UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

TANTO - Tangible and touch interaction combined on a surface and above

Rafael Lourenço Lameiras Nunes

DISSERTAÇÃO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Engenharia de Software

2014

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

TANTO - Tangible and touch interaction combined on a surface and above

Rafael Lourenço Lameiras Nunes

DISSERTAÇÃO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Engenharia de Software

Dissertação orientada pelo Prof. Doutor Carlos Alberto Pacheco dos Anjos Duarte

2014

Acknowledgments

I would like to thank, first and foremost, my mentor Professor Carlos Duarte for guiding me through this master thesis. I am forever thankful for his constant availability and interest in my work and for always providing new insights and ideas where I would, otherwise, be too blind to see. I hope we continue to do great research in the years to come.

I would also like to thank my family for all the support they have given me throughout my academic life, always taking the pressure away, leaving only room for success.

Finally I would like to thank, all my friends and colleagues who were with me through every step, helping me get through many sleepless nights and bringing me up when I was feeling down.

To my parents

Resumo

As interações multi-toque estão tipicamente limitadas a uma superfície mesmo quando combinadas com tangíveis. Os cenários tradicionais, onde os utilizadores interagem com objectos físicos numa mesa e por cima dela, não foram ainda replicados com sucesso utilizando tecnologias existentes como, por exemplo, mesas multi-toque. Estas não suportam as interações naturais do utilizador ao combinar a superfície da mesa com a área acima dela num espaço contínuo de interação, limitando assim a sua aplicabilidade. Este trabalho aponta para a construção e exploração de uma mesa que permita aos utilizadores beneficiar de um espaço contínuo de interação na mesa e acima dela com interações multi-toque e tangíveis.

Para atingir este objectivo, melhorámos uma mesa multi-toque existente, de forma a suportar interações com tangíveis na superfície e por cima. Para alcançar este resultado é necessário recorrer a várias tecnologias. Para enquadrar esse desenvolvimento, apresentamos uma revisão do estado da arte das tecnologias de interação atuais que incluem interações com toque, tangíveis e gestos. Estas tecnologias são implementadas na nossa mesa para oferecer estas formas diferentes de interação.

Suportar todas estas tecnologias de interação traz o problema acrescido de combinar diferentes fontes de informação. Como tal, sentimos a necessidade de desenvolver uma ferramenta que nos permitisse não só juntar todas as componentes, mas também distribuir a sua informação para aplicações clientes de uma forma fácil de compreender e utilizar. Apresentamos a TACTIC, uma API que é capaz de combinar superfícies de toque, tangíveis e interações por cima da mesa de uma forma que permite aos programadores utilizar as suas funcionalidades e distribuir interfaces através de múltiplos aparelhos, se necessário. A TACTIC é desenvolvida em JavaScript, sendo responsável por conectar aplicações executadas em navegadores Web a várias fontes de dados, enviando-lhes informação de toque, tangíveis e gestos de uma forma fácil e rápida.

A TACTIC foi desenvolvida para funcionar com mesas multi-toque existentes, permitindo-lhes tirar proveito do espaço por cima da mesa através de detecção de gestos. Graças ao facto de correr nativamente em navegadores Web, a TACTIC tem o benefício acrescido de ser facilmente disponibilizada numa mesa de toque ou smartphone, suportando abstrações de eventos de toque, permitindo assim que o mesmo código seja reutilizado quer em mesas físicas ou dispositivos móveis. Adicionalmente, permite a disponibilização fácil de

objetos digitais com comportamentos interativos e torna informação de gestos disponível de forma a que um evento de toque ou tangível traga consigo a informação da mão e dedos utilizados por associação.

A TACTIC tem uma arquitetura altamente modular graças ao RabbitMQ, um middleware de mensagens que liga as diferentes componentes e linguagens permitindo comunicação simples e direta entre elas. Desta forma, é possível adicionar novas componentes com facilidade sem se fazer alterações a configurações anteriores. Esta arquitetura inclui um módulo Node.js para comunicação entre aplicações Web em cenários com vários dispositivos, permitindo assim o fácil desenvolvimento de interfaces distribuídas.

