for head mounted displays based on augmented reality conferencing systems (Kato & Billinghurst, 1999) and were lately used on the Magic Book for augmented reality illustrations and collaborative applications (Billinghurst & Kato, 2002), ARToolKit plus (M. Fiala, 2005; Wagner & Schmalstieg, 2007), a modified version of ARToolKit for handhelds and mobile devices and that have lately merged into the Studierstube Tracker (Schmalstieg & Wagner, 2009), and ARTag (Mark Fiala, 2005) markers, which have encoded pattern with built-in robust digital techniques of checksum and forward error correction. All these markers have one feature in common: they are all based on square shaped fiducials with a pattern in the middle [Figure 39]. Their square shape is not an arbitrary decision, given that, to determine the position and orientation of a physical object relative to a camera frame, at least four non-linear points must be matched (Karlsson, Young, & Christensen, 2013; Owen et al., 2002).

[47] http://www.hitl.washington.edu/artoolkit/

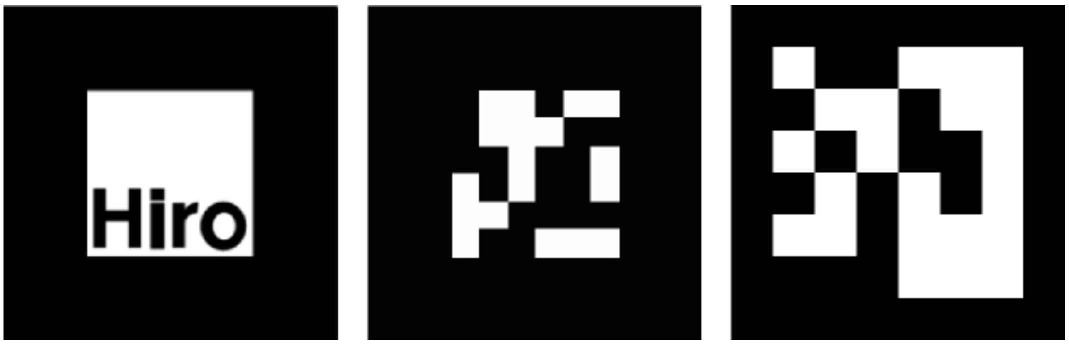


FIGURE 39: FROM LEFT TO RIGHT: ARTOOLKIT TAG; ARTAG AND ARTOLKIT PLUS TAG.

A pose algorithm is responsible for ensuring that the square shape (four points of the square) corresponds with "real" coordinates. Although most pose algorithms such as POSIT coplanar (Oberkampf, DeMenthon, & Davis, 1996) can detect the pose of an image from only three points, the obtained error measure is high, and extra points are needed for disambiguating the right pose from other possible pose candidates. The identification systems for the aforementioned markers are either based on Matrix-Patterns or on Pattern-Matching algorithms, both of which can provide wide subsets of fiducial ids.

Pattern-Matching algorithms (as used, for example, on ARToolKit) are typically designed with offset text or blocks that make the fiducial's orientation unique, which provide markers with user-friendly readable information. Pattern-matching engines have to perform four different controls for each marker (up, down, left, right) in order to find the right marker orientation, whereby the marker's patterns should be asymmetrical. This method is appropriate for books or places where users can understand the meaning of the marker without scanning it with a camera or by using an augmented reality application that does not include a large pattern dictionary. If the pattern dictionary was sufficiently extensive, the identification process could be delayed because the algorithm has to compare each candidate marker four times for each marker stored in the database.

Matrix-Pattern algorithms, such as the ones used on ARToolKit and ARTag, encode a binary digit into each cell. These methods can provide larger ID subsets, but they typically require some additional post-processing algorithms in order to improve robustness and to avoid false positive detections (Mark Fiala, 2005).

SL uses a low-resolution camera that has to detect markers that reach up to 170 centimetres in distance. Therefore, the minimum pixel size required is large. Adapting these AR marker-tracking systems to the SL will produce a set of big makers, which is a drawback given the reduced interaction space of this tabletop. Differently, 2D Tabletop markers are designed to be as compact possible as possible while keeping a reasonable ID-range.

4.2.2 TABLETOP MARKERS

Fiducial or fiduciary markers are based on an encoded pattern that is placed under the tangible object, which will then be placed on the table. These markers need to be as compact as possible and encode an identifier such that they can be distinguished. Tabletop markers, in comparison with the Augmented Reality (AR) markers, do not need any method for 3D pose estimation; they use to provide only the X and Y position plus the Z-axis angle.

A common characteristic of these markers is the use of dot-structures to store information. The dots of the markers are used to group them into planetary constellations such as Surface's tags, or in topological regions such as ReacTIVision's tags, where the dot's size is dictated by the camera's resolution.

Microsoft Surface (PixelSense) Tags are based on a large central point that represents the marker's centre, with a sequence of other smaller dots orbiting around it. By default, it has three satellite dots that determine the tag orientation [Figure 40, left] and other complementary dots encode an 8-bit binary sequence, resulting in a 256 marker variations.

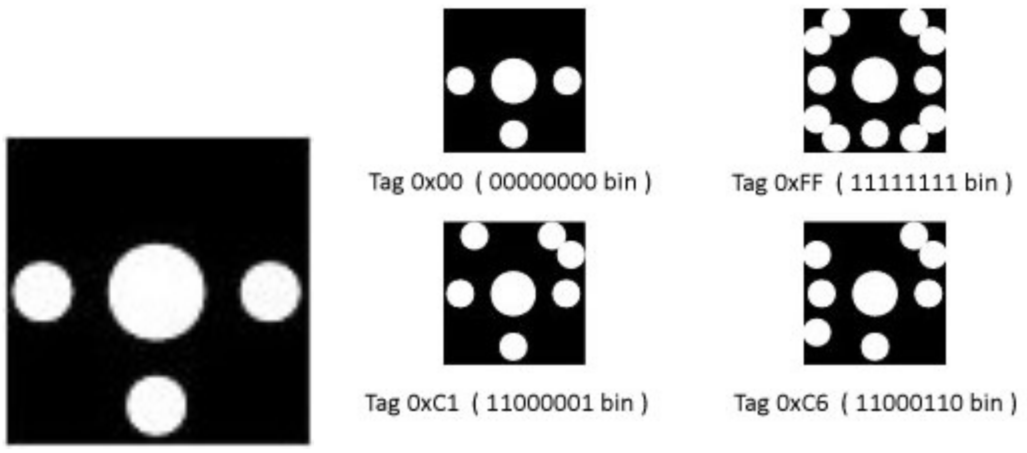

Tag 0x00 (00000000 bin)
Tag 0xFF (11111111 bin)
Tag 0xC1 (11000001 bin)
Tag 0xC6 (11000110 bin)

FIGURE 40 MICROSOFT SURFACE TAGS. BASIC STRUCTURE (LEFT); DIFFERENT COMBINATIONS (RIGHT).

Although Microsoft Surface (PixelSense) tags are very simple, the system does not offer a large ID range, nor a mechanism to reject false positives. These tags are restricted to the Microsoft surface (v1.0) and PixelSense (surface v2.0), thus its use is limited to the tangible tabletop Microsoft platforms.

The most extended and used fiducial tracker for research and homemade tabletop purposes is the **reacTIVision framework** (Kaltenbrunner & Bencina, 2007). Originally developed for the Reactable and released as an application framework under an open source license, this software tracks fingers and markers (fiducials) on a DI tabletop surface and sends the tracked data to the tabletop application through TUIO (Kaltenbrunner, 2009) messages.

Although including other marker detection engines such as D-touch markers (Costanza & Robinson, 2003) and a variation of them called “classic”, the most used marker detection system on reacTIVision is the “amoeba”. This marker engine uses a set of highly compact markers obtained through a genetic algorithm.

The three reacTIVision marker engines are based on the Topological region adjacency tree approach, which does not require large computer consuming disambiguation techniques. In Topological region adjacency, the containership information is expressed as a graph of black and white regions. As shown in [Figure 41, right], each fiducial can be easily translated to an adjacency tree.

Topological region adjacency methods tend to be fast and reliable, and they take a very different strategy for increasing their robustness to that of the Matrix-Pattern and Pattern-Matching methods. While Matrix-Pattern and Pattern-Matching strategies need post-processing techniques such as hamming distance or CRC (Mark Fiala, 2005), topological adjacency relies on the rarity of its structures in order to avoid false positives.

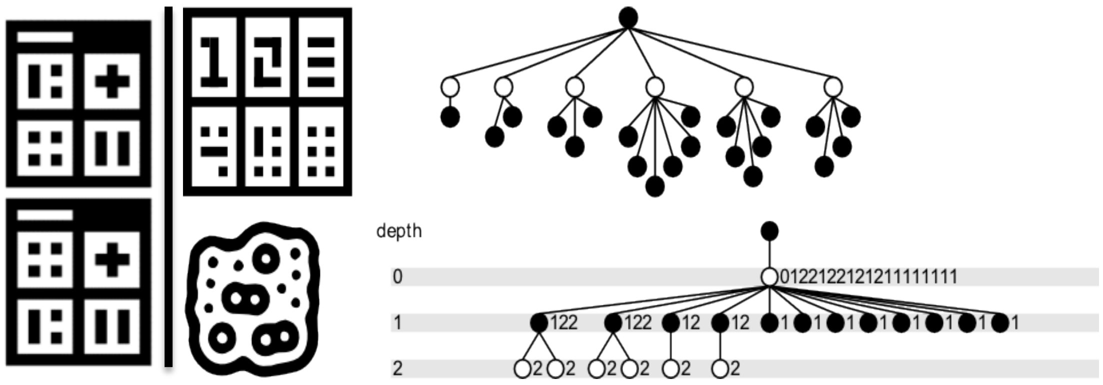


FIGURE 41: D-TOUCH MARKER (LEFT), A CLASSIC MARKER (RIGHT-TOP) AND AN AMOEBA MARKER (RIGHT-DOWN) WITH ITS RESPECTIVE TOPOLOGICAL ADJACENCY TREES.

The topological adjacency trees of the amoeba generate a unique identifier for each different fiducial design, which contains the description of the marker. As seen in [Figure 41, right], the amoeba marker generates a tree that encodes the following sequence: 0122122121211111111. This sequence is a region description of the level of each fiducial: a large white region (0) that contains twelve black regions (1's), of which two contain two white regions each (2, 2), and another two contain one white region (2). Once the fiducial ID is identified and confirmed as a valid fiducial sequence, the position and angle of the marker is calculated. Instead of calculating the midpoint of the marker's outer region, the reacTIVision's amoebas encode its centroid and rotation vector by using the position of the leaf nodes from the generated adjacency trees (black and white dots [Figure 41]):

- The centre of the marker (X, Y) is the midpoint of all black and white dots.
- The rotation vector can be extracted from the marker's centre and the midpoint of the black dots that always points to the upper side of the fiducial.

Although other tabletop computer vision trackers exist, such as those used in MS (Surface 1.0 or PixelSense), ReacTIVision's amoeba allows the production of smaller fiducials by reducing the number of its nodes. In addition, it also reduces the fiducial's ID-range (number of the unique ids). Given the aforementioned space restriction of our system [3.3.2], this size reduction is of great advantage.

However, the tabletop fiducial systems that have been described have been specifically designed for 2D surfaces and they cannot provide all the data that we would need (X, Y, Z plus yaw, pith and roll). Nishino (Nishino, 2010) describes a topological adjacency-based system, designed for 3D interaction, which should be able to provide all the required 3D data. Nishino's tags only encode 17 different topological structures, but, as shown in [Figure 42, centre], they have a sorted dot sequence in which a 16-bit combination is encoded. For this reason, they are able to cover a huge ID range. Therefore, the information needed to determine the tag ID is at the marker's perimeter and a central and bigger black square with a white dot at its centre is used for indicating the marker's orientation. However, it is important to note that this structure results in relatively large markers, and keeping the information on the perimeter makes them more vulnerable to false positives, especially when they are not directly tracked from the top (i.e. when pitch and/or roll do not tend to zero).

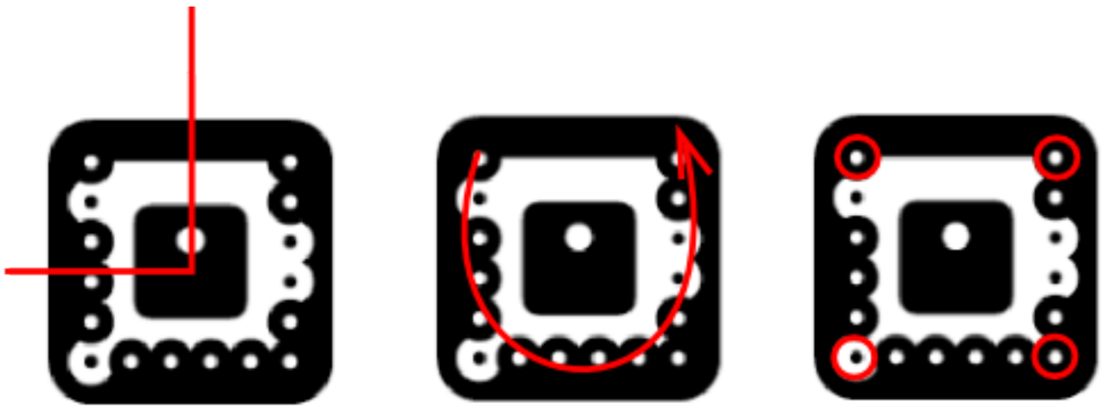


FIGURE 42: NISHINO'S TAG. FROM LEFT TO RIGHT: ORIENTATION, 16 ENCODED SEQUENCE, POSE ETIMATION POTINTS.

While AR tags need complex disambiguation mechanisms, which signify more processing time, Nishino's tags are based on topological adjacency trees. They are therefore easier to verify and, consequently, consume less CPU time. A disadvantage of Nishino's tags is the encoded dot-sequence, which requires a mechanism for reading it. Another weak point is its size because Nishino's markers become too large for the SL.

4.3 Towards the SL Fiducials, markers for the SecondLight

As we have already mentioned, the maximum distance of 170 centimetres of its interaction space z-far is a fixed constraint that will condition the minimal size of the new fiducials. It is also important to find the right balance between size of the fiducials and interaction space, which in the case of SL is limited to a screen of 20 inches. Keeping in mind all these characteristics, we decided to develop our own markers based on those of reacTIVision [Figure 43].

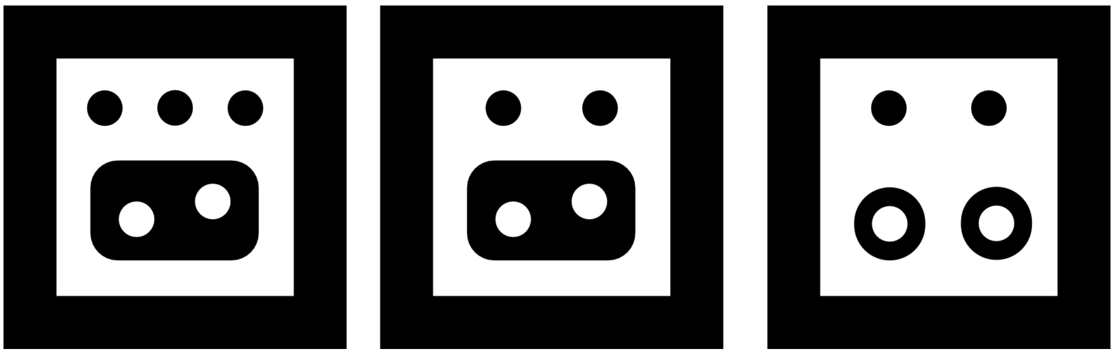


FIGURE 43: DIFFERENT SLFIDUCIALS.

4.3.1 Tracking SLFiducials

The method we propose for 6DoF fiducial tracking combines topological adjacency regions for fiducial identification and a square shape for pose estimation. These markers are closely related to those of reacTIVision, and they use the same mechanism to get the orientation and identification of the fiducial.

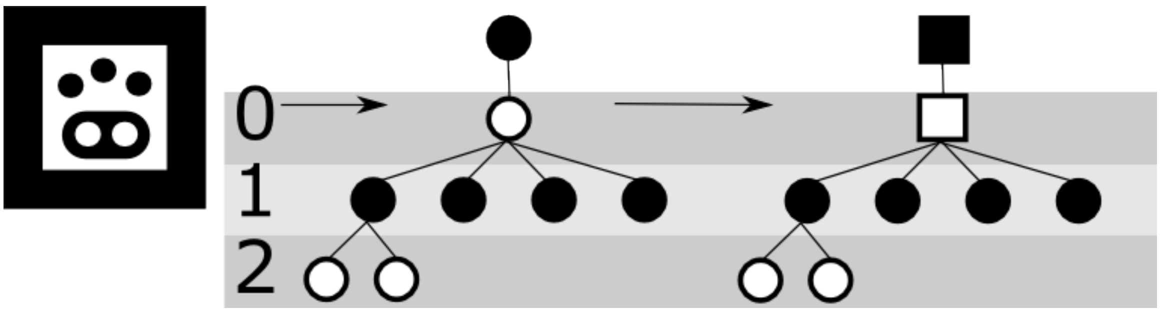


FIGURE 44: SL FIDUCIAL REPRESENTED BY THE ENCODED SEQUENCE: 0122111.

In order to track these fiducials, we use an adaptive threshold to avoid discarding both bright candidates, which are close to the surface, and dark ones, which are located at a particular distance. To avoid false positives and make our system fast and robust, our fiducial finder algorithm is able to detect a valid SLFiducial functions in the following way [Figure 44]:

- For each candidate blob:
    - Detect a topological structure that matches any of the topological codes stored into our fiducial subset database (ID comparison)
    - Find if the root node (Black square) could be approximated to a square shape
    - Find if the immediate node to the root (inner white square) could be approximated to a square shape (level 0 [Figure 44]).

Once our fiducial finder algorithm knows it is a valid candidate, it proceeds to estimate its pose in the real world. However, before applying a pose algorithm, some data is necessary for determine the orientation of the marker.

On the amoeba fiducials of reacTIVision, the centre of the marker is extracted by calculating the midpoint of each final node (white and black dots), while the orientation (angle) is extracted by calculating the vector formed in the centre of the fiducial and the midpoint of all black dots. As shown in [Figure 45, bottom] our SL fiducials use a similar technique, but the resulting vector is only used for detecting the square orientation such that the pose of the marker can be estimated.

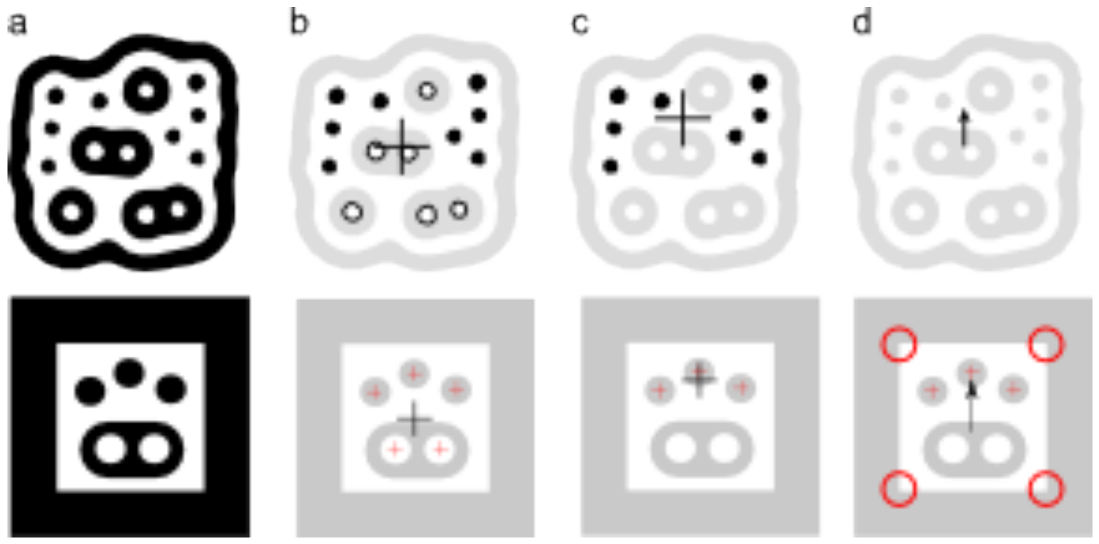


FIGURE 45: REACTIVISION'S AMOEBA CENTER AND ANGLE (TOP); SL FIDUCIAL FOUR POINTS POSE ORIENTATION.

Notice that once we have processed the SL fiducial’s adjacency tree [Figure 44], we have all the necessary data for the pose estimation step:

- Black points correspond to all the last level-one nodes.
- The white ones to the last level-two nodes.
- The contour edges four points have been extracted while applying the first squared shape node check during the fiducial identifying process.

4.3.2 Robustness, ID variation and Size

As in reacTIVision, the robustness of our method relies on the peculiar design of the marker. To improve the marker’s robustness we can perform two different controls:

- Compare the resulting encoded sequence of the adjacency tree with a database, enabling only the codes that will be used (e.g. 0122111 in [Figure 44]).
- Apply a quadrilateral approximation of the shape of the root node, which discards candidates that do not have a square shape allowing us to also use fiducials with a very simple topological tree (e.g. one only node).

The topological structure shown in [Figure 44] is defined by 5 leaf nodes with a 3-level depth topological adjacency tree. It also requires that the first node (the one at level 0) can be approximated to a quadrilateral shape.

Given the size constraints of the markers, a reasonable number of nodes for a SL marker is 5. By using 5 nodes we should be able to obtain 32 different markers (2^5). However, since the order of its nodes is not relevant, and our method requires at least one black dot, we cannot obtain more than 5 variations. Alternatively, we could increase the node number, but this would inevitably produce bigger fiducials. To increase the ID range of our fiducials without making larger markers, we can simultaneously use fiducials with any number of nodes ranging from one to five. This would produce a subset of 15 reduced markers; a number that can be doubled when they are inverted (i.e. including their negative images).

The other factor in our markers that determines their size is the width of the black border. To ensure reliable outline location, the border must be wide enough for quadrilateral simplification (at least for the first white node of the adjacency tree). If the border is too narrow, the noise of the camera could eliminate some peripheral pixels, thus breaking the continuity of the polygon simplification. As a consequence, our algorithm would never find a valid marker. Therefore, our strategy in establishing the size of the border is to make it twice the diameter of the fiducial node. In this manner, the detection of the square regardless on the pose of the marker is ensured.

4.3.3 SL FIDUCIAL COMPARISON AND EVALUATION

In the following subsection, we discuss the results obtained by comparing the performance of our system with other 6Dof fiducial-based tracking systems. These tests have been done with a SL surface functioning as usual; the switchable display flickering at 120Hz (60Hz diffuse states and 60Hz clear states). The computer was a DELL workstation with an Intel Xeon 2.53Mhz processor and 4GB of RAM running Windows7. It is fitted with a 640x480 black and white FireWire camera with an infrared filter pointing towards the surface and beyond.

Performance

In order to test our fiducials, we have developed SLVision (detailed at 4.4). This vision system uses an adaptive threshold to track fiducials at any distance, without being influenced by the variations in incident infrared light. As justified earlier, we ran the SLVision test with a marker dictionary of fifteen items. The Simplelite program we used for evaluating ARToolkit is included in the ARToolkit libraries. It uses a simple threshold and is set up such that it detects only one marker (the marker "Hiro" [Figure 39, left]).