Para investigar a facilidade de aprendizagem e uso da nossa API foi conduzido um estudo com programadores. Os participantes deste estudo foram incumbidos com a tarefa de desenvolver aplicações que requerem conhecimentos de diferentes aspectos da TACTIC, assim como também algumas bases de JavaScript e CSS. O objectivo foi compreender o nível de facilidade e rapidez com que os programadores são capazes de desenvolver aplicações complexas utilizando a TACTIC. Para atingir este objectivo, foi pedido aos participantes o desenvolvimento de uma aplicação de pintura, cuja complexidade iria aumentando gradualmente tarefa a tarefa, juntamente com as funcionalidades da API a utilizar. Ao chegar ao fim das tarefas, os participantes conseguiram construir aplicações que usavam toque, tangíveis e interações acima da mesa em cenários com mais que um dispositivo em pouco tempo. Este estudo comprovou a facilidade de compreensão e uso da TACTIC, graças à sua promoção de reutilização de código e abstrações que permitiram uma rápida implementação das suas várias funcionalidades em aplicações Web.

Apresentamos, adicionalmente, um conjunto de aplicações que demonstram as funcionalidades chave da TACTIC. Estas aplicações distribuem-se em múltiplas formas de interação e interfaces. Este trabalho descreve como estas aplicações utilizam os vários eventos e propriedades da nossa API, variando entre interações de toque e tangíveis na mesa a interações acima da mesa e cenários com vários dispositivos.

Para este trabalho, comprometemo-nos a resolver problemas existentes com mesas semelhantes à nossa. Em cenários de colaboração, por exemplo, as interações à volta da mesa podem causar interferência entre utilizadores. Queremos explorar novas soluções para estes problemas e integrá-las na nossa mesa, explorando diferentes cenários, tanto individuais como colaborativos, para atingir uma interação natural em toda a área do espaço de interação. Desta forma, decidimos expandir as capacidades da nossa mesa para permitir interações em cenários de colaboração. Como tal, apresentamos o processo necessário para tornar esta funcionalidade uma realidade seguido de uma aplicação que demonstra o seu uso.

Sentimos que existe uma falta de estudos sobre a forma como o espaço contínuo de interação causa impacto nas interações de utilizadores. Adicionalmente, não existem comparações de desempenho em gestos semelhantes na mesa e por cima dela. Como

tal, tiramos vantagem da nossa API para contribuir com um estudo sobre o desempenho dos utilizadores quando executam ações na mesa e por cima dela, apontando para resultados que serão úteis para informar o desenho futuro de aplicações que explorem este espaço contínuo de interação. De acordo com o que conseguimos apurar, este é o primeiro estudo que compara ações tanto na mesa como por cima dela. Para tal, escolhemos ações de "Zoom" e Rotação, dado que os gestos de "Pinch" e Rotação são bastante comuns em interações com smartphones e tablets. Na superfície estes gestos são realizados colocando dois dedos na mesa e fazendo um gesto de "pinch" ou rotação, como é normal. Dado que por cima da mesa não há uma superfície sobre a qual se possa repousar os dedos, os gestos utilizados foram ligeiramente alterados. Para se fazer "zoom", os utilizadores devem fechar os dedos num gesto de pinch para selecionar e de seguida controlar o nível de "zoom" ao mover a mão mais perto (zoom in) ou mais longe (zoom out) da mesa. Para fazer uma rotação os utilizadores colocam a sua mão aberta por cima do elemento e rodam-na num plano paralelo ao da superfície da mesa. Este estudo confirmou que o desempenho na superfície é melhor que por cima dela, enquanto que outros resultados permitiram investigar o impacto que a área onde o gesto é feito tem no seu resultado desejado; a relação entre as mecânicas das tarefas e a ergonomia humana; e os benefícios que podem vir de permitir a superfícies de toque o reconhecimento de gestos por cima delas.