Method	**Time**
SLVision with adaptive threshold	19
SLVision with simple threshold	4.8
ARToolkit (simplelite.exe)	17.54

TABLE 7: **AMOUNT OF TIME IN MS TO PROCESS A FRAME BY USING DIFFERENT APPROACHES AND PROGRAMS.**

From [Table **7**] we can see that ARToolkit is faster than our system when running with the adaptive threshold. For our system, running with SL, we have determined by trial and error that an adaptive threshold is highly necessary such that it is possible to track elements beyond the surface where infrared light is unstable (at more than 20 centimetres distance). However, given that ARToolkit is mainly used under non-infrared systems, it does not include a time-consuming threshold. Therefore, our fiducial tracker system, when running with a simple binary threshold, is 12.4 milliseconds faster compared to ARToolkit.

Size and range

As we have already stated in a previous section [4.1], the size of the markers is an important design issue, which signifies that there is a need to find a balance between size and fiducial effectiveness. In SL, this parameter is determined by identifying the minimum pixel size that can be tracked from the largest possible distance of the camera (170 cm, according to the height reached by a user's arm). As a consequence, we ran some size tests to determine the most appropriate size of our SL fiducials and compare it with different markers sizes. The experiment setup was composed of a tripod with an extensible mechanical arm, which was driven by a servo engine and handled the markers at a 1.5 meter distance from the floor. Once the marker was attached to the mechanical arm, we moved it across the camera field of vision and counted the success rate throughout 1000 frames [Table 8].

	SLFiducial		**ARToolkit**		**Nishino's tag**	
Dot. Size	F. Size (cm)	Results	F. Size (cm)	Results	F. Size (cm)	Results
0.3	2	38%	2	0%	2.5	0%
0.5	3.5	74%	3.6	15%	4.25	0%
0.8	**5**	**98%**	5	78%	6.4	37%
1.5	11	100%	11	100%	13	78%

TABLE 8: FIDUCIAL SIZE AND RESULT COMPARISON WITH DIFFERENT PIXEL SIZES FROM 1.7 METERS DISTANCE.

From our tests we detected that the minimum pixel size that our camera can track from a 170 centimetres distance is a square of about 0.3 cm side in size, therefore we decided to print all markers using nodes with widths of 0.3, 0.5, 0.8 and 1.5 cm. It is important to mention that, since ARToolkit markers are based on image pattern detection, we could not accurately define the size of the dots of these markers. In order to compare these markers with those of SL, we printed both marker systems in the same size. Regarding the tags of Nishino, we have not been able to access any tracker that implements them. However, given that they are partially based on an adjacency topological region akin to that of our system, we added some modifications such that they could be detected (at least in the area of the topological region). [Table 8] shows the optimal balance between the size of the marker and resulting success after conducting our experiments, which is attained when using 0.8 cm nodes. This setup produces markers with a size of 5 cm side, which is an acceptable size for our system.

ARToolkit	**ARTooltik plus**	**SLFiducials**	**Nishino's tags**
1 to ?	4096	30	65536

TABLE 9: FIDUCIAL ID RANGE COMPARISON.

ID range is the maximum number of fiducial ID's that can be reached with a given fiducial design. [Table 9] shows a comparison between ID ranges of different fiducials. The ARToolkit is the only one with an undefined range due to the difficulty in testing it with all possible variations as it is based on undefined patterns. On the contrary, the tags of Nishino are the system with the wider range, as it is able to encode a16 bit sequence.

Our system is the one with the least ID variation, although adding sets of larger markers could increase it. Generating and using larger SL fiducials can be done in order to obtain a larger range variation, but as marker real size is used to adequately estimate the fiducial's 3D pose, the system has to know the various sizes. By instructing the system to link sizes and the IDs of the Fiducials, the real size marker can be accessed such that the different marker sizes can coexist. In [Figure 46] we illustrate different marker sizes (the first one is that used in the evaluation) and the corresponding variations in ID range.

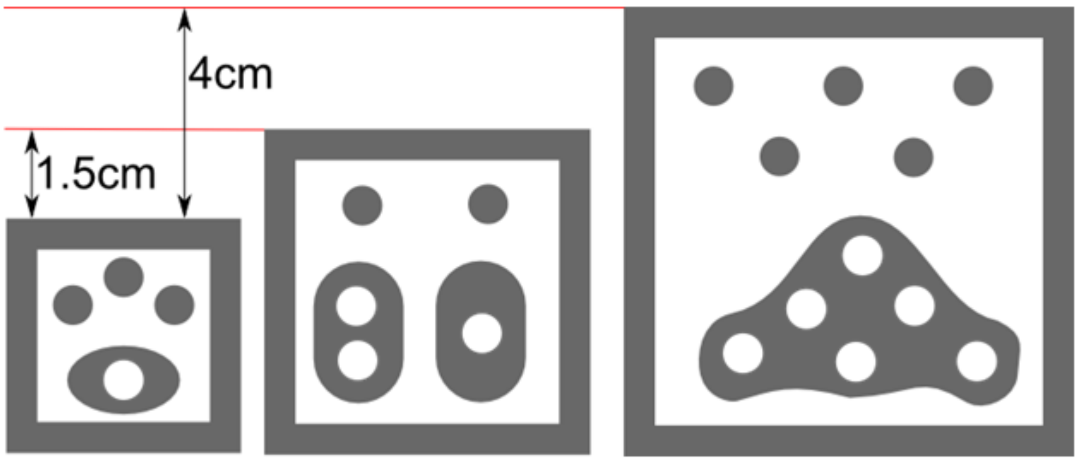


FIGURE 46: FROM LEFT TO RIGHT: 5 CM FIDUCIAL WITH 30 VARIATIONS; 6.5 CM FIDUCIAL WITH 42 VARIATIONS AND 9 CM FIDUCIAL WITH 200 VARIATIONS

The inconvenience of using dots that are smaller than a 0.8 cm diameter is that they could be lost when the marker is not presented horizontally on the surface [Figure 47]. By tilting the 5cm marker at 45degrees [Figure 48], it is easy to observe at the image threshold that the nodes of the Fiducial (black and white dots) are getting closer to the parent region. Similarly, in [Figure 49] the same marker can be seen when tilted 65 degrees, which is the maximum angle at which the marker can be tracked before some of its regions are no longer detected (the bottom dots merge with the background, thus changing the id of the marker).

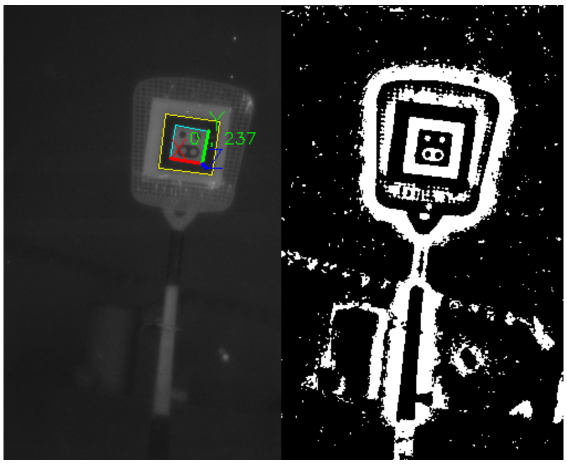


FIGURE 47: SLFIDUCIAL 0 DEGRRES TILT.

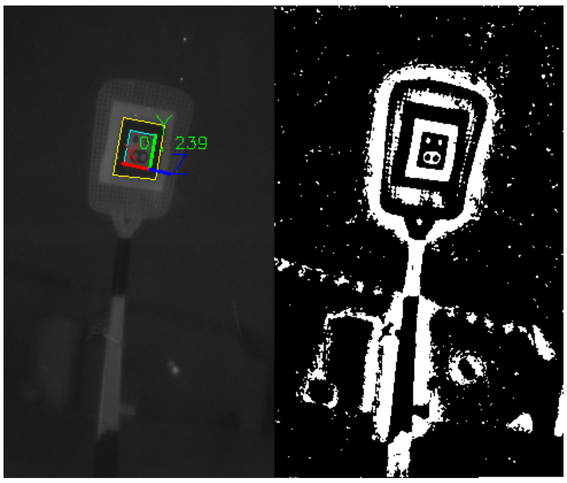


FIGURE 48: SLFIDUCIAL 45 DEGREES TILT.

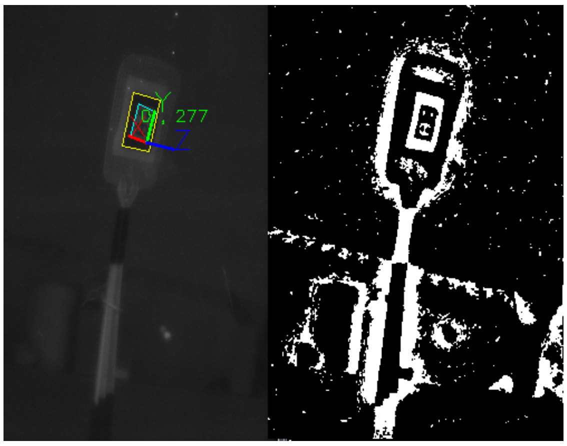


FIGURE 49: SLFIDUCIAL 65 DEGREES TILT.

4.3.4 SLFiducials Conclusion

Although our new markers do not have the largest ID variation range (30 variations for the 5x5 cm markers: 15 and 15 inverted black and white), we can build new and bigger marker sets for complementing to these “reduced size” markers. This results in a 272 fiducial dictionary by combining 5, 6.5 and 9 cm markers, which is more than enough for the surface of a reduced size tabletop such as SL.

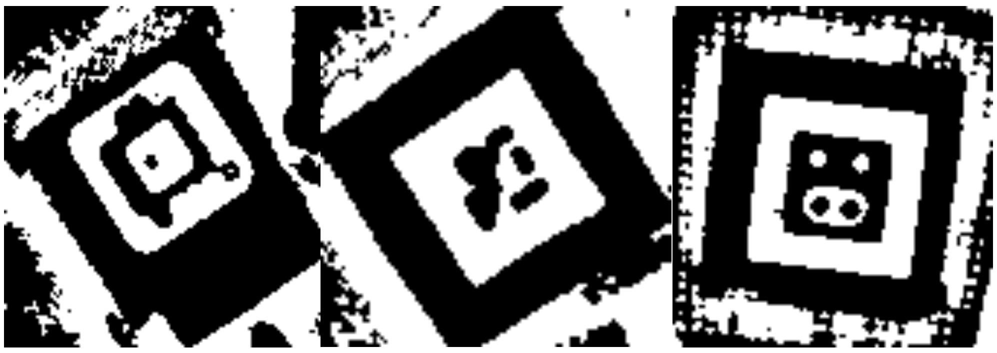


FIGURE 50: FIVE CENTIMETRES MARKERS TRACKED BY OUR SYSTEM: NISHINO'S TAGS (LEFT); ARTAG (CENTRE); SLFIDUCIAL (RIGHT).

By using markers whereby the dots have a size of 0.8 centimetres in diameter, we ensure that they will be tracked even when at a distance of up to 170 cm from the camera. [Figure 50] shows a screen capture of 5 cm printed tags placed at a 1.7 metre distance: On the left is a Nishino’s tag, whereby the entire encoded sequence has disappeared thus making it impossible to recognize the fiducial ID; in the centre is an ARTag with the same problem as that of Nishino’s tag; on the right is a SLVision fiducial that can be recognized by our fiducial engine because it can be selected by our fiducial finder algorithm.

From now on in this thesis the markers used will be the SLFiducials ones as they have been specially made for fulfilling our interaction requirements on the SL and from our tests they have been demonstrated that are the ones with better results in terms of size and tracking speed.

In summary, in this section we have defined SLFiducials, 6DoF markers for 3D tabletop interaction that are specifically designed for SL. We have proven that these markers are faster and recognized more easily in comparison to the various AR fiducials (in terms of size and processing time in the SL platform). However, we still need to be able to track surface touches and hand gestures.

4.4 SecondLight Vision System, SLVision

The aforementioned SLFiducial recognition engine has been included inside SLVision. In this section we focus on the complementary development that we have done for the overall computer vision system for the SecondLight. This section covers the entire process, from the raw camera frame to the “surface and in-the-air” frame disambiguation, hand and finger tracking and the defining of the messages sent by the SLVision to the SL application.

SLVision is a computer-vision software that we have specially created for SL. It combines fiducial tracking, finger detection and hand gestures detection. The tracked data is packed and sent to the graphics application (SL application) by using TUIO2-messages 3D specification[48] and some modified or added messages for hand information and fiducial pose matrices (translation and rotation). This software is licenced under a GNU Affero General Public Licence and is publicly available at the MTG’s github[49]. Although it has been designed for SL, this 6DoF fiducial tracker can be used in other platforms such as AR applications or other tabletop surfaces that can “see” through its surface.

Below is detailed description of the process through which hands and fingers are detected in-the-air, as well as that for disambiguating the gestures performed on the tabletop surface or above it, keeping in mind that SL has a single camera that, at each frame, sees what happens on and above the surface.

4.4.1 Frame processing

In contrast to reacTIVision, which uses a single adaptive threshold for detecting markers and fingers on the surface, we cannot apply this method in order to distinguish between fingers that are on the surface and those above it. As previously commented, SL is fitted with a double illumination system: FTIR for surface tracking and DI for lighting beyond the surface elements. Taking advantage of this double illumination system, it is possible to distinguish very

[48] http://www.tuio.org/?tuio20
[49] https://github.com/MTG/SLVision

lighted elements seen by the camera as bright blobs [Figure 51,centre] from other elements that receive less light [Figure 51, left].

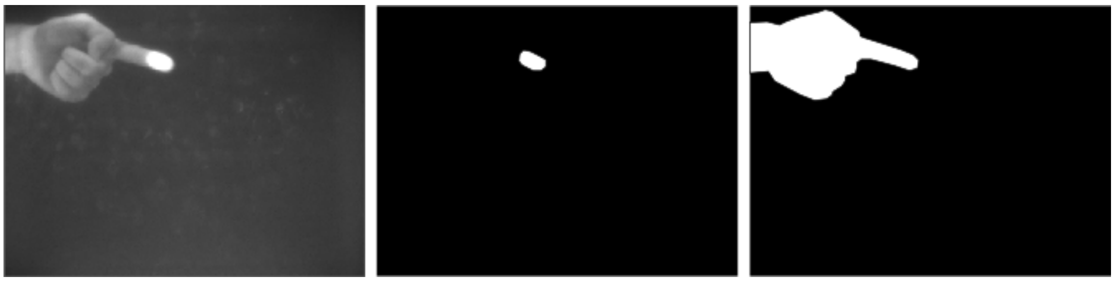


FIGURE 51: SLVISION CAPTURED FRAME (LEFT); THE SAME FRAME WITH THE ON-SURFACE THRESHOLD APPLIED (CENTRE); THE FRAME WITH THE ABOVE-THE-SURFACE THRESHOLD APPLIED (RIGHT).

By applying two thresholds on the captured frames, our system is able to distinguish fingers on the surface (the bright blobs) from the elements (hands, fingers, objects, portable screens) that are above the surface [Figure 51].

Due to this double threshold, one specifically for low light levels (DI at the clear states) and the other for higher light levels (FTIR at the diffuse states), it is possible to process these two frames separately to detect touches on the surface and hand gestures above it. We call these frames the "contact frame" and the "beyond the table frame" or "air frame".

4.4.2 CONTACT FRAME

By analysing the contact frames, we can easily detect blobs on the surface. For now, and given that other solid objects are not lighted by FTIR illumination [as explained in page 49] we only consider all the blobs on this frame as touches, therefore discarding other information. In order to track a "touch", SLVision checks the compactness of the blob and its size and determines if the blob could correspond to a touch. When the system recognizes all touch candidates, these blobs are marked as possible touches to be reported to the SL application but not yet validated.

4.4.3 AIR FRAME

The threshold employed for beyond-the-surface detection is an adaptive threshold, which avoids the possibility of losing the information due to the tracking distance (objects will receive less infrared light when they are further away from the surface). From the "air frames", the current SL tracks two types of objects: SL fiducials and hands. The method for finding SL fiducials has been explained in the previous section, and all objects that are not considered as fiducials by the system are considered as hand candidates.

Tracking a hand beyond the display implies that very complex algorithms might be needed to extract the contour and skeleton of the hands. Therefore, instead of going deep into a very accurate skeleton detection, we decided to focus our algorithm on a very basic hand characteristics recognizer:

- Contour for projecting shadows on the surface and on the hand.
- Fingers with pointing data.
- Wrist and arm for hand orientation.
- Pinch gestures.

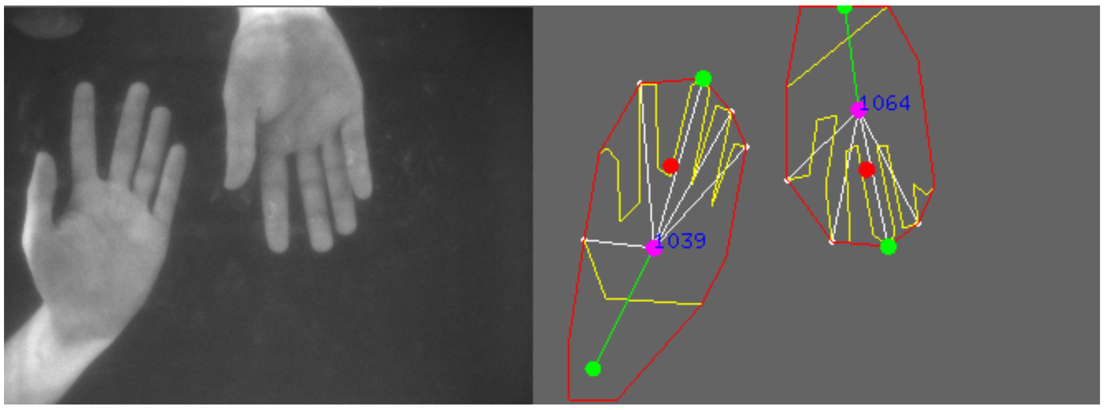


FIGURE 52: CAPTURED HAND FRAME AND ITS PROCESSING DATA.

As we can see in [Figure 52, left], a hand is always linked to an arm. Consequently, we need to separate them such that only the hand data is processed. We started working with the hand skeleton by using the Image Foresting Transform (IFT) (Falcão, 2004), but we realized that the implied calculations highly increased the computational cost, obtaining less than 15 frames per second. We therefore decided to design a simpler algorithm based on the convex hull from OpenCV[50].

SL provides only black and white frames because it uses infrared light, which signifies that we cannot apply any colour filter technique to distinguish the

[50] http://opencv.org/

hands from the arms or other elements. Instead of filtering by colour, we applied some basic constrains for selecting valid candidates: first, a hand is attached to an arm, which can emerge from any side of the table, and second, this hand-arm set cannot appear suddenly (it needs to come from one side of the SL surface area). By applying this constrain, we can thus select blob candidates for hand detection. These blob candidates must remain in permanent contact with all the sides of the table, as seen in [Figure 52]. There is a green point in the table's perimeter for each detected hand pointing where the arm starts. Furthermore, the blob candidates should cover an area that is larger than a given threshold such that false positives can be rejected.

Once the hand candidate blobs have been selected, our hand tracker calculates the convex-hull for each candidate (red path in [Figure 52, right]), and the convexity defects (valleys) between each convex-hull segment (blue circles at [Figure 53]) in order to detect the fingers and wrist. Once this data has been processed, a hand can be identified by finding its wrist and fingers:

- Wrist: defined by the first two valley points from the side of the table where the blob starts (yellow hand shape starting from wrist in [Figure 52]).
- Fingers: defined by the larger distances from each convex-hull points to their valleys (White lines in [Figure 53]).

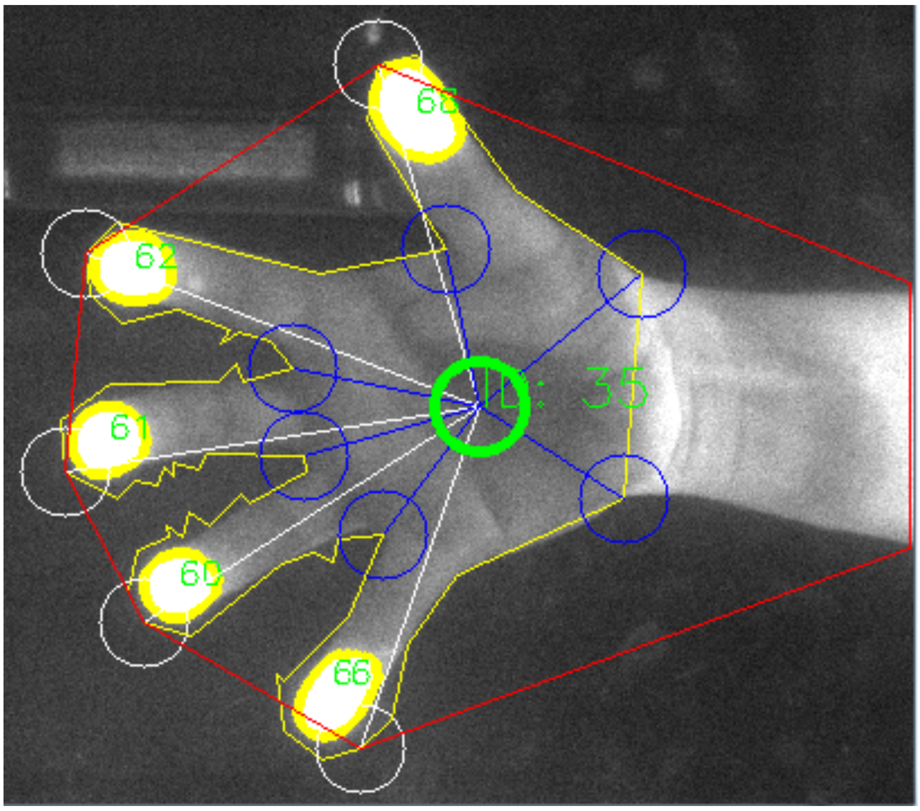


FIGURE 53 DETAIL OF HAND DETECTION.

The data extracted from each tracked hand does not contain any 3D information, but can be expanded by merging the data extracted from the airframes with the data of the contact frames.

4.4.4 Finger disambiguation and hand interaction

One of the problems of using a binary threshold for the detection of fingers on the surface is that any bright point above the surface will be detected as a touch (i.e. Metal reflexions of the IR light or sweaty hands) [Figure 54]. By cross-referencing data from the contact frames and the airframes, we can easily disambiguate the false positives on the contact frame. This is achieved by matching touch candidates with in-the-air hand fingers. Therefore, if a detected finger is not under an in-the-air hand, it is not considered as a finger. By applying this disambiguation technique, we can considerably reduce the false positives and obtain more reliable data.

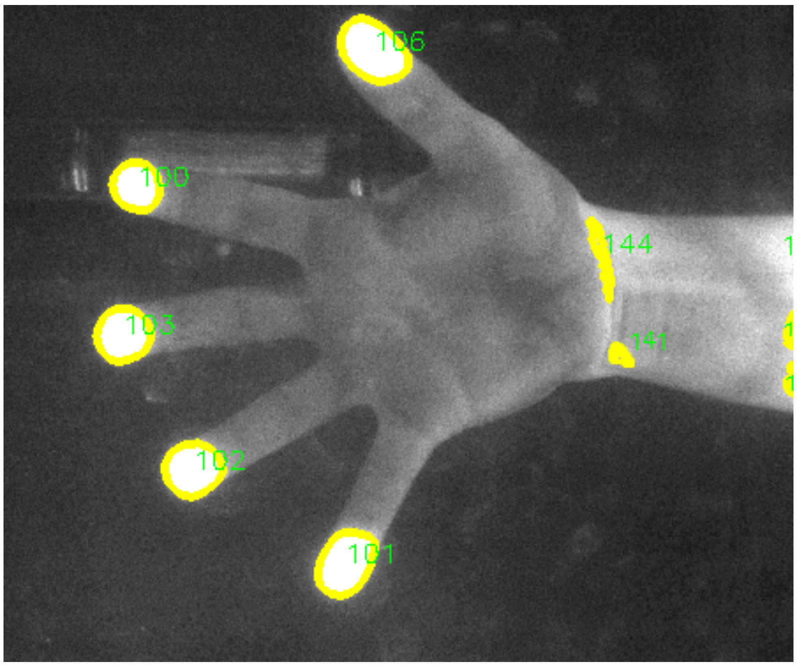
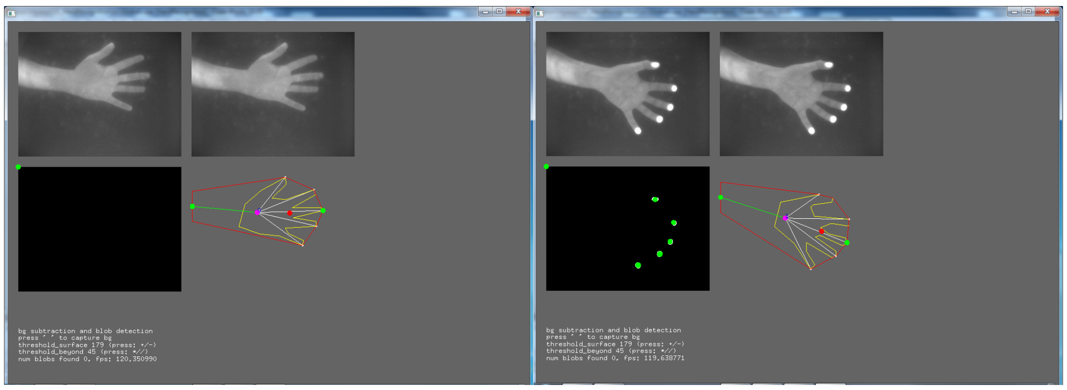


FIGURE 54: EXAMPLE OF FALSE POSITIVE FINGERS BECAUSE BRIGHT AREAS ABOVE THE SURFACE (WIRST AND ARM).

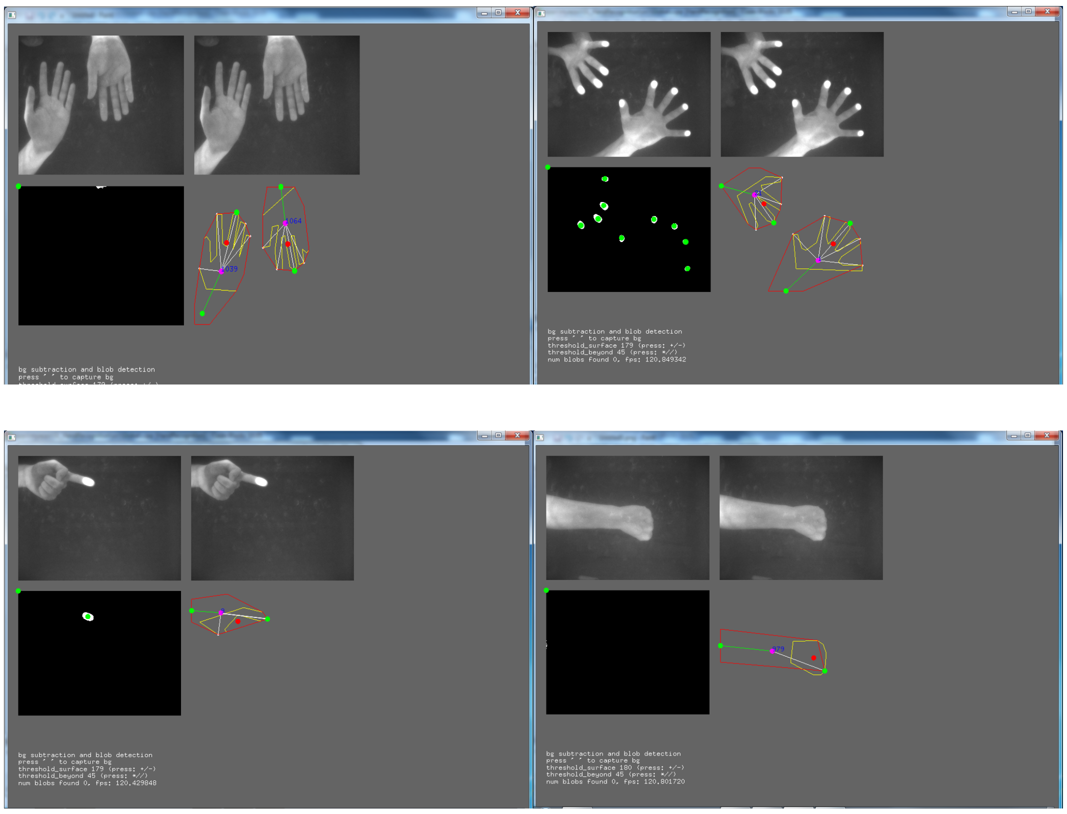
In-the-air hand tracking without cross-referencing tracked data, it is reduced to a planar in-the-air interaction because no depth data is available; the camera of SL does not provide depth data. Enriching this interaction only requires to track when the finger touches the surface. By using the tracked data from the contact frame, we can guess that the hand (or a part of it) is touching the surface, thus tracking the hand not in 2D but in 2.5D:

- One finger touching the surface$\rightarrow$ hand Z = 0
- Any finger touching the surface $\rightarrow$ hand is in the air.
    - At this point we can estimate if a hand is getting closer or further away based on the area it covers.



```
bg subtraction and blob detection
press ' ' to capture bg
threshold_surface 179 (press: +/-)
threshold_beyond 45 (press: */)
num blobs found 0, fps: 120.350990
```

```
bg subtraction and blob detection
press ' ' to capture bg
threshold_surface 179 (press: +/-)
threshold_beyond 45 (press: */)
num blobs found 0, fps: 119.638771
```



```
bg subtraction and blob detection
press ' ' to capture bg
```

```
bg subtraction and blob detection
press ' ' to capture bg
threshold_surface 179 (press: +/-)
threshold_beyond 45 (press: */)
num blobs found 0, fps: 120.849342
```

```
bg subtraction and blob detection
press ' ' to capture bg
threshold_surface 179 (press: +/-)
threshold_beyond 45 (press: */)
num blobs found 0, fps: 120.429848
```

```
bg subtraction and blob detection
press ' ' to capture bg
threshold_surface 180 (press: +/-)
threshold_beyond 45 (press: */)
num blobs found 0, fps: 120.801720
```

FIGURE 55: DETAIL OF FINGER DIAMBIGUATION WITH DIFFERENT HAND-SHAPES.

Time tracking consistency

SLVision has been designed to track fingers, hands, and markers defined by constraining lists that determine whether a particular blob is a hand, finger or marker. Once a blob has been identified via a hand ID or marker ID, SLVision keeps a time-persistency of the data even if it is not possible to disambiguate the blob on certain frames.

In relation to the markers, this mechanism is useful when the marker has an assigned ID, but the marker cannot be recognized in certain frames because the adjacency regions do not match. In this case, SLVision (when possible) only tracks the marker's square perimeter and uses the data of the marker (ID and orientation) from the previous frame for pose estimation.

The same mechanism does not apply for hand persistency in time, being based on a changeable shape and size blob. The data from the previous hand frame is used for disambiguating hand occlusion and crossing hands. By keeping the information of the arm direction (start and end), a more persistent and reliable hand tracker is achieved.

4.4.5 Hand and Finger Results and evaluation

Finger detection, as mentioned before, is based on tracking blobs on the "contact frames" by checking their size and compactness. The problem of this method is the tracking of false positives introduced by interferences on the frame to be processed (a shinny object beyond the screen, light reflections, etc.). However, combining touch information with the hand data can reduce this problem. [Table 10] shows a comparison of processing contact frames with reacTIVision, SLVision without checking hand data and SLVision while checking hand data during 7200 frames (two minutes video) that contain 100 real touches and 52 false touches (blobs) caused by light interferences.

	ReacTIVision	SLVision without disambiguation	SLVision with finger disambiguation
Tracked fingers	82	97	100
False positives	30	48	4

TABLE 10: TRACKING FINGER RESULTS USING REACTIVISION AND SLVISION.

ReacTIVision follows a similar finger detection algorithm to SLVision, but it also checks the shape of the touch, thus reducing the detection of false positives. In comparison to SLVision without finger disambiguation, this method also discards some real touches. SLVision tracks every blob (of a certain size) on the surface and considers them as touches. This means that it will validate almost all false positives. However, by applying finger disambiguation, SLVision reduces almost all false positives and provides a robust touch tracking system.

Hand tracking is often affected by noise interferences as seen in [Figure 56, right] whereby the blob (white shape) becomes unstable at the edges or some other areas because the IR-light does not arrive uniformly.

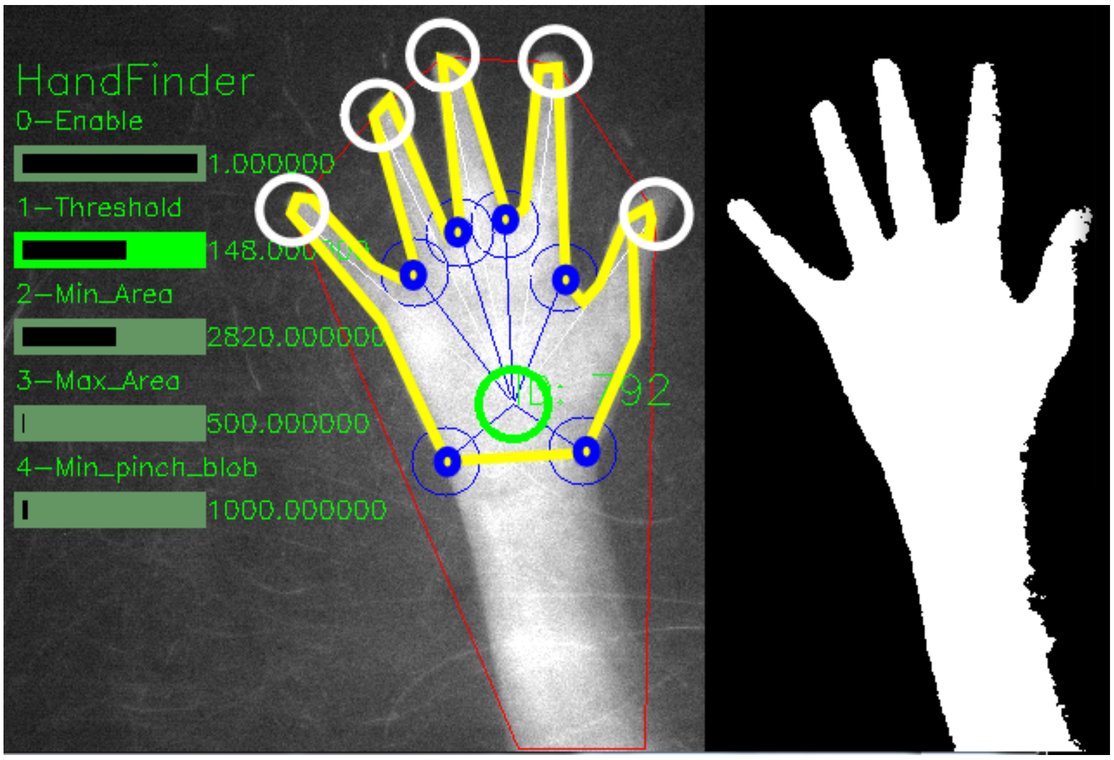


FIGURE 56: TRACKED HAND (LEFT); AIR-FRAME (RIGHT).

We do not apply any algorithm for avoiding these noise interferences SLVision just simplifies the blob contour to reduce the number of points from the hand path, which also decreases the sharp and irregular edges caused by the noise. Hand finder system runs at 60 FPS, which is the maximum frame rate attained by the camera taking between 1 and 5 milliseconds to process a frame depending on the number of hands and objects above the surface.

4.4.6 Reported TUIO Data

Once all these data (markers, hands and fingers) is tracked, we need a system for sending these data to the application that will use it for the gesture analysers and the application's logic. SLVision sends the tracked data to the graphics application via TUIO Messages. SLVision (TUIOServer) is connected to the camera and it is in charge of tracking anything that cameras can see on or above the surface. The Graphics application (TUIOClient) is connected to the projector, receiving the data from the TUIOServer and generating graphics for the surface and above the surface projections [Figure 57].

The data that SLVision sends to the TUIOClient application is based on the TUIO2 3D specification, but differs in some areas because it sends more data than the expected:

- Fingers:
    - Session ID
    - X position
    - Y position
    - Area → mainly used as a finger pressure indicator.
    - Is_in_the_air → Boolean flag that says that the finger is not touching the surface.
    - Hand_ID → the hand identifier where the finger belongs.
- Hands:
    - Session ID
    - Centroid X → all blob centre (including arm)
    - Centroid Y → all blob centre (including arm)
    - Hand-area → blob area
    - Start-arm X → x point at the perimeter of the surface where the hand starts.
    - Start-arm Y → y point at the perimeter of the surface where the hand starts.
    - End-arm X → x point at the perimeter of the surface where the hand ends.
    - End-arm Y → y point at the perimeter of the surface where the hand ends.
    - Hand X → hand centre x.
    - Hand Y → hand centre y.
    - Hand-influence → hand diameter influence.
    - Pinch X → pinch gesture x position.
    - Pinch Y → pinch gesture y position.
    - Pinch-influence → pinch diameter influence.
    - Num-fingers → number of visible fingers.
- SLFiducials:

- Session ID
- Fiducial ID
- X position
- Y position
- Z position
- Yaw
- Pitch
- Roll
- Rotation matrix (9 numbers).

For a better gesture detection system, SLVision sends as much data as it can to the TUIOClient application (SL application). In comparison to the “finger” section of the normal TUIO data, we have added three new fields: the area, which is useful for pressure detection, the in-the-air flag, which indicates whether the finger is in the air or not, and the hand identification system, which associates the finger with the hand. In the “fiducials” section, SLVision follows the TUIO 3D specification, but adding at the end the rotation matrix for CPU saving time when matrix reconstruction on the graphics part.

Hand TUIO message, which is not included in the TUIO specification, can be divided into three parts. The first includes general blob data such as the blob’s centroid, the start and end points of the arm that is useful for tracking the users’ position, and the blob area. The second refers to the hand-only gesture-related data, which include the hand position with its influence area on the surface and the visible fingers to provide data for grasping and finger pointing gestures. The last part is the pinch gesture, which corresponds to any closed area inside the hand reporting the pinch-position and its influence area.

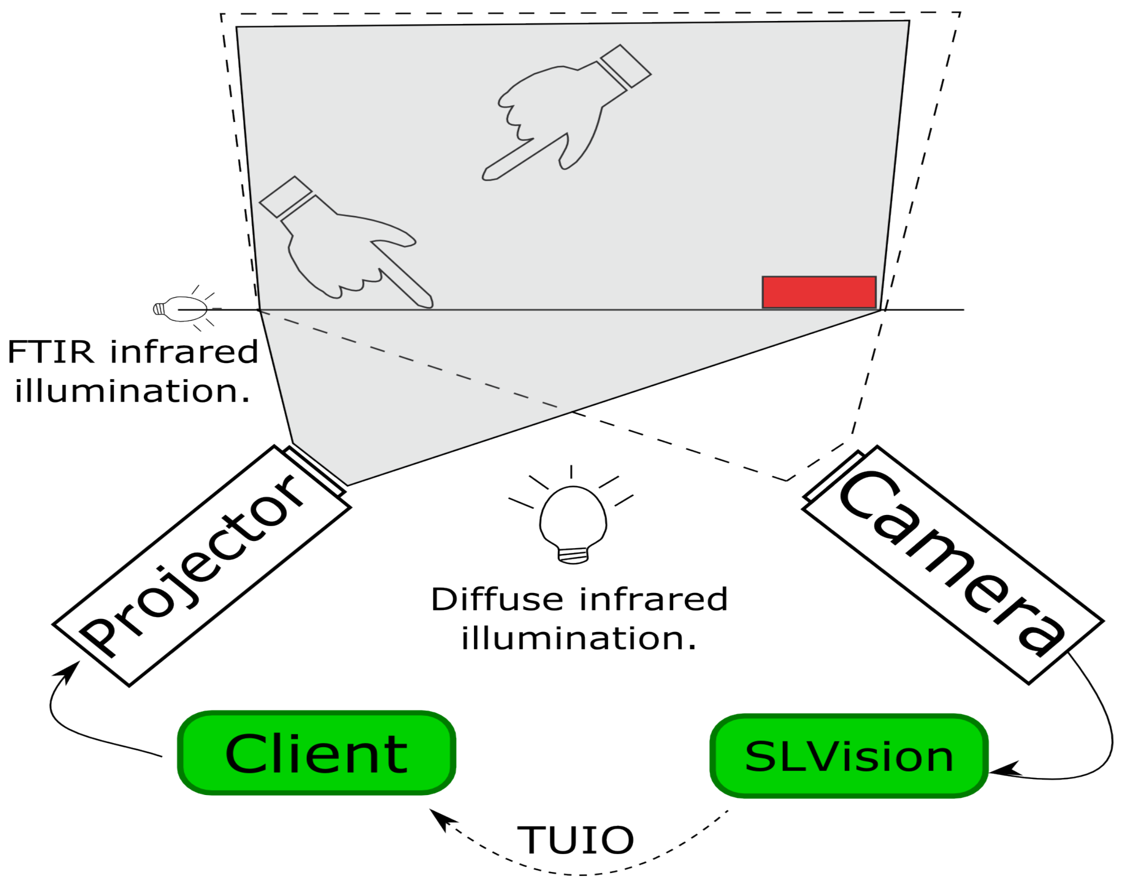


FIGURE 57: STRUCTURE OF A SECONDLIGHT APPLICATION.

4.5 Results and conclusions

In this Chapter we have introduced SLVision, a computer vision system specifically built for the 3D tabletop device SecondLight. This vision software is able to track with only one camera and two light sources three kinds of objects on and above the surface: markers, by indicating their position on the 3D space, fingers, by differentiating when they are on the surface or above it, and hand data. We have also defined new markers for tabletop interaction designed to be as small as possible but keeping a considerably high ID range. Finally, we have presented a comparison and evaluation of these fiducial markers.

All the captured data are sent to the graphics application, which will define the various gestures and interactions on the SecondLight. This graphics application can be built with a special framework that we have named SLFramework, which will be addressed in the next chapter.

Chapter 5 SECONDLIGHT TABLETOP INTERACTION

In this chapter we present the work we have conducted on the definition and development of a framework for the SecondLight based on the TableGestures framework and that interfaces with the SLVision mentioned in previous chapters. This framework is ready for in-the-air tabletop interaction as well as managing the continuous interaction space. In order to test that this framework is able to extend tabletop interaction, we have also developed some demo applications to explore different interaction metaphors that will generate new gestures. These will be evaluated later in this chapter and sorted in a new 3D Tabletop Gesture classification system.

5.1 SECONDLIGHT FRAMEWORK

The SecondLight (SL) unit we received did not contain any kind of graphic framework or library for the development of applications. Consequently, we need to explore and develop a new framework that allows us to develop SL applications, deal with the double projection (surface and beyond) and receive the messages generated by the SLVision system.

Following the same lines of TableGestures, and encouraged by the several tabletop applications developed for the Reactable platform with the same coding framework, we decided to create a SecondLight framework (SL Framework) to manage the new SL features. Before building this codding framework, we started by adding some modifications to the surface 1.0 SDK and XNA3 for the 3D objects representation and, later, to the surface 2.0 SDK and XNA4, which were resulting to be a very limited tool.

Surface SDK is a content oriented framework where controls and chrome are secondary (MS.Corporation, 2011). The Microsoft surface guidelines maintain that this surface is a 360^0 oriented tabletop despite being a rectangular table.

They define a subset of widgets and rules whereby all control elements, such as menus and input areas (e.g. virtual keyboards) can be oriented towards any side of the table. In contrast to circular tables, rectangular tables always have a dominant side where the interaction is more fluid and all elements are within an easy reach.

SecondLight is a relatively small rectangular table, where the surface interaction space is much reduced even when using the above-the-surface interaction space. Due to the physical constraints and the tracking method used by SL (a camera below the surface), we do not consider multiuser interaction with this device: if a user were to be occupying the surface using tangibles and touch, the camera would not be able to see elements beyond the surface controlled by a second user because the objects on the surface would occlude them.

The new SL framework is oriented towards a single user interaction because of the size constrains. It allows real-time interaction, which signifies that the system does not have to wait for the user input. Furthermore, it is gesture-oriented instead of content-oriented and benefits from the second projection in order to visually augment the tangibles, the external displays and the graphic feedback beyond the screen. This framework is coded in C# using the XNA4 libraries for the 2D and 3D graphic output and allows the use of graphic components from the very basic polygon shapes by using triangles to more complex graphics calculations programmed with shaders.

In the following subsections, the parts that were specially built for the SLFramework are explained:

- Graphics
- Gesture Manager
- Simulator
- External Display
- Graphic feedback beyond the screen.

5.1.1 GRAPHICS

As commented on the (3.2.3) describing the SL hardware, SecondLight uses a single 3D projector for displaying images on and above the surface. In order to manage the two projections, SL has a built-in proxy graphics card that has two VGA video inputs and one VGA output for the 3D projector. Therefore, the graphic engine from the framework has to produce an output for two different graphic contexts: one for the surface and the other for the above-the-surface projection. These graphic contexts are possible due to the extended desktop mode of Windows7 and the creation of two different windows for displaying graphics: one on the main desktop (surface) and the other on the extended one (beyond the surface).

Regarding the coding, these two contexts are transparent to the programmer, achieved by enabling two methods at the SL application base class: one for drawing on the surface (DrawSurface) and another for the above-the-surface drawings (DrawBeyond) without having to change the screen output context each time [Figure 58]. These two methods are initially prepared for an output of 2D graphics, but they can be easily set up for 3D graphics by modifying the XNA's graphic context camera.

```
namespace TestGraphics
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class SLApp : SecondLightApp
    {
        public SLApp()
        {
        }

        protected override void LoadContent()
        {
            //initialize your objects here
        }

        protected override void DrawSurface(GameTime gameTime)
        {
            //put here daws for the surface
        }

        protected override void DrawBeyond(GameTime gameTime)
        {
            //put here draws for the second projection
        }
    }
}
```

FIGURE 58: EXAMPLE OF A BLANK SL APPLICATION

As in the case of TableGestures, SLFramework also implements tangible areas, which are gesture sensing surface regions. Tangible areas are user-defined areas that detect specific input events that take place on them. Because tangible areas are two dimensional, they have been designed to be used (or at least declared) on the surface's methods such that they can properly receive any kind of gesture event generated on or above the surface. These areas can be polygons without any visual feedback or inherit any of the SLFramework shape subset:

- Circle
- Square
- Rectangle
- Regular polygon
- Polygon

These shapes are implemented under the Figures library that provide methods for detecting gesture collisions and rendering very basic graphic options such as texture mapping, stroke painting, rounded edges, circle resolution, etc. In this library, the base class of all figures is the Polygon shape that represents any convex and non-convex polygon thus allowing the addition and subtraction of other Polygon instances. Polygon implements a triangle tessellation based on “the triangulation by ear clipping algorithm” (Eberly, 1998) that splits any shape into a set of triangles to calculate and draw collisions [Figure 59 generated by the code at the Annex 3].