Contribuímos, também, para um estudo com utilizadores cegos, que nos forneceu a oportunidade de testar as aplicações da TACTIC e a nossa mesa no campo da acessibilidade. Este estudo captura dados de desempenho de utilizadores ao explorar elementos numa superfície com uma ou duas mãos, revelando que a exploração da superfície com duas mãos consegue melhorar as suas habilidades para este efeito. A TACTIC foi responsável por detectar as mãos e dedos utilizados a todo o momento. Tirámos proveito da modularidade da sua arquitetura para incorporar com facilidade uma componente áudio e de auditoria existentes com a aplicação desenvolvida. Esta forma de interação com duas mãos, demonstrou ser benéfica para algumas tarefas, particularmente a relação entre alvos e promover uma melhor estruturação na tarefa de exploração.

Palavras-chave: Tangível, Multi-toque, Dispositivos Móveis, Gestos

Abstract

Multitouch interaction is usually limited to one surface, even when combined with tangibles. Traditional scenarios where people interact with physical objects on and above the table or other surfaces have failed to be fully translated into existing technologies, such as multitouch setups, which don't support natural user interactions by combining the surface and the area above it into one continuous interaction space. We built on top of an existing multitouch setup to support tangible interactions on and above the surface.

Various technologies are necessary to achieve this result, which brings the added problem of combining the different sources of information. We present TACTIC, an API that is capable of combining touch surfaces, tangibles, and the interaction space above the surface, in a way that allows developers to easily combine all these features, and distribute interfaces across multiple devices if required. Additionally, we present the results of a developer study showing how TACTIC is easy to learn and use.

We take advantage of TACTIC's capabilities to conduct a study on user performance when performing actions on and above the table, aiming for results that will be useful towards informing the design of applications that explore a continuous interaction space.

We showcase TACTIC's capabilites through a set of applications that draw from its many features, demonstrating its flexibility and ease of use.

Keywords: Tangible, Multitouch, Mobile devices, Gestures

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis work studies interactions on and above the surface. We aimed to achieve a setup that is capable of supporting these interactions in a continuous interaction space, but also to better understand how they can impact user experience. This is coupled by an API that can serve as a tool to handle communication between interaction technologies and help developers in the creation of applications that draw from these interactions with ease. In this chapter we will detail the motivation behind this work and our main goals going forward as well as the structure for this document.

1.1 Motivation

Multitouch surfaces are emerging in an ever growing list of everyday scenarios. Traditionally, this type of setup allows interaction with elements on the surface projection through touch input. This is a paradigm that has been studied in many researches ([30], [34]) and it became clear that it could benefit from some augmentations that would add to the interaction experience ([17], [36], [20]).

There is a whole area above the table surface that can pave the way for new interactions. This continuous interaction space ([25]) allows the user to, for example, interact with gestures freely throughout its area. Gesture recognition adds natural user interaction with hands above the surface, while touch recognition allows it on the surface.

Prior to this work, we had built a multitouch tabletop that supported interactions both on and above the surface. We experienced how the continuous interaction space adds a new dimension and allowed us to build richer and more diverse applications. Thanks to this, it is no longer necessary to end a use case when the user's hand leaves the table surface, since the interaction can continue above it. Interaction elements can grow in number, becoming more than just projected elements on the surface, being that it is possible to interact with virtual objects above the surface that can exist in a 3D space.

We asked ourselves, "how can we improve this setup and add more to the experience, while still maintaining natural user behaviour?". The answer came from one of the most

common everyday interactions, which is manipulating physical objects.

Handling tangibles comes natural to any user and so we felt that translating that interaction to our setup would add to the experience in a positive way, while maintaining the natural user behaviour. Tangible objects are input elements that can exist in the 3D space while also interacting with the surface. Users can grab objects and move them anywhere bringing information along with them. By taking advantage of the continuous interaction space, we can keep track of the whole process of moving an object, from the moment it is picked up from the surface until it is put down again. We want to explore these possibilities and new ways of interaction. As such, our setup needs to be augmented to keep track of objects touching the surface and hovering above it.

We set out to solve existing problems with similar setups. For example, in a collaboration scenario around a multitouch tabletop, there can be interference between interactions. We wish to explore new solutions for these issues and integrate them into our setup, exploring different scenarios, both individual and collaborative, to achieve seamless and natural interaction in every area of the interaction space.