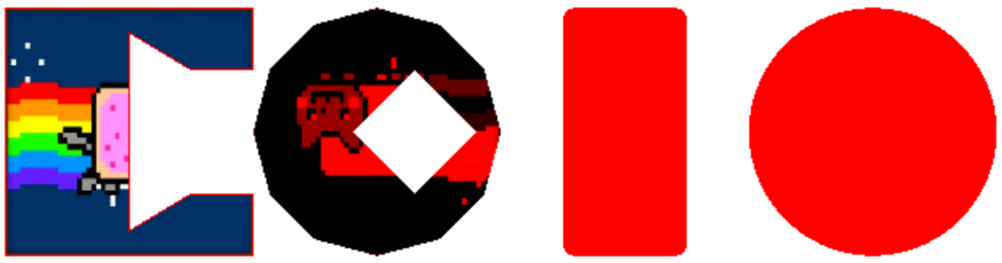


FIGURE 59: DIFFERENT SHAPES RENDERED BY USING SLFRAMEWORK FIGURES' LIBRARY.

All graphics under this coding framework are driven by the XNA4, a library for game drawings and the generation of real-time graphics. XNA needs a virtual camera for rendering the drawings within the graphics context. By default, the surface and the beyond-the-surface camera are centred and point to the surface from above (Z is at 1.2 pointing to 0 in the following lines) in order to display a 2D view of the graphics:

```
slg.camera = new Camera(
            this,
            (float)(BackBufferWidth / 2f) /
            (float)BackBufferHeight,
            new Vector3(ww, hh, 1.2f),
            new Vector3(ww, hh, 0f),
            Vector3.Up);
```

This camera can be modified at any time by accessing the SL graphics helper (slg) camera and modifying it such that it allows dynamic camera parameters to change with the movement of a fiducial marker as shown in one of the SL Application demos (below).

5.1.2 Gesture manager

The manager for receiving gestures on this framework follows the same strategy to that in TableGestures: all the gesture reports are incremental, starting from the “gesture” TUIO-message, which is validated by the gesture TUIO2dot0_validator and, at the same time, is used by:

- **Tuio2_Input_Figures6DoF** generates events reporting the position and orientation of the different tangibles on and above the surface.
- **Tuio2_Input_Hand** generates events with the received hand data.
- **Tuio2_Input_touch** generates input touch events.

By using composite gestures to define other gestures, some complex gestures can be implemented that would otherwise be very difficult to generate from scratch (Hoste & Signer, 2014). As an example of these composite gestures we define the gestures **pick, move and drop** as a composite of other basic gestures:

- Pick an object from the table (made of two basic gestures):
  - Gesture Pick
    - Tuio2_Input_touch tracking fingers of the same hand that converges to a centre.
  - Gesture Hand-up
    - Tuio2_Input_hand reporting when the fingers’ hands are not touching the table.
- Hand-drag an object above the table (made of one basic gesture):
  - Gesture Move-hand:
    - Tuio2_input_hand reporting the hand’s position when it is closed (detection of any finger being tracked).
- Hand-drop (made of one basic gesture):
  - Gesture Open-hand:
    - Tuio2_input_hand reporting when a hand changes it state; from closed to open (all fingers visible).

This gesture example is implemented by using 3 different complex gestures (table-pick, hand-drag and hand-drop) that use one or more basic gestures (pick, hand-up, move-hand, open-hand) composed of the very basic input gestures from the SLVision.

To define these composite gestures, is needed a SL surface unit furbished with all of the mentioned SL modifications (Chapter 3), and the SLVision (Chapter 4) must be running. When these elements are not available, an input simulator has been added for testing and developing applications without having to need a SecondLight interface.

5.1.3 Simulator

Many tangible tabletop simulators exist. PixelSense has the "Microsoft surface input simulator" that simulates finger touches with orientation and fiducial objects through the process of dragging and dropping virtual elements on the windows desktop. Another, ReacTIVision, has the TUIO-simulator, a java-based application that simulates a tabletop surface where users can drag virtual fiducial objects inside a virtual table area and simulate finger touches "clicking" with the mouse.

Differently, SLFramework has a built-in simulator [Figure 60], but SL has implicit complications that make the simulation experience a difficult task. Regarding the input and output, the simulator would have to be able to generate 6DoF object events, touch input and hand interaction, while it should also allow to simultaneously view the data on the surface and beyond the surface in the same output window.

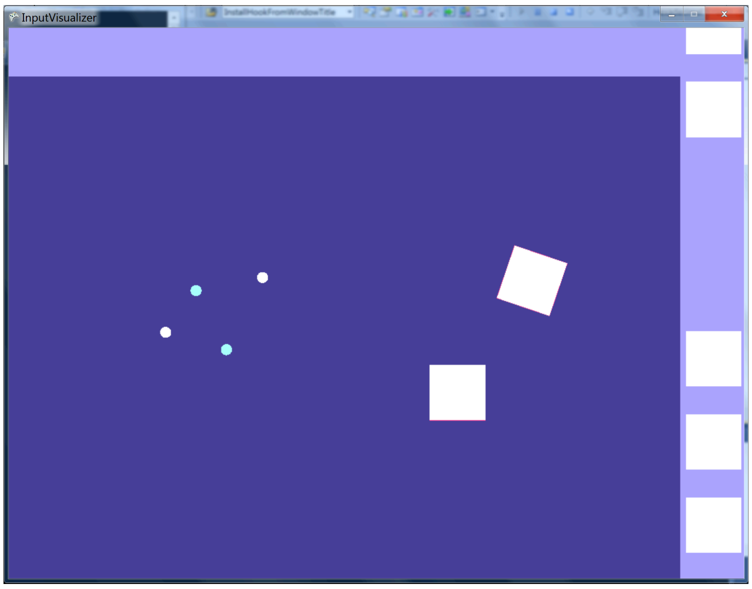


FIGURE 60: SECONDLIGHT SIMULATOR.

Simulating touch interaction (without hand grouping) is not a difficult task. Fingers are added and dragged with the mouse and, by pressing a key combination; the finger changes its state from being on the surface to being in the air. Furthermore, by taking advantage of the mouse's scroll wheel, we can simulate the pressure made by the finger when spinning the wheel.

Each tangible (object) can be dragged to the tabletop from a tray menu placed on the right side of the simulator. They can be moved along the X and Y axes by moving the mouse, and the Z coordinate is modified by spinning the mouse's wheel. By pressing the right mouse button and moving it along the X and Y axis, it is possible to modify the pitch and roll angles of the tangible. Furthermore, spinning the mouse's wheel while pressing the right button, the yaw angle is modified [Figure 61].

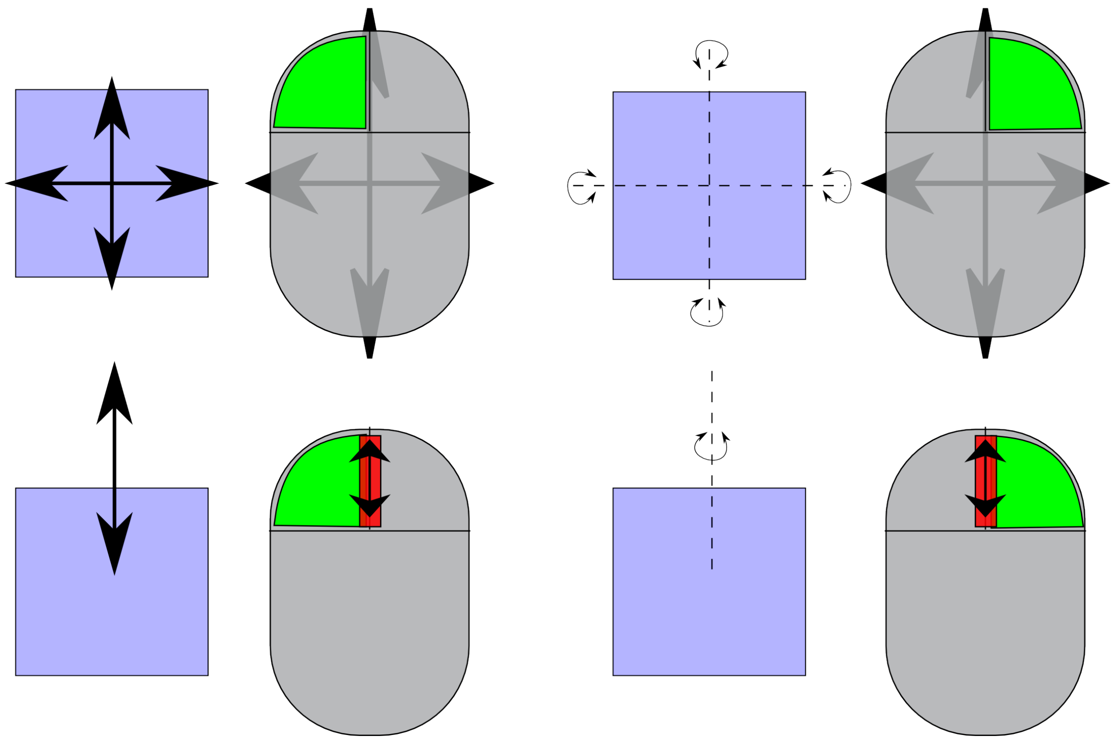


FIGURE 61: SL SIMULATOR: MOUSE MOVEMENTS MAPPED INTO A 6DOF VIRTUAL TANGIBLE. (TOP, LEFT) X AND Y MOVEMENT; (BOTTOM, LEFT) Z MOVEMENT; (TOP, RIGHT) PITCH AND ROLL MODIFICATION; (TOP, DOWN) YAW MODIFICATION.

Hand gestures are very difficult to simulate and SLVision does not include a simulator for hand interaction because different parameters must be moved at the same time in order to simulate a single hand, which would be very hard to conduct using only a keyboard and a mouse.

Regarding the graphics, we tried to visualize all the graphics in the simulator screen simultaneously, but the results were not those expected; the graphics "beyond the screen" occluded the graphics "on the screen", and finally the simulator shows the visual feedback in two separated windows instead of one to avoid the object occlusion. This took place particularly when using external displays showing large figures.

5.1.4 External display

SL's external displays are made of a Plexiglas surface with built-in FTIR illumination [Figure 62, left]. The original SL tracks these screens with the help of two reflective IR stripes that delimitate it. The problem of this method is that the system is unable to distinguish between different screens. Taking advantage of the new 6Dof fiducial development, and that the side of the screen where the battery is located is not transparent for the projection we have marked these screens with an SLFiducial under the battery place for knowing The screen's pose in the air as well as identifying and distinguish them.

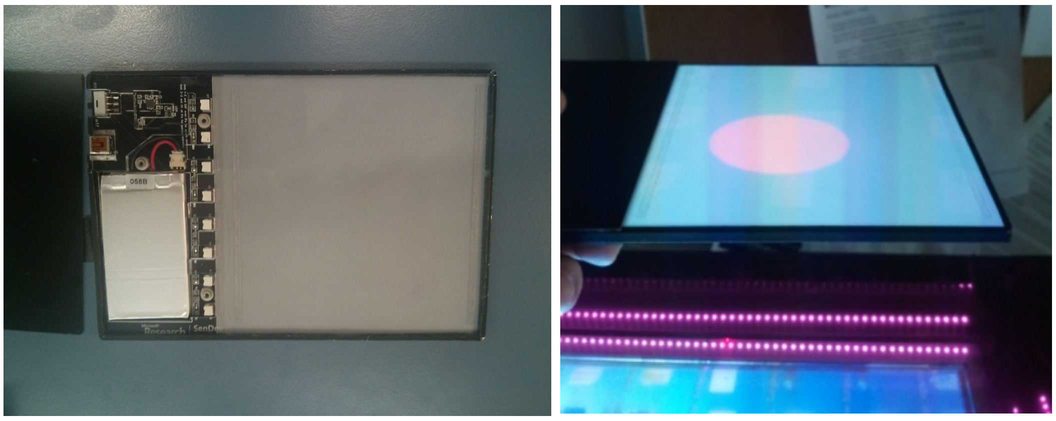


FIGURE 62: EXTERNAL DISPLAY FTIR LIGHTING (LEFT); PROJECTING A RED BALL ON AN EXTERNAL DISPLAY (RIGHT).

In order to project images on the external screens, we simply have to know the SLFiducial displacement in relation to the projectable area. Afterwards it is easy to calculate the 3D pose from the positions of the markers and the rotation matrix [Figure 62, right].

5.1.5 Graphic feedback beyond the screen

Interacting beyond the screen implicates the loss of the physical component of touching and grasping virtual elements that is possible with surface computing.

Furthermore, by manipulating virtual objects beyond the screen, the direct data manipulation when any visual feedback is displayed beyond the screen is no longer feasible. However, external displays and tangible interaction beyond the screen have the implicit feedback provided by the physical component of the object, which does not occur with hand and finger interaction. To obtain visual feedback, the SLFramework can use two methods:

- Rendering shadows on the screen.
- Projecting images on hands and fingers above the screen.

By mapping shadows on the screen, the user achieves more precision than by performing "blind" gestures in the air, since the hand projection on the table facilitates gesture positioning in relation to the virtual surface objects.

Some gestures for holding virtual elements in the air, assigning a special function to a hand or simply identifying each finger with a colour for a finger-painting application, need to be treated differently compared to when a virtual shadow is projected on the surface. This must take place in order to emphasize the particular finger/hand ability or to keep the surface as clean as possible without an overload of information. This effect can easily be achieved by directly projecting images on the hands or fingers above the surface, for example, by projecting virtual objects on the hands when an element is taken from the table or by projecting a colour to each finger for the aforementioned finger-painting application.

5.2 SL Applications

To test the viability of this framework we have developed some demo applications. In this section some of the developed demo applications for SL are presented:

- Photo viewer.
- Colour torch.
- SL Theremin.
- Map depth navigation.
- Volume slicing display.

These demos use different interaction metaphors and gestures for resolving a concrete task. They are presented in an ascendant "in-the-air" order starting from the simplest tabletop interaction to the "in-the-air" exploration of 3D volumes.

5.2.1 PHOTO VIEWER

Photo viewer is based on the same concept as all multi-touch photo browsers: a set of pictures that can be translated, scaled and rotated by performing multi-touch gestures on them. In fact, Photo viewer is based on the same demo presented for the original SL where, by applying the concept of magic lenses, a hidden image appears when a piece of paper is placed on the photos shown on the surface. This application adds to the traditional photo browser applications by using some extra multi-touch and in–the-air gestures:

- Multi-touch surface gestures used:
  - Surround-and-expose: this gesture consists on grouping a set of images on the table by surrounding them with a finger trace. Once the photos are surrounded, they are stacked until the gesture finalizes and can be exposed by dragging a finger over the photo stack [Figure 63].
  - Expose: the gesture consists in double-tapping an image to orientate it automatically to the user's position by tracking the arm direction.

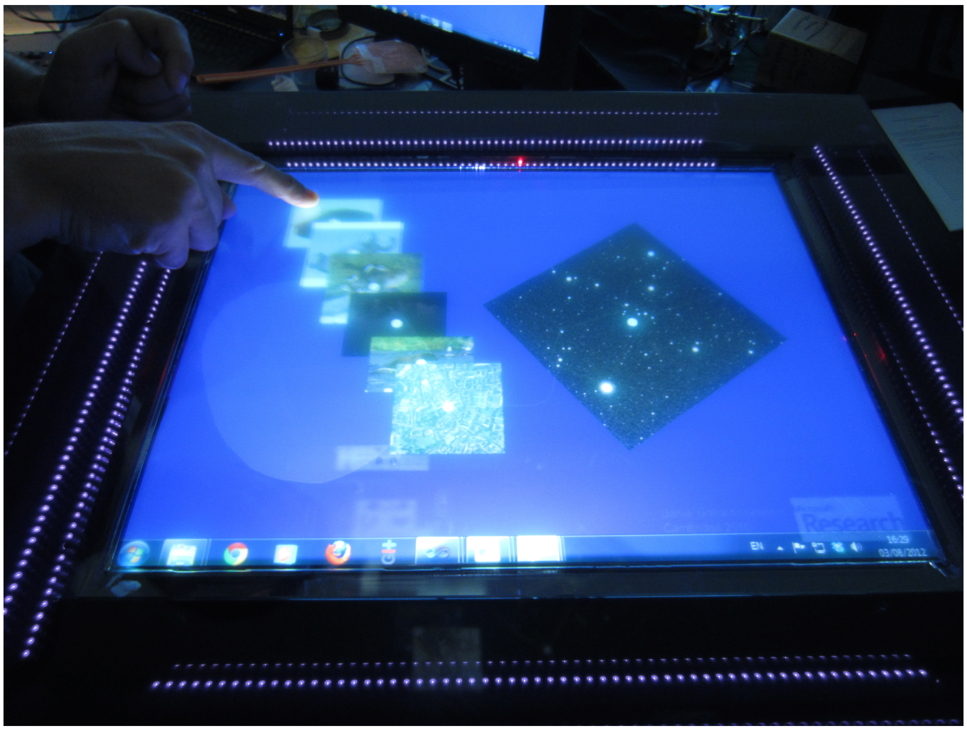


FIGURE 63: SURROUND AND EXPOSE GESTURE ON THE SL.

- In the air gestures used:
  - Air-pinching: this is the default gesture included in the SLVision tracker (4.4), It consists in making a pinch gesture in the air to select an image, dragging it to any part of the screen and releasing it. This gesture must always be accompanied by a visual representation of the hand's shadow on the surface for position hinting.
  - Grasping: Consists in opening the hand over a group of pictures and closing it in the air to select, move, and later stack them on the desired place by opening the hand again [Figure 64]. This gesture needs the same shadow feedback as the air-pinching gesture to visualize the candidate virtual elements to be grasped.
  - Picking: This gesture mixes surface and air interaction. It starts by touching a picture with more than two fingers (two fingers are used for the multi-touch pinch gesture), putting them together on the surface virtual element. By closing the hand and raising it, the element is displaced and later released at the end of the gesture when the hand is opened again it. Unlike the two previous

gestures, this one does not need shadow feedback on the screen because it starts by directly touching the element to be manipulated. Yet, a projection on the hand is necessary in order to indicate that the hand has picked something.

- Expose: Achievable by doing any of the previously described "in-the-air" gestures, this gesture consists in selecting a picture and releasing it in the same place in order to guide the picture to the user's position.

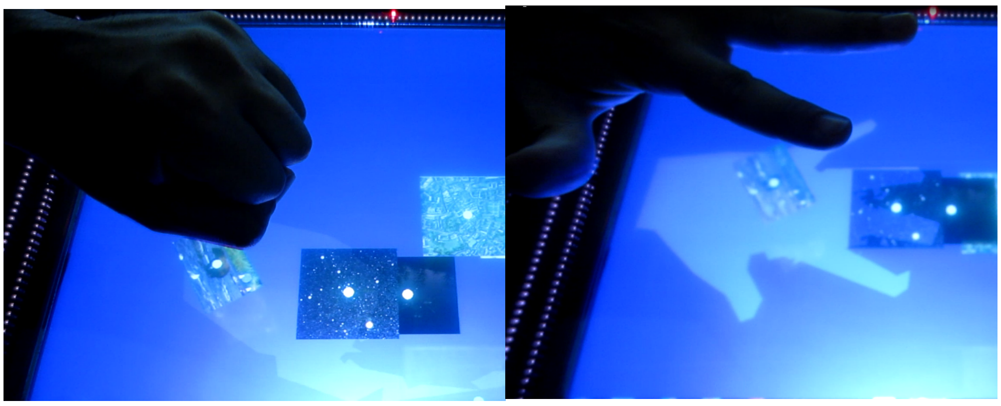


FIGURE 64: IMAGE GRASPING (LEFT); RELEASE GESTURE (RIGHT).

In this application, some gestures and concepts have also been used to grab pictures and drop them somewhere on the surface. The gestures used for this purpose have been named: "air-pinching", "grasping" and "picking". These gestures are also accompanied by different feedback from on or beyond the display: projected hand-shadow, projected item on the hand above the screen and a combination of both.

Projecting hand shadows on the surface is not a novelty given that it has also been used in multiple 3D tabletop applications (De Araújo et al., 2013; Grossman & Wigdor, 2007; Hilliges et al., 2012; Huppmann et al., 2012; Sutcliffe et al., 2013), but projecting the grasped item directly on the hand is a not as common and few examples can be found (Wilson & Benko, 2010).

Photo Viewer User tests

Two user tests were done in the context of this application: one to determine the best way to project the hand shadow and another to establish the precision of the aforementioned gestures and to assess the extent to which users feel comfortable with the in-the-air gesture feedback. During the tests, users were asked to carry out some tasks and answer a questionnaire at the end. These tests were conducted on a balanced male-female sample of 30 users aged between 25 and 35 years.

In the first test, which explored hand shadow feedback, we proposed different shadow alternatives for the task of grasping a picture and releasing it on a stack of pictures [Figure 64]. The different feedback options were:

- None: no feedback was provided.
- Hand contour: only a polyline that followed the contour of the hand was drawn.
- Opaque shadow: the hand blob was painted in black.
- Transparent shadow: the hand blob had been drawn but with an applied alpha at 50%.

By using a 0 to 5 scale questionnaire, the following categories were evaluated for each gesture:

- Easy to use: the user had to determine if the feedback was suitable for performing the grasp gesture in terms of pointing and locating the virtual data on the surface. In this point, 0 meant completely unusable, the virtual object on the surface is impossible to be located, and 5 meant that even the feedback as the object location are easy to see / reach.
- Efficiency: The user had to indicate how much they felt that the gesture during the task was precise.
- Comfortable: the user had to establish if the shown feedback was graphically pleasant or confortable. In this point, 0 meant completely unpleasant feedback and 5 pleasant and comfortable to use.

- Efficiency: In this evaluation point we did not wanted to analyse quantitative speed results and we asked how much they felt that the gesture during the task was precise.

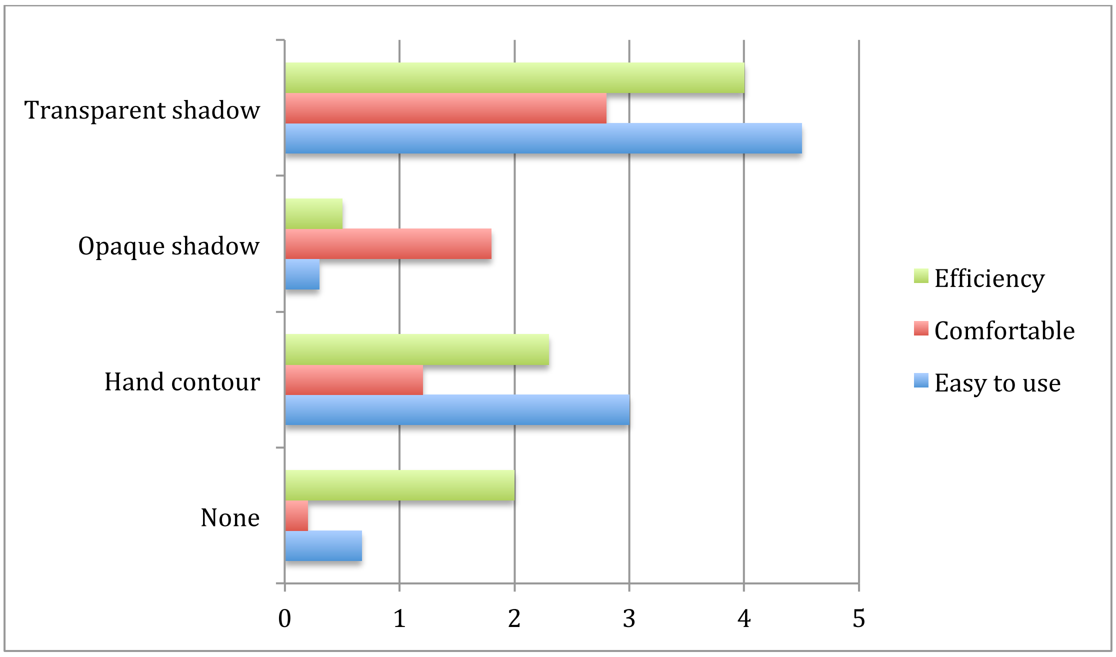


FIGURE 65: RESULTS FOR DIFFERENT SHADOW REPRESENTATIONS

While showing any kind of feedback or drawing the hand contour was confusing (not comfortable) to the users because it was very difficult to determine on which element the gesture was performed, showing an opaque shadow made it harder to determine the target element for performing the requested action (low level at the “easy to use”). This was because the same hand feedback occluded the virtual data behind the hand, thus the final candidate shadow feedback was the transparent shadow. The transparent shadow is sufficiently large such that it is not confused with the different virtual data, but, at the same time, it lets the user see through it and view the virtual data behind the hand and thus carry out the gesture.

Regarding the evaluation of the various “air gestures”, we asked for the following perceptive data by evaluating different “air gestures” with in-the-air graphic feedback and without it:

- Precision on single Figures: A subjective evaluation of the gesture applied on a single picture, where 0 meant very impossible to reach the objective of the task and 5 meant very easy to perform the task.
- Precision on multiple Figures: A subjective evaluation of the gesture applied on multiple pictures, where 0 meant very impossible to reach the objective of the task and 5 meant very easy to perform the task.
- Easy to use: An evaluation of how easy to use was the gesture (not related with the task). In this evaluation point, 0 meant impossible to learn and perform and 5 meant completely intuitive and easy to reproduce.

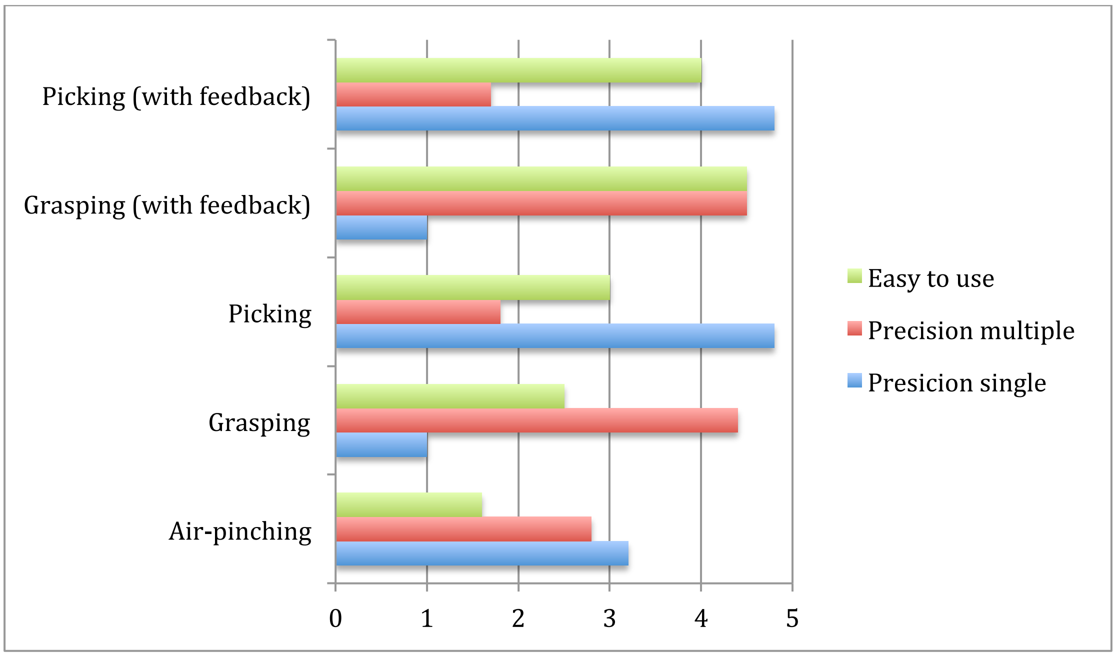


FIGURE 66: RESULTS ON DIFFERENT AIR-GESTURE COMPARISON.

From the data in [Figure 66], it is possible to see that some gestures are useful for single-object manipulation, while others are suitable for multiple-object manipulation. We can therefore extract the following conclusions:

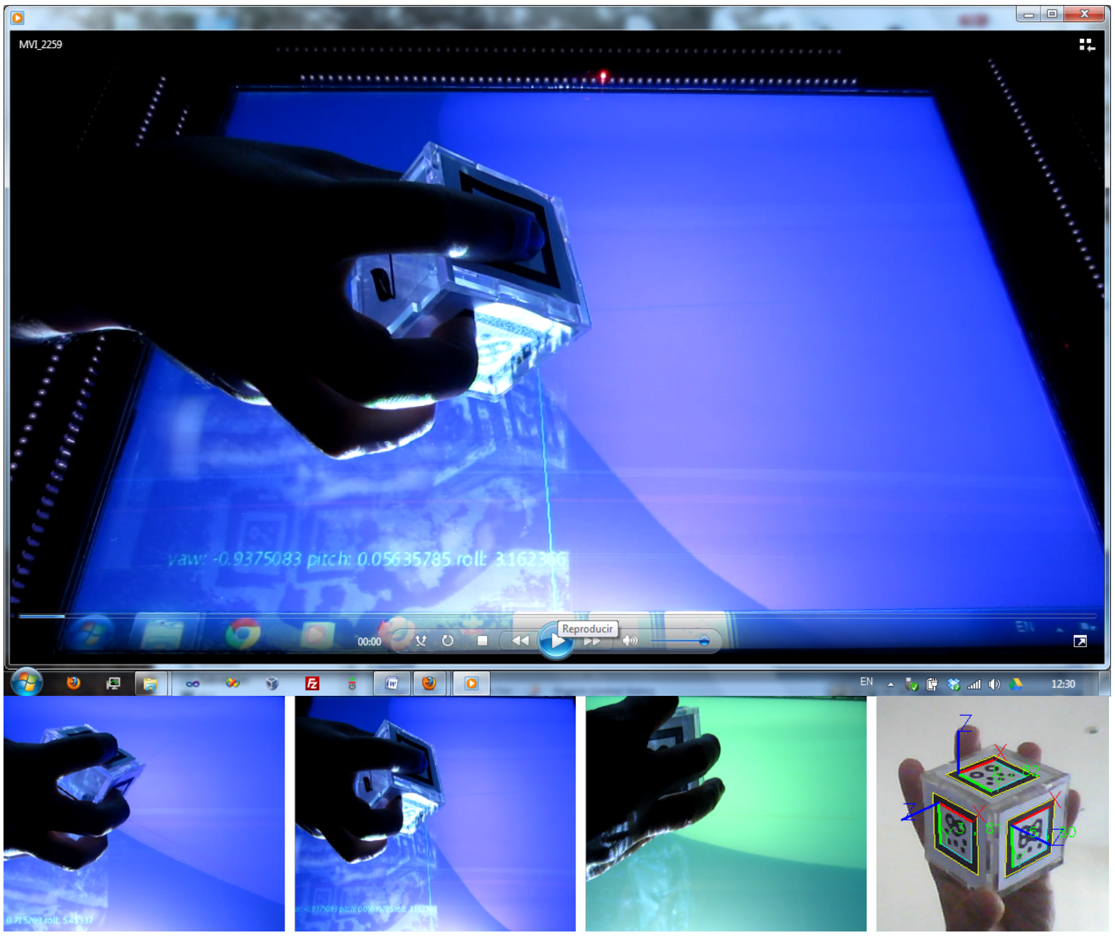
- Showing in-the-air feedback helps the user understand the actions performed. From the users' comments, we assume that, by projecting the element that had been picked or grasped on the hands, the users were better able to confirm that everything had gone well (the item selected was the correct one).
- As the picking gesture starts on the table, it is the most precise gesture; it acts only on the touched elements but it is also more complex and therefore harder to perform than the grasping gesture.
- Air pinching is the gesture that has the most balanced results for the gathering of single and multiple objects, but it is the least intuitive gesture as it only takes place in the air and requires the movement of a finger for pinching and releasing.

Photo viewer only uses hand and finger interaction. In the following demos, the use of tangible objects is introduced in order to explore the continuous interaction space by manipulating tangibles.

5.2.2 Colour Torch

Colour torch is a 3D tabletop application whereby a virtual 3D canvas is projected below the surface. This canvas can contain several 3D models that can be manipulated by catching them and releasing them using the air gestures described above. The novelty of this example is the manipulation of the light source.

The light source is represented by a physical cube with different SLFiducials attached on it; each side representing a colour. When the user moves the cube over the surface, they can light the scene with different colour beams as if they were handling a coloured lantern or a torch [Figure 67].



yaw: -0.9375083 pitch: 0.05635785 roll: 3.162366

FIGURE 67: CHANGING LIGHT SOURCE ORIENTATION BY USING A SLFIDUCIAL.

Colour Torch user test

As in the previous example, a user test to the same user set was done to demonstrate that 3D positioning cameras or light sources such as our Colour Torch demo for SL can offer better results and sensations than positioning cameras by using a mouse and cursors (like in many First person video games).

The experiment was done with the same Colour Torch application and the users were asked to find a hidden element in the darkness of the screen by using either the tangible cube of the colour torch, or the keyboard and mouse for moving and positioning a light source. Afterwards, we asked their opinion (0-5) on the following:

- Efficiency: Defining, how fast it can be used for a chosen task. In this point, 0 meant slow and 5 meant fast.

- Precision: The user’s perception of achieving an accurate pose. Where 0 was a system that perceives an unwanted pose and 5 matches exactly with what the user wants.
- Easy to use: Its intuitiveness according to the user’s opinion.

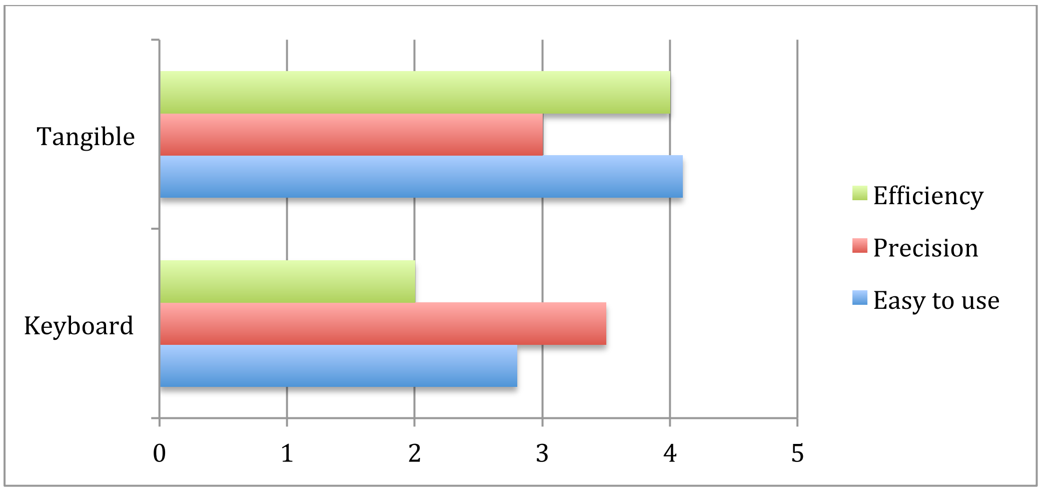


FIGURE 68: COMPARISION OF POINTING WITH A KEYBOARD AND MOUSE VERSUS USING A TANGBILE.

From the chart in [Figure 68] we can maintain that the keyboard and mouse are perceived as more precise tools compared to tangibles (for the pose task), but tangibles are more efficient and easy to be use on this application as in this application, they have been created based on natural gestures.

5.2.3 SL Theremin

A Theremin is a touch-less musical device consisting of two antennas that sense the relative position of the Thereminist's hands, and control an oscillator to detect frequency with one hand and amplitude with the other. In this SL application, we have mapped all the parameters into a single marker in order to explore expressivity with only one object: amplitude is mapped on the Z of the SLFiducial and frequency at its yaw angle. In addition to the traditional Theremin, where only two parameters are controlled, we also mapped two extra effects onto the roll and pitch angles: a tremolo and a reverberation [Figure 69]. The SL Theremin allows the use of two oscillators at the same time. Holding and interacting with more than two objects simultaneously cannot be controlled by a single user and nor placed them in the reduced surface space of the SL.

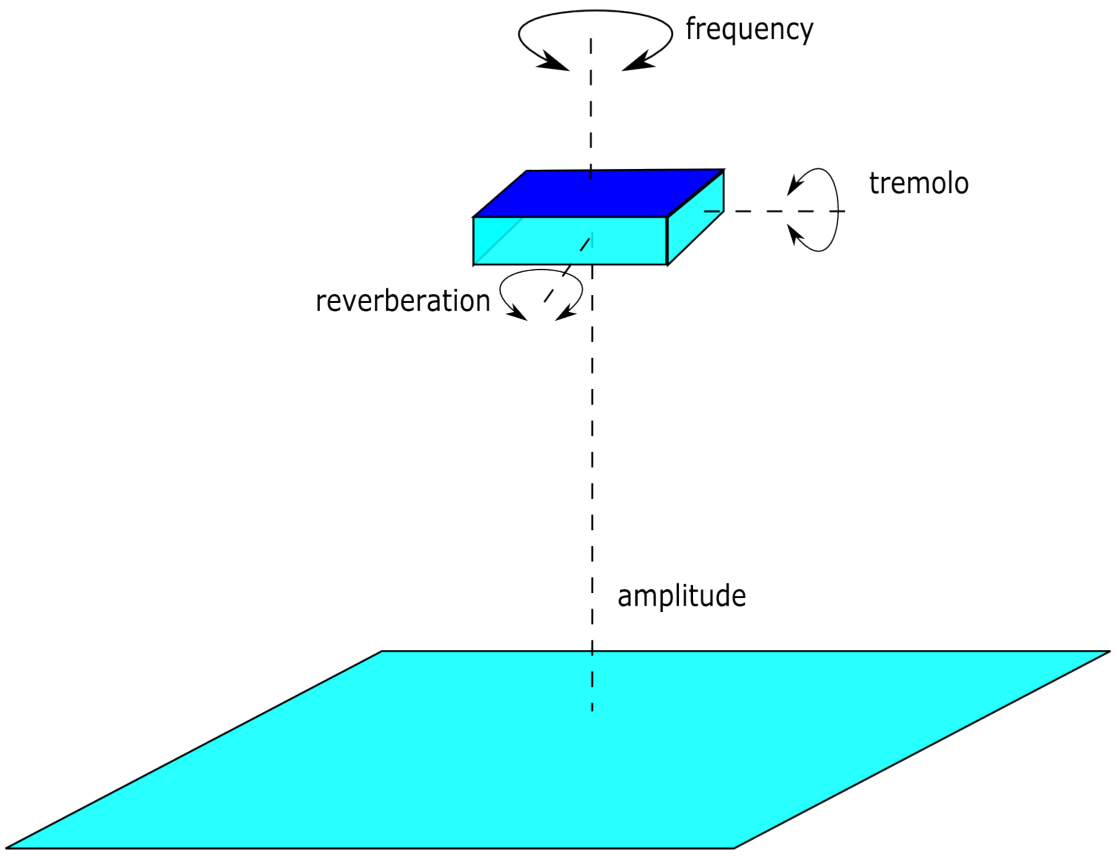


FIGURE 69: THEREMIN CONTROL MAPPED ON A SLFIDUCIAL.

On the Reactable, the oscillator's frequency is mapped onto the rotation angle, tremolo is mapped onto the distance from the table's centre and the amplitude is controlled with a slider, thus resulting in a two-hand interaction [Figure 70].

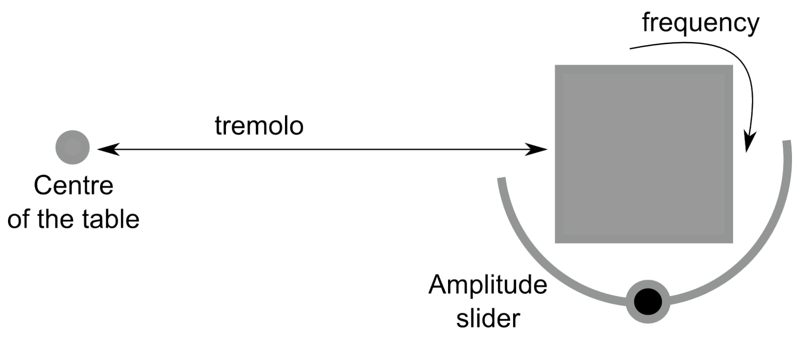


FIGURE 70: REACTABLE OSCILLATOR'S PARAMETERS.

Only one hand is needed in our SL Theremin to control the same reactable's oscillator parameters plus a reverberation effect by manipulating an object in the air. In contrast, always-on systems like the SLTheremin have often been criticised for being too stressful for the user or performer (S Jordà, 2007). As a matter of fact, in what can be described as the "always-on-syndrome", performers tracked by a camera-based system can hardly escape from being permanently analysed. A large advantage of the Reactable's tangibles is that they can be left unattended on the table while the performer is manipulating other parameters. This is contrary to the SL Theremin, where the user must hold the tangibles all the time in order to play music.

SL allows these two kinds of tangible interaction, and these have been found to be useful in different contexts. While with the in-the-air tangible interaction the user could manipulate different parameters at the same time, the tangible surface interaction is less stressful to the user because the "always-on-syndrome" is not present.

5.2.4 MAP DEPTH NAVIGATION

This map viewer demo uses an external display to view different layers or variations of a map. While on the surface there is the representation of a terrain map, other perspectives can be obtained by moving the external display vertically.

The distance from the table to the external display (Z) is adjusted in order to view different additive perspectives when the display moves further away from the surface (e.g. terrain, vegetation, streets, urban transports, shops and points of interest etc.).

This application uses traditional surface controls for translating, rotating and scaling a map. Furthermore, with the incorporation of a virtual display acting as an extended magic lens, this map viewer can show different perspectives of a virtual map without having to access menus and only moving up and down an external display.

5.2.5 VOLUME SLICING DISPLAY

Unlike the previous application, this one uses an external display for continuous volumetric data exploration such as in Cassinelli's slicing display(Cassinelli & Ishikawa, 2009). In fact, the display is used to slice a 3D model that is virtually placed above the screen, thus allowing the user to reach "cuts" of different angles and positions of the three-dimensional model.

Internally, this application reconstructs the volumetric data taken from hundreds of images of a sliced head taken from “the visible human project” [51] and renders them for showing a virtual 3D model of a head beyond the surface [Figure 71, right]. The external display is linked to the graphics renderer virtual camera. When it is moved or rotated, the camera is also displaced and re-oriented. In this manner, it is possible to see rendered real-time sliced images that will be shown above the surface on the external displays generated by the cameras near the clipping plane [Figure 71, left].

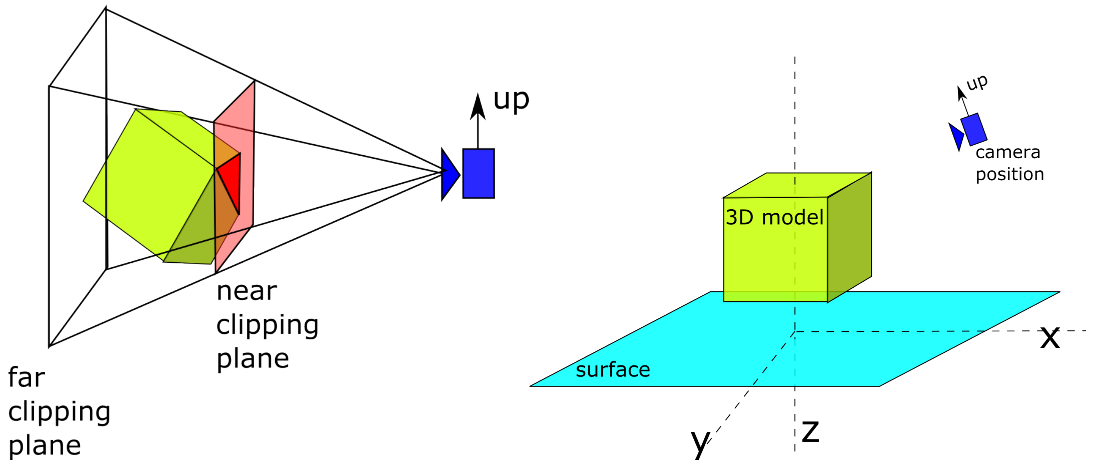


FIGURE 71: REPRESENTATIONS OF THE SLICED DATA ACCUISITION (LEFT) AND THE MOVABLE CAMERA ON THE SECONDLIGHT SURFACE POINTING THE THREE-DIMENDSIONAL MODEL (RIGHT).

Other controls that are included in this application for rotating the 3D model and fixing the cut slice on the surface to view other perspectives are implemented as on-surface gestures or mapping actions onto other tangibles . The orientation of the 3D model (yaw, pitch and roll) can be modified in order to generate “cuts” that would otherwise be impossible to obtain, for example, those upside-down or perpendicular to the surface. Using the same cube employed in colour torch, a cube with different SLFiducials attached at each face, the user can vary the orientation of the virtual model while keeping the feedback given by the physical properties of the cube and viewing a projection of the side of each model (elevation, plan and profile) at the cube’s faces.

[51] http://www.nlm.nih.gov/research/visible/visible_human.html

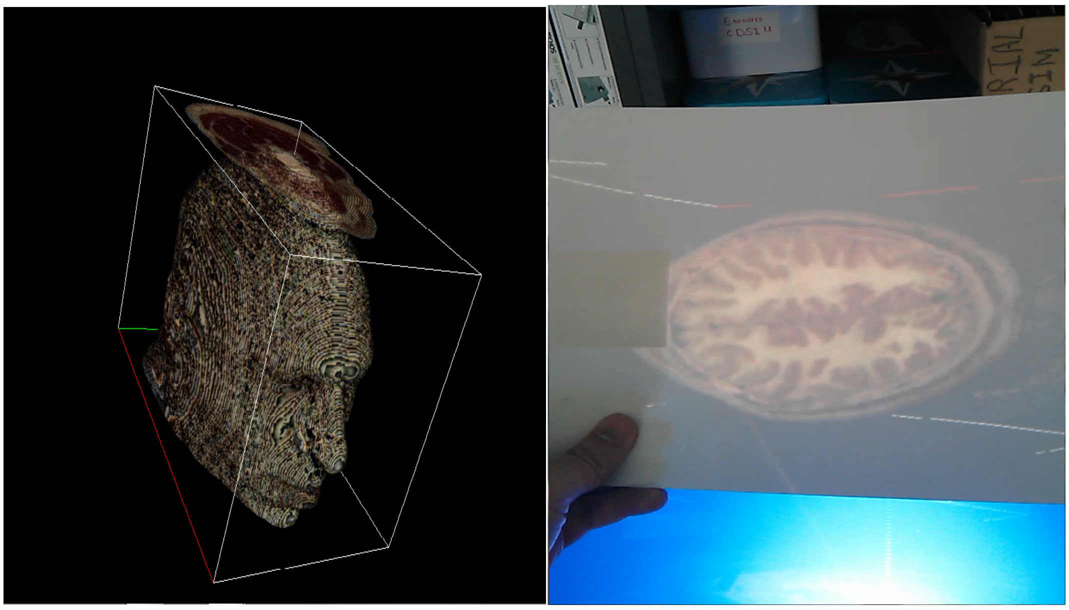


FIGURE 72: RENDERED HEAD (LEFT); A SLICE GENERATED WITH AN EXTERNAL DISPLAY (RIGHT).

5.2.6 SUMMARIZING SL DEMO APPLICATIONS

In this section we have presented some demo applications, some of which have been tested based on the users' subjective opinion. We have not conducted quantitative user tests (timings, precision, accuracy) because we were interested in the users' opinion when performing some gestures on our interface, rather than the efficiency of our gesture and graphic system.

From these demo applications it is possible to conclude that this framework, and the previous hardware and software developments, meet our objective to extend tabletop interaction to the third dimension and considerably increase the user bandwidth in comparison to the traditional tangible tabletop interaction. Furthermore, the overall system includes 6DoF tangible tabletop interaction beyond the display, which is a novelty in contrast to the existing 3D tabletop interfaces.

New gestures on the surface, above the surface and in the continuous interaction space have been generated, including ones that can be performed with tangibles. In the following section we introduce a new gesture classification that can be used as a gesture guideline for future developments on this platform.

5.3 BEYOND THE TABLETOP GESTURES

Referring back to the 2.3.2 In the air tabletop gesture affordances from Chapter 2 where the continuous interaction space by Marquardt was introduced, in this section we want to go a step further and enlarge Marquardt's gestures by adding the newly generated hand gestures from our SL interface and the new tangible interaction gestures that our interface can offer.

Tangible interaction has been seen in many tabletop interfaces, for example, the Reactable, Turtan, Microsoft surface applications, etc. However, tangible interaction above the surface of a horizontal display, and using the tangibles in the continuous interaction space has not yet been analysed (except in AR applications).

5.3.1 EXPLORED GESTURES AND METAPHORES

From the aforementioned example demo applications we can affirm that the SL surface with the various modifications and the SLVision tracker are suitable for extending surface interaction to beyond the screen. These applications also introduce a new set of gestures and metaphors that were very difficult or impossible to achieve with a traditional tabletop surface. In addition, these gestures (listed below) introduce a new and richer gesture vocabulary that, if used adequately, can benefit the communication between users and machines.

- Photo viewer uses touch gestures on the surface as conventional tabletops but enriched with additional information to position the images on the user's side extracted from the SLVision's finger-hand association. It also introduces in-the-air gestures by combining several visual feedbacks according to the precision of the gesture.
    - Multi-touch gestures used: Move, pinch, rotate, grouping and double tap.
    - "In-the-air" gestures used
        - Grasping: This is the coarsest gesture because it acts on all the virtual objects that are placed below the hand.
        - Air-pinching: This is more precise than the grasping gesture because an air-pinch is reduced to a controlled area

        but it is large enough for it to be mistaken for the selection of a single element on the surface.
        - Picking: This is the more precise gesture for selecting an element and moving it above the surface because it directly implies touching the virtual element before grasping it.

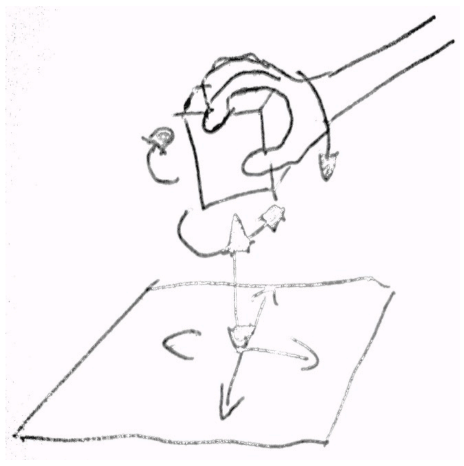
- Colour torch uses the Photo viewer "in-the-air" gestures in order to manipulate virtual 3D models represented below the surface, and introduces a 6DoF tangible interaction for lighting the scene.
    - In–the-air gestures used:
        - Grasping, air pinching and picking.
        - Marker pointing (light source pointing), which is used for virtually displacing and orientating a light beam at the fiducials' position.
- SL Theremin uses a 6DoF tangible for mapping its coordinates and angles onto an oscillator in order to play an oscillator and extends the user musical expressivity while she is holding only one marker in the air.
    - In-the-air gestures used:
        - Marker parameter mapping.
- Map depth navigation shows a terrain map on its surface where interaction can take place by performing surface gestures or can be used for map information visualization by using an external display.
    - 2D gestures used: Move, pinch and rotate.
    - 3D gestures used:
        - Discretized depth mapping.
- SL-volume slicing display shows different cuts of a virtual 3D model by moving an external surface in the air and allows the rotation of the model by spinning a cube in the air.
    - 3D gestures used:
        - Cutting plane and camera positioning.
        - Model pose manipulation (cube).

5.3.2 Tangible interaction beyond the screen

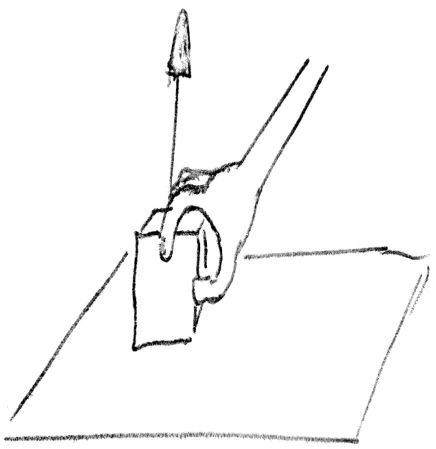
As shown in the demo applications, the new SL, with our modifications and software developments on both the coding framework and the computer vision, is able to detect and track the position above the surface of a set of tangibles marked with an SLFiducial. This tangible interaction above the surface introduces a new paradigm on tangible tabletop interaction as that presented by Marquard on the continuous interaction space with hand and touch gestures. In the following lines some gestures that can be detected by our system using tangibles in-the-air into a continuous interaction space are explained:

Controlling multi-parameter systems

This gesture is based on mapping each of the 6Dof parameters that the SLFiducial provides into a multi-parameter controlled system. In SL Theremin, this gesture is used for controlling different parts of an oscillator. It is important to note that this kind of interaction has to be carefully chosen because the "always-on-syndrome" may tire the user.

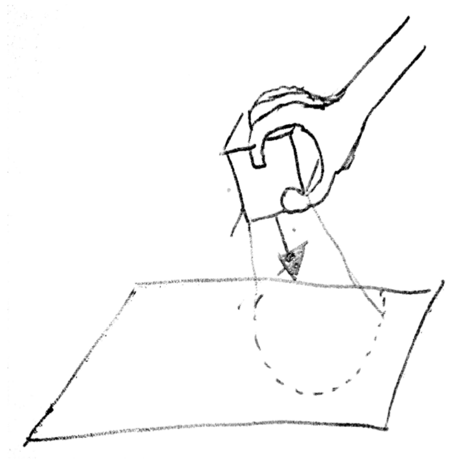


Object Pickup



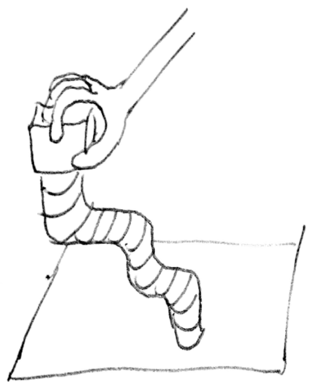
When using tangible objects on the surface, some applications such as Turtan(Gallardo et al., 2008) or Tangible Jukebox (Gallardo & Jordà, 2010) can store virtual elements inside. As in the gestures of grasping, pinching and picking performed with the hands, it is possible to use these objects as a magnet for transporting virtual data above the surface.

Air pose estimation



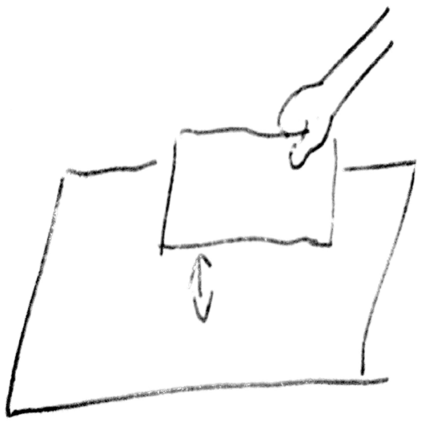
By mapping virtual tools into tangibles that can be tracked in the air and obtaining the real position of the tangible, it is possible to produce a set of tools that can be managed as if they were real tools by performing real-life gestures. Camera, torch or laser pointer actions can be easily mapped into a tangible. This is the case of the Colour torch SL demo application, whereby a coloured torch was mapped into a tangible.

3D drawing tool or extrusion doodler



Being able to track the X, Y and Z positions and the angles of different tangibles above the screen in our platform, we can use the obtained information to draw in a 3D space as if it were a 3D canvas. This gesture has been used in (De Araújo et al., 2013) in order to draw volumetric models by using fingers. However, given the tracked data by our SLVision we can use tangibles as shape extruders to create various 3D-shaped volumes.

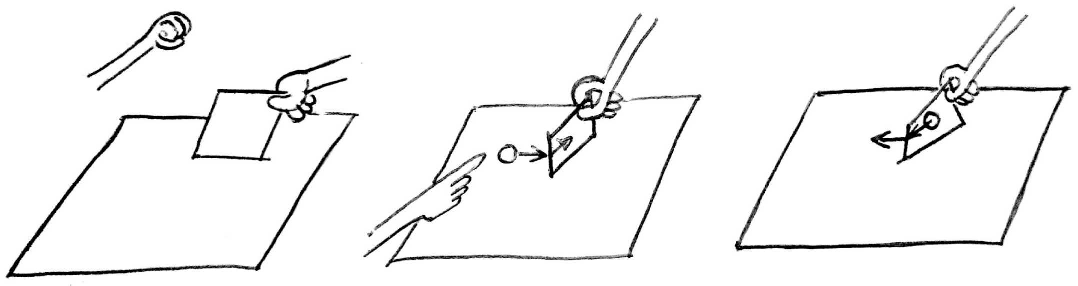
Multi-layered external display



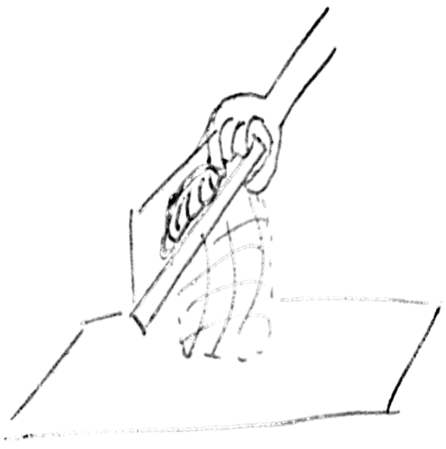
This gesture has been introduced in many publications (Marquardt et al., 2011; Subramanian et al., 2006). It consists on differentiating the distance from an external display in a reduced set of levels. As an example, we built the Map depth navigation demo where different views of a map were exposed depending on the distance of the screen from the table surface.

Virtual data manipulation onto external displays

Manipulating virtual data above the screen can be done by using hand gestures, object picking gestures or other mechanisms. These gestures imply object manipulation, but instead of using tangibles or hands for grasping elements, we can use external displays as trays to place and transport virtual data. Three different approaches are presented below; two for collecting data and one for releasing it as can bee seen at the images below.



Volume slicer



As the one used in the SL Volume slicer display demo application, this gesture consists in moving loose an external screen above the surface where a virtual 3D model representation is supposed to be placed. When the external screen intersects the model, a slice on the external screen is seen performing cutting gestures from any angle and in any position.

5.3.3 Creating classification categories

Putting together the different existing 3D tangible tabletop gestures and the new ones produced by our new SL system, enough candidates exist to start a gesture classification chart and to fit them into a specific category according to the interaction space where they act.

In this thesis, we propose to add a new gesture category between tabletop surface interaction and Marquardt's continuous interaction for the classification of gestures that are performed in the air or on the surface with some 3D interaction affordances. We have decided to name this category as Surface 2.5D interaction as an analogy to videogames where 2.5D ("two-and-a-half-dimensional") and pseudo-3D are terms, mainly in the video game industry, used to describe 2D graphical projections and similar techniques used to cause a series of images (or scenes) to simulate the appearance of being three-dimensional (3D) when in fact they are not.

While some 3D tabletop applications such as (Hancock et al., 2009), (Baudisch et al., 2010) and (Dalsgaard & Halskov, 2012) use only touch and tangible interaction on the surface by manipulating a 3D virtual canvas or augmenting tangible interaction, which results in 2D gestures on a horizontal surface, other tabletop gestures [(Hilliges, Izadi, Andrew, Hodges, & Armando, 2009), (Wilson & Benko, 2010) & (Subramanian et al., 2006)] explore the space above the surface. In (Marquardt et al., 2011) the space between surface interaction and above the surface interaction is explored. Importantly, none of these examples use tangible interaction (Subramanian only uses external displays) above the surface or in the continuous interaction space. The use of these gestures with tangibles above the surface is novel in the field of tabletop applications.

Our classification process starts from the very basic single touch surface interaction and moves onto the multi parameter in-the-air continuous interaction by applying three incremental steps:

- Traditional surface interaction.
- Two-state and discrete interaction.
- In-the-air continuous interaction.

Traditional surface interaction (2D gestures) uses the same tracking resources as a current tabletop device such as the Reactable. Our system can track objects and fingers to report 2D data. Therefore, almost any tabletop application can be carried out on the SL. It is important to note that, because the surface of SL is not large enough to detect more than 10 tangibles simultaneously, applications such as the Reactable, which sometimes require interactions with more than 10 tangibles, cannot be used on SL.

Our system differs from other traditional tangible tabletop surfaces because of the additional functions it offers. Interactions based on transparent tangibles, touchable tangibles and other magic lenses were included, by default, in the SL. Particularly novel, due to the data tracked by SLVision, is the possibility to track the user's position, hand-finger relationship and finger direction, even though finger direction is included in some systems such as PixelSense. By tracking the user's position, the system can orientate the virtual information to the user's side of the tabletop, thus losing the orientation restrictions imposed by a rectangular surface (up, down, left and right) and being able to distinguish the distinct touches of its users. Finger tracking includes hand identification; by applying finger-hand information to the various gesture analysers we can enrich finger gesture vocabulary. This is achieved by differentiating between two hand touch gestures on the surface (i.e. zooming with two hands, each using one or multiple fingers) from one hand interaction (surface pinch gesture) or assigning a different role to each hand touch (i.e. the fingers of one hand draw and those of the other erase).

Two state and discrete interaction (2.5D gestures) only uses the gathered data beyond the display for hands and fingers and discriminates the in-the-air distances between the surface and the different tangible objects.

In-the-air hand and finger gestures and hand-pose interaction are included in this stage of the classification process. These gestures, more related to an AR system than to tabletop interaction, lose the physicality of manipulating things by touching or grasping and directly collides with the concept of direct manipulation. It is not possible to provide tangible feedback beyond the display but, due to the second projection of the SL surface, we can project images on hands and fingers to offer at least visual feedback beyond the surface. The precision of these gestures for virtual object location on the surface is poor, but displaying the hand or finger shadow on the surface can enhance precision.

Tangible interaction in 2.5D can be divided into two groups: discrete interaction and two-state interaction. Discrete interaction, mainly adjusted for use on the external displays, is the tangible interaction where the Z space is divided into different portions and the information presented can differ depending on which part (height) is the external display in relation with the surface. Differently, two-state interaction also uses this criterion to create subdivisions, but these are reduced to two states: on surface and beyond the surface. This kind of tangible interaction, which we have called “complex tools”, is mainly adjusted to tangibles with special affordances that require two distinct contexts for functioning, for example, configuration and to use a tool.

In-the-air continuous interaction (3D gestures) is achieved by making gestures that start on the surface and continue beyond it. In this classification system, the continuous tangible interaction is also included, which uses all the tracked parameters without subdividing them.

Hand and finger gestures in the continuous interaction space, in contrast to 2.5D gestures, are more precise when selecting the virtual elements of the surface since these gestures start from the direct touch of the virtual element on the surface. In this case, the representation of a virtual shadow for surface location is not needed, but when these gestures are above the surface, they have the same problem as 2.5D air gestures and some form of projection feedback is needed.

The use of 6DoF fiducial markers introduces three new degrees of freedom and a precise pose orientation, thus allowing the direct mapping of virtual tools onto their position (a torch, a slicer camera) or the control of various parameters with the same object (e.g. SL Theremin).

5.3.4 DEFINING THE WHOLE GESTURE CLASSIFICATION

By including these three aforementioned incremental steps as the root of our new classification chart, we can enhance the classification system with multiple variations that distinguish between touch and hand interaction, tangible interaction and other subcategories that require the use of a concrete metaphor [Figure 73].

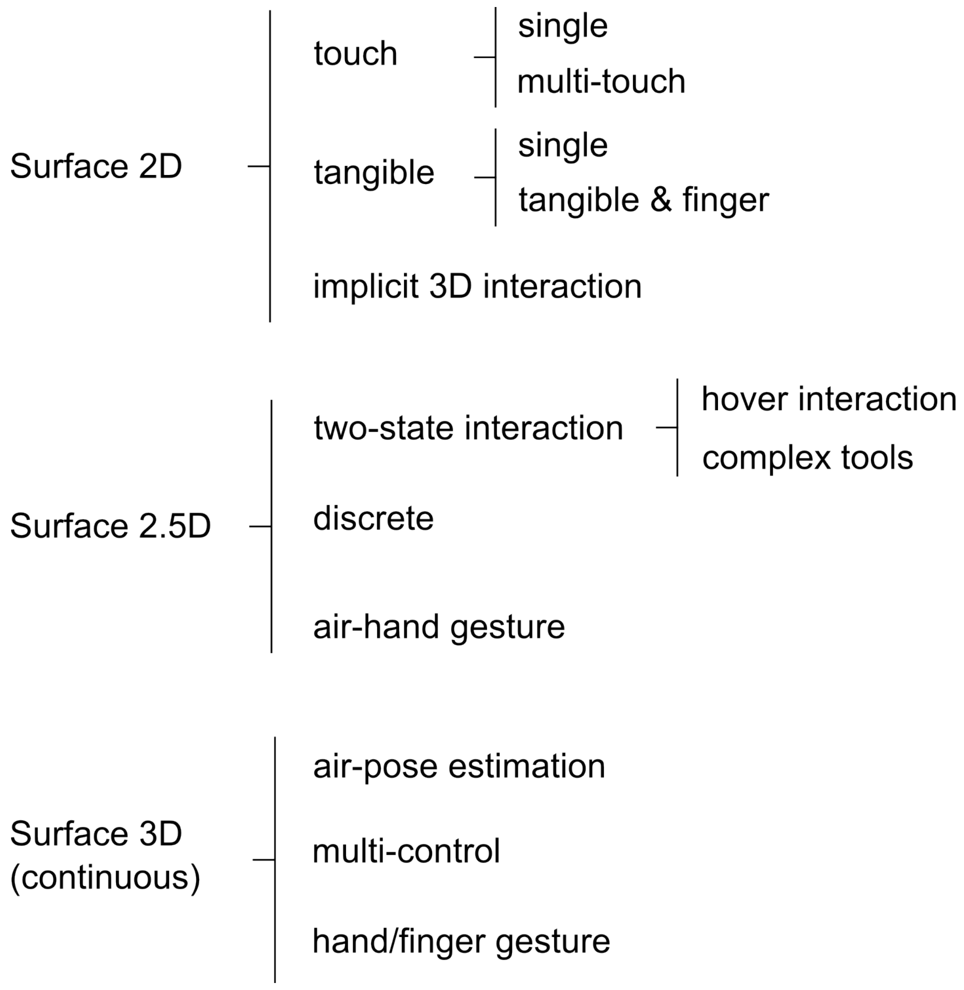


FIGURE 73: 3D TANGIBLE TABLETOP GESTRUE CLASSIFICATION.

In the following lines each classification point of this chart is explained.

5.3.5 Surface 2D interaction

This classification point covers all on-the-surface interaction and the gestures are classified into three sublevels depending on the object used: fingers, tangibles or complex multi-touch, or tangible gestures.

Single touch interaction can be compared to WIMP interaction as it is mainly used for menu navigation and button tapping. Among its gestures we can find: tap, double-tap, tap-and-drag or more complex gestures requiring time, such as tap and hold or tap and throw, in order to give a direction and an acceleration to a certain virtual object. By using all the SLVision data, these gestures can be identified with a user position, assigning certain controls to a specific user, or the gesture can be associated with a hand to extract finger orientation [Figure 74].

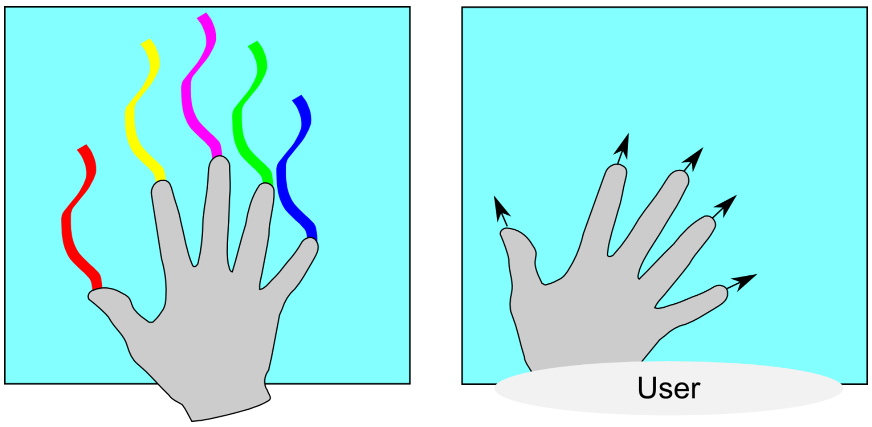


FIGURE 74: SINGLE-TOUCH GESTURES WITH DIFFERENT SL EXTRA INFORMATION: (LEFT) HAND-FINGER IDENTIFICATION; (RIGHT) FINGER ORIENTATION AND USER LOCATION.

Multi-touch interaction comprises all gestures that involve more than one finger in order to be performed. Gestures such as pinch and rotate will be included in this category, but similar to single touch, they can be extended with the additional information provided by the SL to generate more advanced multi-touch gestures by identifying touches made with different hands (ie. Pinch with two fingers of different hands or grouping hand fingers to rotate and zoom).

Single tangible interaction is the surface interaction that involves the use of tangibles to physically manipulate the data. The gestures in this classification point can access to the tangible identification, as well as the X and Y positions and the Z angle. Taking advantage of what SL can provide, which entails the use of the second projection when the tangibles are transparent, tangibles can be employed to view data on its surface augmenting the tangible. In addition fingers can be used to interact on the tangible's surface resulting in a new category that we have called **tangible & finger interaction** where touch gestures are performed onto the tangibles like in (Baudisch et al., 2010).

Implicit 3D interaction is the 2D interaction gestures that entail some actions in the air, although these gestures are not tracked beyond the surface. These gestures are the ones that result from trying to "imitate" 3D tangible tabletops with traditional tabletops. Examples are walking hands or tangible feinting that involves some kind of implicit 3D interaction (explained at 2.2.3).

5.3.6 Surface 2.5D interaction

The gestures classified in this level are those that implicate interaction beyond the screen when the surface and air gestures are strictly separated or when the depth data is not treated as a continuous variable.

Two-state gestures are all gestures whereby the interaction takes place in two different spaces (on the surface and "in the air"), but without the presence of continuity among them. Given the complexity and multiple gestures that can be performed and placed under this denomination, this sub-level has been divided into two different groups:

- **Hover interaction gestures** are above the surface gestures (mainly pointing) that cue the user to notify that something can be done before touching the surface. This includes examples such as glowing buttons when a finger is over them without touching them, the appearance of contextual help in certain interaction areas, or indicating that an area can interact with the object that the user is holding before placing it on the surface.
- **Complex or configurable tools** are two-state gestures that differentiate interaction in the air from surface interaction. It is common in configurable tangibles that, while in the air they are in a configuration mode, when they are on the surface (working area) they act as tools with the configured parameters.

Discrete air gestures groups all gestures that function above the surface and that use the depth data as a non-continuous variable, thus reporting a layered or discrete depth data. The gestures that can be found in this level have external displays that show different contents depending on the distance from the surface, such as in the aforementioned depth map navigation example, or finger interaction that is performed onto the external displays.

Air hand gestures are hand and finger gestures that controls virtual objects on and above the surface without touching them. These gestures are more related to augmented reality applications than tabletop ones. In air hand gestures the physical components of the tangible elements (tangibles and surface) and the concept of direct manipulation cannot be applied. As a consequence, these gestures frequently need the representation of some kind of visual feedback, for example, shadows on the surface or in-the-air hand projection. In this set, we can find gestures such as those explained in the description of the photo viewer application, namely air-pinching and grasping, but in this group all hand shape gestures and in-the-air finger pointing (Carreira & Peixoto, 2007; Epps et al., 2006; Sutcliffe et al., 2013) can also be included.

5.3.7 Surface 3D interaction

These are the gestures whereby surface computing and in-the-air computing converge in a continuous manner. However, it is also possible to include gestures that involve a continuous representation of depth data in the air, as in the case of the 6DoF markers of SL.

Air-pose-estimation gestures are the gestures that use the 6Dof tangibles for in-the-air object pose calculation. Using an SLFiducial attached to a physical object (tangible), the system can know the coordinates and angles of that object beyond the surface. Depending on the object that is being controlled, these gestures can be of a “mapped” or “physical” nature.

- Air-pose estimation **mapped** gestures use the marker information for rotating and translating a virtual object. Those gestures used for positioning the head by rotating a cube in the air in the volume slicing display application can be part of this group, as well as any gesture that uses coordinates to position data rather than absolute coordinates to link the data with the exact position of the marker.
- Air-pose estimation **physical** gestures use the position of the marker to place an effect or data in the space occupied by the marker. As an example, in this section we can find the colour torch, whereby the tangible becomes a beam of light, or in the volume slicing display, whereby the manipulation of the slicer tool (portable screen) results in the movement of a cutting plane in order to perform real-time cuts of a virtual model.

Multi-parameter control gestures take advantage of the multiple parameters extracted from SLFiducials to control non-3D data such as multiple parameters of a controller. An example is the SL Theremin, whereby using a single marker controls six parameters of an oscillator.

Hand/finger continuous gestures are a mix of single and multi-touch gestures and air-hand gestures resulting on continuous hand-finger interaction starting at the surface and ending beyond it (or vice-versa). An example of these gestures is the "pick and drop" gesture, which consists in the user collecting a surface object by grasping it with the fingers and raising the hand. These gestures often entail the use of a support projection to contextualize the targeted virtual element.

5.3.8 Gestures sumarizing and guide

Summarizing all the aforementioned gestures and as well as the possible graphic feedback and their area of influence. We have elaborated two tables; the hand and finger interaction and the tangible interaction tables. These tables are divided into three columns indicating the area in which the gestures are performed: 2D, 2.5D and 3D.

Notice in [Table 11: Finger gestures summary] that 3D or continuous interaction is an addition to the 2D and 2.5D. Therefore, all feedback gestures and actions on the 2D and 2.5D areas can also be included within the 3D column. This is not the case of the tangibles in [Table 12: Tangible interaction summary]. Here the gestures and actions are not accumulative.

Hand and finger interaction			
	2D	2.5D	(2D) + (2.5D)+ 3D
Data	Finger position. Finger orientation. User position. Finger identification.	In the Air detection. Hand data. Pinch data.	
Feedback	On Screen feedback.	Contextual feedback without touching (hover interaction). On-screen hand-shadows.	In the air projected feedback.
SL implemented gestures	Tapping. Dragging.	Hover move. Pointing. Pinching. Grasping.	Picking.
Area	Surface only (Finger)	In-the-air only (Finger and hand)	Surface + in-the-air. (Finger and Hand)
Classification	Surface 2D: • Single touch. • Multi touch. • Complex (touch).	Surface 2.5D: • Hover interaction. • Air-hand gesture.	Surface 3D: • Hand/finger gesture.

TABLE 11: FINGER GESTURES SUMMARY

Tangible interaction			
	2D	2.5D	3D
Data	Position (x, x). Z-angle.	Position (x, y, discrete z) Z-angle	Position (x, y, z) Yaw, pitch, roll
Feedback	Surface feedback. Magic lens (2nd projection).	Portable screen. Projection on the tangible. Surface feedback.	Portable screens. On tangible projection.
SL implemented gestures	Move Rotate Tapping on (transparent tangibles)	Move Rotate Tapping & finger (on portable screens) Layer navigation Two-state interaction	Positioning (x, y, z, yaw, pitch, roll) Tapping & finger (on portable screens)
Area	Surface only On-tangible interaction	Portable screen Surface Discrete z-depth above the screen.	Portable screen On-the-air
Classification	Surface 2D: • Tangible single • Tangible & finger	Surface 2.5D: • Two-state interaction – complex tools • Discrete (layer interaction)	Surface 3D: • Air pose estimation • Multi-control
Pros/cons	+ Unattended tangibles - Occludes air interaction - Three degrees of freedom	+ Different interaction levels (in external displays) + Two states tangibles - Three degrees of freedom - One hand blocked (in the air)	+ Precise location + 6 degrees of freedom - Always-on interaction - One hand blocked (always)

TABLE 12: TANGIBLE INTERACTION SUMMARY

5.4 Conclusions

In this chapter we have introduced the SLFramework, a framework built for the modified SecondLight surface (Chapter3) using the SLVision (Chapter4). It facilitates the tasks of projecting images on and above the surface, implements gesture sensing areas that can be drawn by taking any 2D shape, establishes a mechanism of gesture creation by generating composed gestures, provides a finger and tangible simulator for when it is not possible to use the SL device, has mechanisms to project feedback from external objects such as portable screens and it is furnished with in-the-air shadow feedback mapping on the surface.

All the functionalities of this framework have been presented by developing five different applications that test its functions and opens a new gesture vocabulary with the expansion of surface interaction to interaction beyond the screen.

Later in this chapter, beyond-the-screen continuous tangible interaction was introduced by showing different gestures that can be tracked with our new system. Finally, we introduced a new gesture classification that is valid for all the gestures that can be tracked by our system and can be used as a beyond the display gesture guideline in further application developments.

Chapter 6 CONCLUSIONS

In this dissertation we presented the overall process of defining, building, and generating a hardware and software solution for an extended tabletop interface inspired on the recent 3D tabletop interfaces. During the development of the interface, some areas were addressed from varying perspectives: hardware, computer vision, framework development and gesture interaction.

Research on tangible and tabletop interaction is always searching new interaction metaphors that bring natural gesture language to tabletop interfaces, pushing them to adopt a more natural and fluid gesture language. In Chapter two, we introduced the definition of a 3D tabletop interface and our objective to extend tangible tabletop interaction to the third dimension in order to enhance the communication bandwidth of these interfaces beyond the display. Current research on 3D tangible tabletop interaction defines the continuous interaction space as the space encompassed by direct touch and the space above it, portraying some gestures that use this space as examples. In Chapter 2 we finished presenting our aim to enhance tabletop interaction in order to explore this continuous interaction space, and to introduce tangible interaction metaphors to that space.

During this dissertation we also presented some hardware modifications for the SecondLight that enable the tracking of hands and markers beyond the display. These modifications are needed in order to track objects above the surface because the original lighting system could not reach objects located further than about twenty centimetres away from the surface.

A full computer vision solution was developed for tracking 6DoF markers, touch and hands both on and above the surface. This solution, named SLVision, comprised the designing and development of new special markers based on topological adjacency regions that can be tracked in the 3D space. SLFiducials, the new markers, were specially designed in order to fulfilling our need for 6DoF markers that could be tracked with the SecondLight tabletop from one metre above the surface and be as small as possible, keeping a balance between size and ID-range. These markers were tested and it was demonstrated that they met our conditions in terms of size and ID-range. In addition, they can be tracked faster than similar competitors while providing 3D pose data.

Closing this dissertation, a framework for developing 3D tangible tabletop applications on the SecondLight was presented. This framework facilitates the tasks of projecting images on and above the surface, manage gesture events and simulate SecondLight input data when this tabletop device is not available.

6.1 Contributions

Over the course of this dissertation many contributions were published in the form of publications, hardware improvements or software solutions. Other contributions are not in this dissertation, but were listed in Annex 1 regarding tangible programming languages, metaphors on tabletop interfaces and horizontal surface strategies.

6.1.1 Published contributions

- With **MTCF** we developed the basic framework for developing tangible tabletop applications without the need to learn a complete, complex programming language. MTCF changed the development cycle of tabletop programming from compile, upload and test to real time tabletop coding.
- **mMTCF** is the mobile version of the aforementioned MTCF. It has the same developing cycle to that of real-time coding, but it is adjusted to function on tablet and smartphone devices, thus enabling the access to all embedded sensors of these “ever-present” devices.
- **SLFiducials** are new 6DoF tags built for 3D tabletop interaction. They were especially designed to be as compact as possible and to be detectable with a “low resolution” camera when up to 170 centimetres away.

6.1.2 ORIGINAL CONTRIUTIONS

- We explored how to expand tangible tabletop interaction by using special tangibles with embedded electronics, as shown in the active Cube hardware.
- We introduced some modifications to the SecondLight device in order to extend its tracking range to up to 100 centimetres away from the surface.
- We defined and developed a full computer vision software that was built for tracking hands, fingers and markers on and above the surface.
- We provided the definition and described the development of a full framework for the SecondLight that was based on our previous work with tangible tabletop interfaces.
- We defined a new 3D gesture classification system, adding the new possible categories generated by the overall process of this thesis.

6.1.3 SOFTWARE CONTRIBUTIONS

- OfxTableGestures is a framework for tangible tabletop interfaces built to aid the task of developing tangible applications and exploit the distinctive tabletop characteristics. It is publicly available under an open source license and is currently being used.
- MTCF is a framework for the rapid prototyping of tangible applications on a tabletop interface. Originally created for music tangible tabletop application development, this framework is being used as fast tabletop prototyping language. It is publicly available under an open source licence and currently is mainly used to teach tangible tabletop interaction.
- mMTCF is the mobile version of MTCF. It has been released as an android application in Google Play, whereby users can program (real-time) their applications by connecting the android device to the computer or they can upload their own programs such that they run as standalone applications. The source code has not yet been published.
- SLVision is the computer vision software developed for the SecondLight. It includes the work presented in Chapter 4 and is fully compatible with any client application that listens OSC TUIOMessages. It is publicly

available under an open source license and is currently being used on other SL interfaces.

- The SLFramework is the framework presented in Chapter 5. It comprises all the developments described in that chapter, including projector management and second display access methods. It has not yet been publicly released because it is in the process of licence definition.

6.2 FUTURE WORK

Multiuser interaction has been discarded because of the reduced interaction space of the SecondLight interface. It Chapter 3 has been commented that camera and projector-based tabletops can be scaled by using a bigger surface and multiple cameras and projectors. SecondLight has the PFCT surface, which is an extremely fragile hardware that can be transparent when it is driven with a 200V DC. This large amount of DC current is needed in order to change the state of the surface from transparent to translucent as fast as possible such that it remains synchronized with the 3D projector. By using a larger surface, the amount of DC to be driven through the bigger PFCT will also increase and the overall hardware and safety mechanisms would need to be revised.

Another possibility is to explore other 3D tabletop approaches. In Chapter 2 we introduced the use of the mMTCF as a way of extending traditional tangible tabletop interfaces. Furthermore, in Chapter 3 we showed how to use an accelerometer-based token to track tangible pose estimation above the table. A mobile device has multiple sensors such as accelerometers, gyroscopes, compass and altimeter, among others. By using mMTCF, we can easily access to the sensor data. In Chapter 3 we demonstrated, by describing the active cube, that it was not possible to track the absolute position of a tangible above the surface (only the 3D angles) but, by using mobile device sensors such as the rear camera, we were nonetheless able to access the data and to extend any horizontal surface beyond the screen.

BIBLIOGRAPHY

Baudel, T., & Beaudouin-Lafon, M. (1993). Charade: remote control of objects using free-hand gestures. *Communications of the ACM*, *36*(7), 28–35. doi:10.1145/159544.159562

Baudisch, P., Becker, T., & Rudeck, F. (2010). Lumino : Tangible Blocks for Tabletop Computers Based on Glass Fiber Bundles.

Beckhaus, S., Schröder-Kroll, R., & Berghoff, M. (2008). Back to the sandbox: playful interaction with granules landscapes. *... and Embedded Interaction*, 141–144. Retrieved from http://dl.acm.org/citation.cfm?id=1347421

Bier, E., Stone, M., & Pier, K. (1993). Toolglass and magic lenses: the see-through interface. *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, 73–80. Retrieved from http://dl.acm.org/citation.cfm?id=166126

Billinghurst, M., & Kato, H. (2002). Collaborative augmented reality. *Communications of the ACM*. Retrieved from http://dl.acm.org/citation.cfm?id=514265

Bishop, D. (1992). Marble answering machine. *Royal College of Art, Interaction Design*.

Borg, K. (1990). Ishell: A visual unix shell. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 201–207).

Carreira, J., & Peixoto, P. (2007). RETRIEVING AND EXPLOITING HAND ' S ORIENTATION IN TABLETOP INTERACTION, 1003–1006.

Cassinelli, A., & Ishikawa, M. (2009). The Volume slicing display. *ACM SIGGRAPH ASIA 2009 Art Gallery & Emerging Technologies: Adaptation on - SIGGRAPH ASIA '09*, 88. doi:10.1145/1665137.1665207

Costanza, E., & Robinson, J. (2003). A Region Adjacency Tree Approach to the Detection and Design of Fiducials. *Video Vision and Graphics 03*. Retrieved from http://eprints.ecs.soton.ac.uk/20958/

Czernuszenko, M., Pape, D., Sandin, D., DeFanti, T., Dawe, G. L., & Brown, M. D. (1997). The ImmersaDesk and Infinity Wall projection-based virtual reality displays. *ACM SIGGRAPH Computer Graphics*, *31*(2), 46–49.

Dalsgaard, P., & Halskov, K. (2012). Tangible 3D tabletops: combining tangible tabletop interaction and 3D projection. *Proceedings of the 7th Nordic Conference on …*, 109–118. Retrieved from http://dl.acm.org/citation.cfm?id=2399033

Davidson, P., & Han, J. (2006). Synthesis and control on large scale multi-touch sensing displays. *Proceedings of the 2006 Conference on New …*. Retrieved from http://dl.acm.org/citation.cfm?id=1142269

De Araújo, B. R., Casiez, G., Jorge, J. a., & Hachet, M. (2013). Mockup Builder: 3D modeling on and above the surface. *Computers & Graphics*, *37*(3), 165–178. doi:10.1016/j.cag.2012.12.005

De Araújo, B. R., Casiez, G., Jorge, J., & Hachet, M. (2012). Modeling on and above a stereoscopic multitouch display. *3DCHI-The 3rd Dimension …*. Retrieved from http://hal.inria.fr/hal-00670565/

Dietz, P., & Leigh, D. (2001). DiamondTouch: a multi-user touch technology. *… Symposium on User Interface Software and Technology*, *3*(2), 219–226.

Eberly, D. (1998). Triangulation by ear clipping. *Geometric Tools, LLC. Http://www. Geometrictools. Com*, 1–13. Retrieved from http://omploader.org/vMWYzcg/triangulation byt ear clipping - AD501C84d01.pdf

Epps, J., Lichman, S., & Wu, M. (2006). A study of hand shape use in tabletop gesture interaction. *CHI '06 Extended Abstracts on Human Factors in Computing Systems - CHI EA '06*, 748. doi:10.1145/1125451.1125601

Falcão, A. (2004). The image foresting transform: Theory, algorithms, and applications. *Pattern Analysis and …*, *26*(1), 19–29. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1261076

Fiala, M. (2005). ARTag, a fiducial marker system using digital techniques. *Computer Vision and Pattern Recognition, 2005. …*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1467495

Fiala, M. (2005). Comparing ARTag and ARToolkit plus fiducial marker systems. *IREE International Worksho on Haptic Audio Visual Environments and Their Applications, 2005.*, 147–152. doi:10.1109/HAVE.2005.1545669

Fitzmaurice, G. W., Ishii, H., & Buxton, W. A. S. (1995). Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 442–449).

Gallardo, D., & Jordà, S. (2010). Tangible jukebox: back to palpable music. *… of the Fourth International Conference on Tangible, …*. Retrieved from http://dl.acm.org/citation.cfm?id=1709922

Gallardo, D., & Julià, C. F. (2007). TDesktop : Disseny i implementació d'un sistema gràfic tangible. *UPF Computer Science Final Degree*.

Gallardo, D., Julià, C. F., & Jordà, S. (2013). Using MTCF for live prototyping on tablet and tangible tabletop devices. *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction - TEI '13*, 443. doi:10.1145/2460625.2460724

Gallardo, D., Julià, C., & Jordà, S. (2008). TurTan: A tangible programming language for creative exploration. *Tabletop*. Retrieved from http://mtg.upf.edu/node/1081

Gaver, W. W., Bowers, J., Boucher, A., Gellerson, H., Pennington, S., Schmidt, A., … Walker, B. (2004). The drift table: designing for ludic engagement. In *CHI'04 extended abstracts on Human factors in computing systems* (pp. 885–900).

Genest, A., & Gutwin, C. (2011). Characterizing Deixis over Surfaces to Improve Remote Embodiments, (September), 24–28.

Grossman, T., & Wigdor, D. (2007). Going Deeper: a Taxonomy of 3D on the Tabletop. *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'07)*, 137–144. doi:10.1109/TABLETOP.2007.18

Han, J. (2005). Low-cost multi-touch sensing through frustrated total internal reflection. *Proceedings of the 18th Annual ACM Symposium on …*, 115–118.

Han, J. Y. (2006). Multi-touch interaction wall. In *ACM SIGGRAPH 2006 Emerging technologies* (p. 25).

Hancock, M., Cate, T., & Carpendale, S. (2009). Sticky Tools : Full 6DOF Force-Based Interaction for Multi-Touch Tables. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '09).*

Hilliges, O., Izadi, S., Andrew, D. W., Hodges, S., & Armando, G. (2009). Int teractions in the Air : Adding Further Depth to Interactive Tabletops. In *UIST 2009* (pp. 139–148).

Hilliges, O., Kim, D., & Izadi, S. (2012). HoloDesk: direct 3d interactions with a situated see-through display. *Proceedings of the SIGCHI …*. Retrieved from http://dl.acm.org/citation.cfm?id=2208405

Hinckley, K., Pausch, R., Proffitt, D., & Kassell, N. F. (1998). Two-handed virtual manipulation. *ACM Transactions on Computer-Human Interaction*, *5*(3), 260–302. doi:10.1145/292834.292849

Hodges, S., Izadi, S., Butler, A., Rrustemi, A., & Buxton, B. (2007). ThinSight: versatile multi-touch sensing for thin form-factor displays. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (pp. 259–268).

Hornecker, E., & Buur, J. (2006). Getting a grip on tangible interaction: a framework on physical space and social interaction. *Proceedings of the SIGCHI Conference on Human …*, 437–446. Retrieved from http://dl.acm.org/citation.cfm?id=1124838

Hoste, L., & Signer, B. (2014). Criteria, Challenges and Opportunities for Gesture Programming Languages. In *1st International workshop on Engineering Gestures for multimodeal Interfaces* (Vol. i). Retrieved from http://egmi.gispl.org/papers/Hoste-EGMI2014.pdf

Huppmann, D., Luderschmidt, J., & Haubner, N. (2012). Exploring and Evaluating the Combined Multi-Touch and In-the-Air Tabletop Interaction Space. *VR/AR*.

Retrieved from http://www.mi.hs-rm.de/~schwan/Veroeffentlichungen/docs/EvaluateInTheAir2012.pdf

Ishii, H. (2007). Tangible user interfaces, 1–17. Retrieved from http://books.google.com/books?hl=en&lr=&id=tRVRK8UhuacC&oi=fnd&pg=PA141&dq=Tangible+user+interfaces&ots=DaBqlbqS9P&sig=t5U4-KmCFIraiLidkj5Fw5dUMZ8

Ishii, H., Fletcher, H. R., Lee, J., Choo, S., Berzowska, J., Wisneski, C., … Bulthaup, C. (1999). MusicBottles. In *ACM SIGGRAPH 99 Conference abstracts and applications* (p. 174).

Ishii, H., & Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems* (pp. 234–241).

Izadi, S., Hodges, S., Taylor, S., Rosenfeld, D., Way, M., Cb, C., & Wa, R. (2008). Going Beyond the Display : A Surface Technology with an Electronically Switchable Diffuser. *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, 269–278.

Jordà, S. (2007). Interactivity and Live Computer Music. *The Cambridge Companion to Electronic Music*.

Jordà, S., Julià, C. F., & Gallardo, D. (2010). Interactive surfaces and tangibles. *XRDS: Crossroads, The ACM Magazine for Students*, *16*(4), 21. doi:10.1145/1764848.1764855

Jordà, S., Kaltenbrunner, M., Geiger, G., & Bencina, R. (2005). The reactable*. In *Proceedings of the international computer music conference (ICMC 2005), Barcelona, Spain* (pp. 579–582).

Julià, C., Gallardo, D., & Jordà, S. (2009). TurTan: Un Lenguaje de Programación Tangible Para el Aprendizaje. *X Congreso Internacional de Interacción Persona-Ordenador, Interacción 2009*.

Julià, C., Gallardo, D., & Jordà, S. (2011). MTCF: A framework for designing and coding musical tabletop applications directly in Pure Data. *Proceedings of Nime' 11*, (June), 457–460. Retrieved from http://www.nime.org/proceedings/2011/nime2011_457.pdf

Julià, C., & Jordà, S. (2009). SongExplorer: A Tabletop Application for Exploring Large Collections of Songs. *ISMIR*, (Ismir), 675–680. Retrieved from http://ismir2009.ismir.net/proceedings/PS4-17.pdf

Kaltenbrunner, M. (2009). reacTIVision and TUIO: a tangible tabletop toolkit. *Proceedings of the ACM International Conference on …*. Retrieved from http://dl.acm.org/citation.cfm?id=1731906

Kaltenbrunner, M., & Bencina, R. (2007). reacTIVision : A Computer-Vision Framework for Table- Based Tangible Interaction. *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*.

Karlsson, A. H., Young, J. F., & Christensen, M. (2013). Model-Based Object Pose in 25 Lines of Code. *Meat Science*. doi:10.1016/j.meatsci.2013.05.001

Kato, H., & Billinghurst, M. (1999). Marker tracking and HMD calibration for a video-based augmented reality conferencing system. *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, 85–94. doi:10.1109/IWAR.1999.803809

Katsaprakakis, A. (2011). *Extending the control of a musical tangible tabletop with mobile phones*. Retrieved from http://mtg.upf.edu/system/files/publications/Katsaprakakis-Alexandros-Master-thesis-2011.pdf

Kim, D., Izadi, S., Dostal, J., & Rhemann, C. (2014). RetroDepth: 3D silhouette sensing for high-precision input on and above physical surfaces. *Proceedings of the …*, 1377–1386. Retrieved from http://dl.acm.org/citation.cfm?id=2557336

Krueger, M. W. (1991). *Artificial reality II* (Vol. 10). Addison-Wesley Reading (Ma).

Krueger, M. W., Gionfriddo, T., & Hinrichsen, K. (1985a). Videoplace-- an artificial reality. *ACM SIGCHI Bulletin*, (April), 35–40. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:No+Title#0

Krueger, M. W., Gionfriddo, T., & Hinrichsen, K. (1985b). VIDEOPLACE—an artificial reality. In *ACM SIGCHI Bulletin* (Vol. 16, pp. 35–40).

Krueger, W., & Froehlich, B. (1994). Responsive Workbench. *Virtual Reality'94*, 12–15. Retrieved from http://link.springer.com/chapter/10.1007/978-3-662-10795-9_6

Lee, J. C. (2008). Hacking the nintendo wii remote. *Pervasive Computing, IEEE*, *7*(3), 39–45.

Leithinger, D., Follmer, S., Olwal, A., & Ishii, H. (2014). Physical telepresence: shape capture and display for embodied, computer-mediated remote collaboration. *UIST 2014*. Retrieved from http://www.olwal.com/projects/research/physical_telepresence/leithinger_physical_telepresence_uist_2014.pdf

Li, Q. (2012). *Liquid crystals beyond displays: chemistry, physics, and applications*. John Wiley & Sons.

Marquardt, N., Jota, R., Greenberg, S., & Jorge, J. (2011). The continuous interaction space: interaction techniques unifying touch and gesture on and above a digital surface. *... Interaction–INTERACT 2011*, 461–476. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-23765-2_32

Marquardt, N., Kiemer, J., & Greenberg, S. (2010). What caused that touch?: expressive interaction with a surface through fiduciary-tagged gloves. *... Tabletops and Surfaces*, 139–142. Retrieved from http://dl.acm.org/citation.cfm?id=1936680

Modugno, F., Corbett, A. T., & Myers, B. A. (1995). Evaluating program representation in a demonstrational visual shell. In *Conference Companion on Human Factors in Computing Systems* (pp. 234–235).

Moeller, J., & Kerne, A. (2012). ZeroTouch: an optical multi-touch and free-air interaction architecture. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2165–2174).

MS.Corporation. (2011). *Microsoft surface 2.0: Design and Interaction Guide* (p. 70). Microsoft corporation. Retrieved from http://www.microsoft.com/en-us/download/confirmation.aspx?id=26713

Newcombe, R. A., Davison, A. J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., ... Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on* (pp. 127–136).

Nielsen, J. (1993). Noncommand user interfaces. *Communications of the ACM*. Retrieved from http://dl.acm.org/citation.cfm?id=153582

Nishino, H. (2010). A 6DoF fiducial tracking method based on topological region adjacency and angle information for tangible interaction. *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction - TEI '10*, 253. doi:10.1145/1709886.1709937

Oberkampf, D., DeMenthon, D. F., & Davis, L. S. (1996). Iterative Pose Estimation Using Coplanar Feature Points. *Computer Vision and Image Understanding*, *63*(3), 495–511. doi:10.1006/cviu.1996.0037

Owen, C. B., Xiao, F., & Middlin, P. (2002). What is the best fiducial? *The First IEEE International Workshop Agumented Reality Toolkit,* 8. doi:10.1109/ART.2002.1107021

Patten, J., Ishii, H., Hines, J., & Pangaro, G. (2001). Sensetable : A Wireless Object Tracking Platform for Tangible User Interfaces. *Proceedings of the SIGCHI ...*, 253–260. Retrieved from http://dl.acm.org/citation.cfm?id=365112

Patten, J., Recht, B., & Ishii, H. (2002). Audiopad: a tag-based interface for musical performance. In *Proceedings of the 2002 conference on New interfaces for musical expression* (pp. 1–6).

Pyryeskin, D., Hancock, M., & Hoey, J. (2012). Comparing elicited gestures to designer-created gestures for selection above a multitouch surface. *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces - ITS '12*, 1. doi:10.1145/2396636.2396638

Quam, D. L. (1990). Gesture recognition with a dataglove. In *Aerospace and Electronics Conference, 1990. NAECON 1990., Proceedings of the IEEE 1990 National* (pp. 755–760).

Rekimoto, J. (2002). SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 113–120).

Schlömer, T., Poppinga, B., Henze, N., & Boll, S. (2008). Gesture recognition with a Wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction* (pp. 11–14).

Schmalstieg, D., & Wagner, D. (2009). Mobile phones as a platform for augmented reality. *Connections*, 5–6. Retrieved from http://www.icg.tugraz.at/publications/SCHMALSTIEG_SEARIS08.pdf

Schöning, J., & Brandl, P. (2008). Multi-Touch Surfaces : A Technical Guide. *... Interactive Surfaces*. Retrieved from http://mediatum.ub.tum.de/doc/1094399/1094399.pdf

Shneiderman, B. (1993). 1.1 direct manipulation: a step beyond programming languages. *Sparks of Innovation in Human-Computer Interaction*, *17*.

Subramanian, S., Aliakseyeu, D., & Lucero, A. (2006). Multi-layer interaction for digital tables. *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology - UIST '06*, 269. doi:10.1145/1166253.1166295

Sutcliffe, S., Ivkovic, Z., & Flatla, D. (2013). Improving Digital Object Handoff Using the Space Above the Table. *chi'13*. Retrieved from http://hciweb.usask.ca/uploads/289-2013_CHI_Handoff.pdf

Sutherland, I. E. (1968). A head-mounted three dimensional display. *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I on - AFIPS '68 (Fall, Part I)*, 757. doi:10.1145/1476589.1476686

Takeoka, Y. (2010). Z-touch : An Infrastructure for 3D gesture interaction in the proximity of tabletop surfaces. *ITS 2010*, 3–6.

Ullmer, B., & Ishii, H. (1997). The metaDESK: models and prototypes for tangible user interfaces. *Proceedings of the 10th Annual ACM Symposium on ...*. Retrieved from http://dl.acm.org/citation.cfm?id=263551

Ullmer, B., Ishii, H., & Glas, D. (1998). mediaBlocks: physical containers, transports, and controls for online media. *Proceedings of the 25th Annual Conference on ….* Retrieved from http://dl.acm.org/citation.cfm?id=280940

Underkoffler, J., & Ishii, H. (1999). Urp: a luminous-tangible workbench for urban planning and design. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 386–393).

Wagner, D., & Schmalstieg, D. (2007). *Artoolkitplus for pose tracking on mobile devices*.

Weiss, M., Hollan, J. D., & Borchers, J. (2010). Tangible Controls. *Tabletops – Horizontal Interactive Displays*, 149–170. doi:10.1007/978-1-84996-113-4

Wilson, A. D., & Benko, H. (2010). Combining multiple depth cameras and projectors for interactions on, above and between surfaces. *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology - UIST '10*, 273. doi:10.1145/1866029.1866073

Wu, M., & Balakrishnan, R. (2003). Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology - UIST '03*, 193–202. doi:10.1145/964696.964718

Yee, W. (2009). Potential limitations of multi-touch gesture vocabulary: Differentiation, adoption, fatigue. In *Human-Computer Interaction. Novel Interaction Methods and Techniques* (pp. 291–300). Springer.

Yu, N.-H., Chan, L.-W., Lau, S. Y., Tsai, S.-S., Hsiao, I.-C., Tsai, D.-J., … others. (2011). TUIC: enabling tangible interaction on capacitive multi-touch displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2995–3004).

Zimmerman, T. G., Lanier, J., Blanchard, C., Bryson, S., & Harvill, Y. (1987). A hand gesture interface device. In *ACM SIGCHI Bulletin* (Vol. 18, pp. 189–192).

ANNEX 1 LIST OF PUBLICATIONS

- Gallardo, D., Julià C. F., & Jordà S. (2008). **TurTan: a Tangible Programming Language for Creative Exploration.** Third annual IEEE international workshop on horizontal human-computer systems (TABLETOP).

  http://mtg.upf.edu/node/1081

- Gallardo, D., Julià C. F., & Jordà S. (2009). **Turtan: Un lenguaje de programación tangible.** (AIPO, Ed.).Interacción 2009.

  http://interaccion2009.aipo.es/barcelona/

- Jordà, S., Julià C. F., & Gallardo D. (2010). **Interactive surfaces and tangibles.** XRDS: Crossroads, The ACM Magazine for Students. 16(4), 21-28.

  http://dl.acm.org/citation.cfm?id=1764848.1764855&coll=portal&dl=ACM

- Gallardo, D., & Jordà S. (2010). **Tangible Jukebox: Back to palpable music.** Tangible Embedded and Embodied Interaction. (TEI10).

  http://dl.acm.org/citation.cfm?id=1709922&dl=ACM&coll=DL&CFID=456323648&CFTOKEN=15397908

- Jordà, S., Hunter S., Pla P., Gallardo D., Leithinger D., Kaufman H., et al. (2010). **Development strategies for tangible interaction on horizontal surfaces.** Tangible Embedded and Embodied Interaction. (TEI10)

  http://dl.acm.org/citation.cfm?id=1709977&dl=ACM&coll=DL&CFID=456323648&CFTOKEN=15397908

- Julià, C. F., Gallardo D., & Jordà S. (2011). **MTCF: A framework for designing and coding musical tabletop applications directly in Pure Data.** New Interfaces for Musical Expression (NIME11). 457-460.

  http://mtg.upf.edu/node/2703

- Gallardo, D., Julià C. F., & Jordà S. (2013). **Using MTCF for live prototyping on tablet and tangible tabletop devices.** Tangible Embedded and Embodied Interaction. (TEI13).

  http://dl.acm.org/citation.cfm?id=2460724

- Gallardo, D., & Jordà S. (2013). **SLFiducials: 6DoF markers for tabletop interaction.** Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces (ITS13).

  http://dl.acm.org/citation.cfm?id=2514914

ANNEX 2 SLFIDUCIALS

In this annex is shown a subset of the thirty markers designed for the SecondLight, they are designed as five centimetres markers. This subset includes twelve markers plus its twelve inverted patterns that results in a twenty-two markers subset.

ID's from left to right and top to bottom:

0122111

012211

0121211

01212111

012111

01211

01221

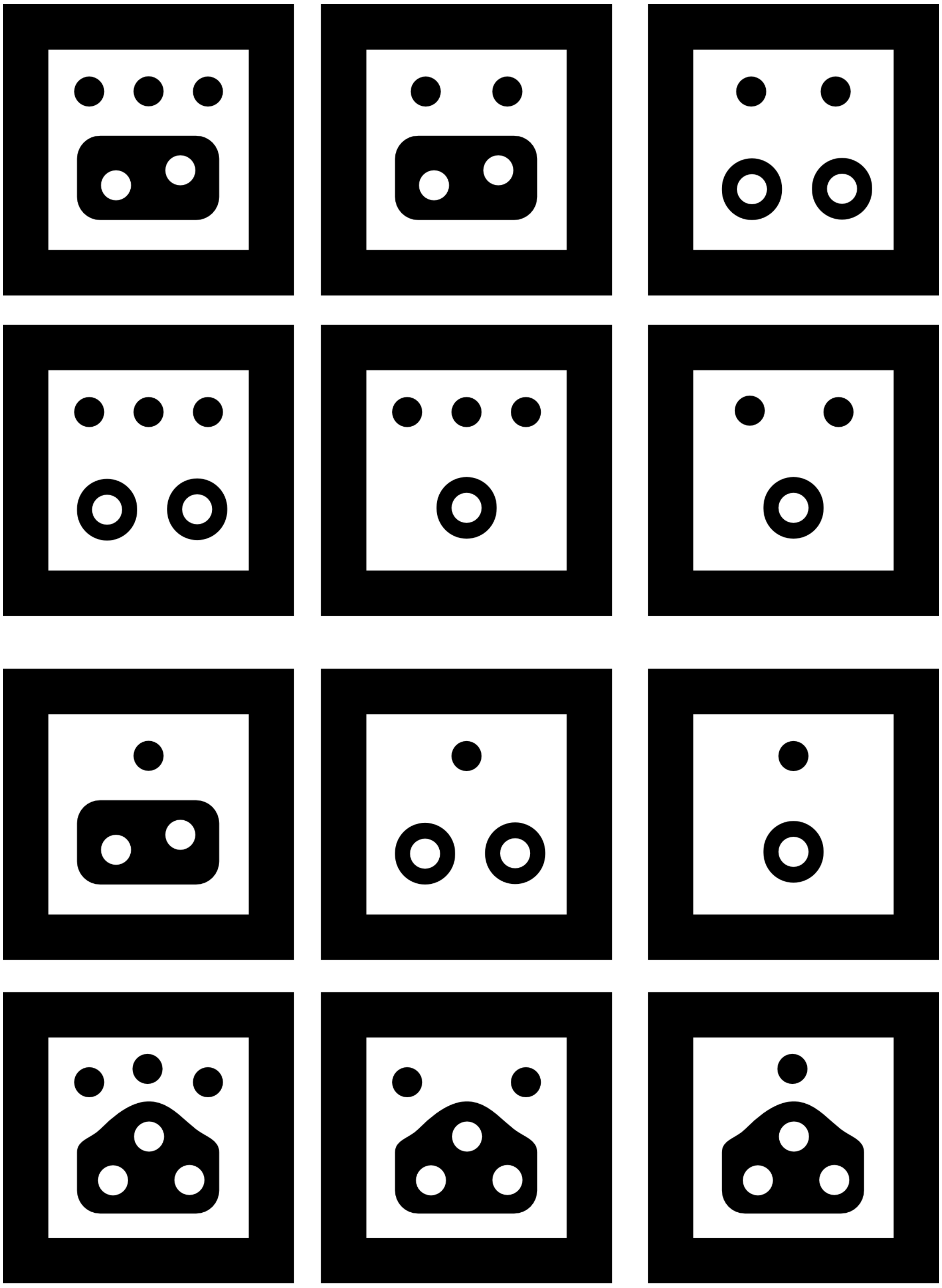
012121

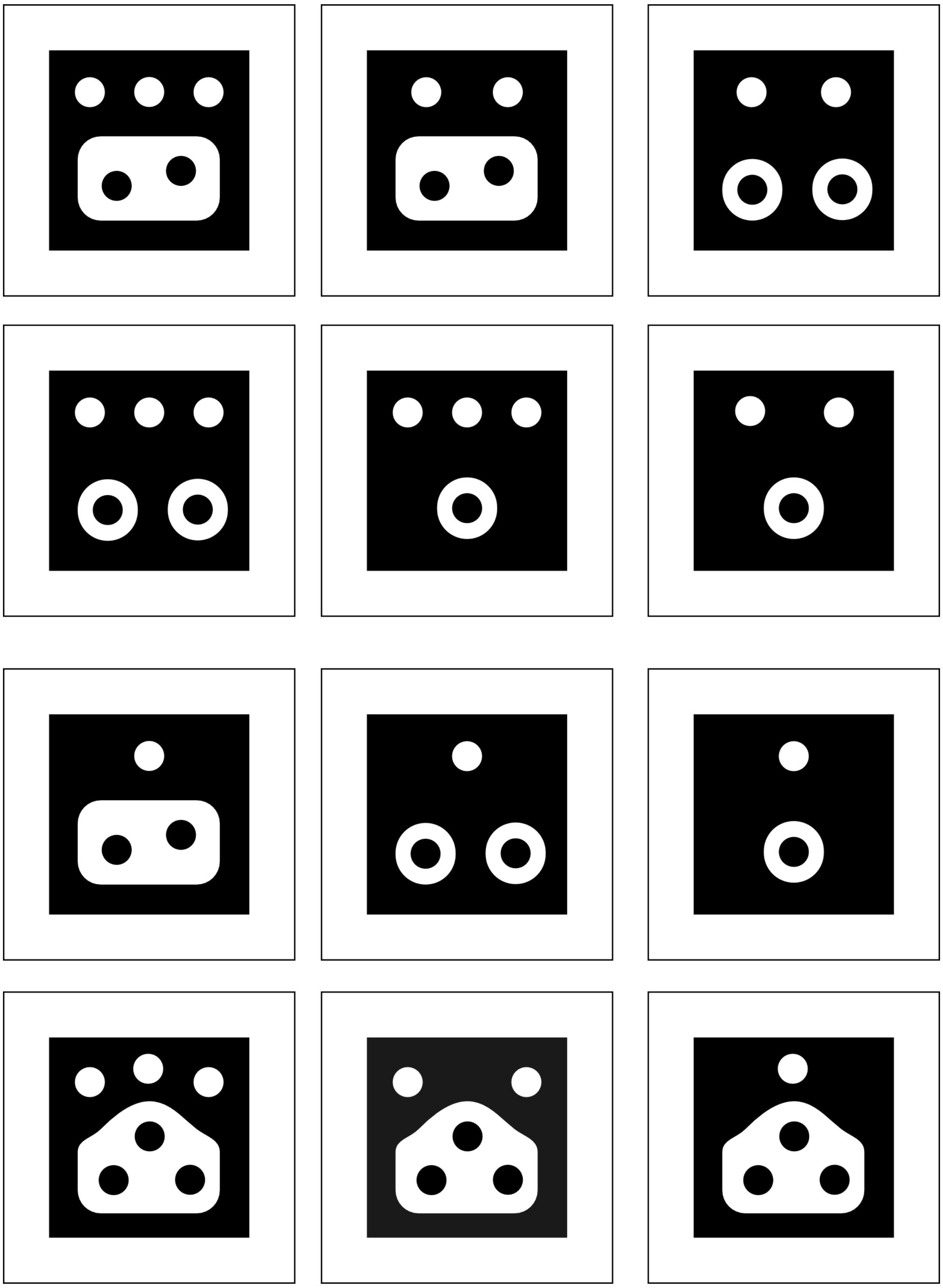
0121

01222111

0122211

012221





ANNEX 3 SLFRAMEWORK GRAPHICS CODE EXAMPLE

This code generates the polygons shown at Figure 59

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using SLFramework.Graphics;
using SLFramework;
using SLFramework.Graphics.Figures;
using Microsoft.Xna.Framework;
using SLFramework.Gestures;
using SLFramework.Gestures.Data;

namespace TestGraphics
{
    class Class1 : TObject
    {
        //objects to be drawn
        TriangulatedShape tshape;
        RegularPolygon rpol;
        RegularPolygon sqr;
        SLFramework.Graphics.Figures.Rectangle rec;
        Circle cir;
        public Class1()
        {
            //subscribe to fingeradd gesture
            Tuio2_Input_Touch.Instance.TouchEvent += new
TouchEventHandler(ProcessTuio2Touch);
            //shapes Initialization
            Vector2[] shape = new Vector2[10];
            shape[0] = new Vector2(-0.1f, -0.1f);
            shape[1] = new Vector2(-0.1f, 0.1f);
            shape[2] = new Vector2(0.1f, 0.1f);
            shape[3] = new Vector2(0.1f, 0.05f);
            shape[4] = new Vector2(0.05f, 0.05f);
            shape[5] = new Vector2(0.0f, 0.08f);
            shape[6] = new Vector2(0.0f, -0.08f);
            shape[7] = new Vector2(0.05f, -0.05f);
            shape[8] = new Vector2(0.1f, -0.05f);
            shape[9] = new Vector2(0.1f, -0.1f);

            tshape = new TriangulatedShape(shape);
            tshape.SetTexture("peng",true);

            Vector2[] sshape = new Vector2[4];
            sshape[0] = new Vector2(0f, 0f);
            sshape[1] = new Vector2(0f,0.01f);
            sshape[2] = new Vector2(0.01f, 0.01f);
            sshape[3] = new Vector2(0.01f,0f);

            rpol = new RegularPolygon(12, 0.1f);
            sqr = new RegularPolygon(4, 0.05f,1,0.03f,0);
            rpol.SubstractPolygon(sqr);
            rpol.SetTexture("peng", true, slg.TEXTURE_INVERT_XY);
            rec = new SLFramework.Graphics.Figures.Rectangle(0.1f,
0.2f,0.01f);

            cir = new Circle(0.1f);
        }

        protected override bool CheckCollision(float x, float y)
        {
            //Console.WriteLine("check");
            return tshape.Collide(x,y);
        }

        protected override void Draw()
        {
            slg.SetColor(255, 255, 255);
            slg.Translate(0.2f, 0.5f);
            //r.Draw();
            slg.Translate(0.2f, 0);
            tshape.Draw();
            slg.SetColor(255, 0, 0);
            tshape.DrawStroke();
            slg.Translate(0.2f, 0);
            rpol.Draw();
            slg.Translate(0.2f, 0);
            rec.Draw();
            slg.Translate(0.2f, 0);
            cir.Draw();
        }

        protected override void DrawSecond()
        {
            base.DrawSecond();
        }

        public override Microsoft.Xna.Framework.Vector2 GetPosition()
        {
            return tshape.GetCentre();
        }

        protected override void Update()
        {
            base.Update();
        }

        protected void ProcessTuio2Touch(int state, InputFinger
finger)
        {
            Console.WriteLine(" ");
            switch (state)
            {
                case Tuio2_Input_Touch.NEW_TOUCH:
                    Console.Write("FingerAdd");
                    break;
                case Tuio2_Input_Touch.UPDATE_TOUCH:
                    Console.Write("FingerUPDATE");
                    break;
                case Tuio2_Input_Touch.REMOVE_TOUCH:
                    Console.Write("FingerREMOVED");
                    break;
            }
            if (this.Collide(finger.X, finger.Y))
                Console.WriteLine(" COLLIDE");
            else Console.WriteLine(" ");
        }
    }
}
```