When considering the hardware and software necessary to support interactive surfaces, mid-air gesture and object recognition above the surface, and tangible user interfaces, there is a clear challenge in making all the components communicate and exchange information from each other. Furthermore, having the information from each platform available in a common programming environment is another limiting factor that prevents a wider exploration of the interaction possibilities made available by these platforms. We aim to create an API that bridges all of these components together while managing all of the existing information, allowing developers to build applications that explore the continuous interaction space with more ease and efficiency.

We feel that there is a lack of studies on how the continuous interaction space can impact user interactions, furthermore, comparisons of the performance of similar gestures on and above tabletops are also missing. As such, we wish to take advantage of our API to contribute with a study on user performance when performing actions on and above the table, aiming for results that will be useful towards informing the design of applications that explore a continuous interaction space.

1.2 Objectives

This work aims to achieve the following goals:

- Build upon our existing setup to allow object manipulation on and above the surface
- Explore various interaction settings on and above the surface with touch, gestures and tangibles

- Development of an API that allows easy and transparent development of applications for our setup integrating each technology and managing communication between different components.
- Test and validate our API for future distribution
- Develop a set of applications that test new ways of interaction with the combination of technologies we proposed in both collaborative and individual scenarios.

1.3 Contributions

While aiming to achieve our previously set objectives, these are our main contributions :

- **Augmented Setup** - our proposed setup that supports touch interactions, tangible interactions and mid-air gesture and object recognition above the surface, allowing new forms of interaction and collaboration around the table.
- **TACTIC API** - our proposed API bridging different technologies to allow developers to easily create applications that combine touch surfaces, tangibles, and the interaction space above the surface as well as cross device scenarios. This API was validated through a developer study and distributed at http://accessible-serv.lasige.di.fc.ul.pt/˜tactic/
- **Paper publication** - paper on *"Combining multitouch surfaces and tangible interaction towards a continuous interaction space"* [28] detailing our proposed setup and API.
- **Gesture performance study** - a study on user performance on zoom and rotation gestures on and above the surface, informing on future design of interactive systems that aim to explore the continuous interaction space.

1.4 Document Structure

This document is structured as follows: Chapter 2 details a set of technologies and techniques used in the computer vision field, followed by a detailed description on how these were implemented towards building our proposed setup to support various types of interactions; Chapter 3 discusses various works that include similar setups with different characteristics and used techniques, as well as existing API's that enable the development of applications for different interactive scenarios and various works regarding tangible interactions and user collaborations. Chapter 4 presents a detailed description of our proposed API, including feature analysis and a brief tutorial, followed by a developer study validating its ease of use. Chapter 5 presents a user study comparing the performance of

zoom and rotation tasks on and above an interactive surface, aiming to contribute to the knowledge about interactive gestures, complementing existing characterization of gestures either on tabletop surfaces, or in mid-air. Chapter 6 showcases our API through a set of applications that draw from its core features, followed by a contribution for a user study in the accessibility field and our take on interactions in a collaborative scenario. Finally chapter 7 presents our conclusions and our thoughts on future efforts for this work.

Chapter 2

Interactive Table Setup and Technology

This chapter presents a description of the built interactive setup, supporting interaction on the table and above it.

2.1 Technologies

This section describes how each of the deployed technologies work. The presented technologies range from enabling multitouch interaction, tangible interaction and tracking and recognition of mid-air gestures

2.1.1 FTIR: Frustrated Total Internal Reflection

Frustrated Total Internal Reflection is a multitouch technology developed by Jeff Han. It uses the concept of Total Internal Reflection, which is a condition present in certain materials when light enters one material from another material with a higher refractive index [1]. As seen in Figure 2.1, Infrared Light is flooding the inside of a piece of acrylic and remaining trapped. When the user comes into contact with the surface the light rays are frustrated and can pass through. The infrared camera below can capture this light which allows touch point detection.

There are a number of key aspects to ensure a good realization of this effect. An array of infrared emitting diodes needs to be mounted around the acrylic. The compliant surface placed on top of the acrylic needs to be coated with a silicon rubber layer to improve adherence which in turn improves the Total Internal Reflection. And finally the camera used to capture the light should have a specific filter to allow it to only capture light that is inside the same spectrum as the IR light being emitted. In Figure 2.2 the white blobs represent infrared light coming down and being captured by the camera, allowing for finger tracking.

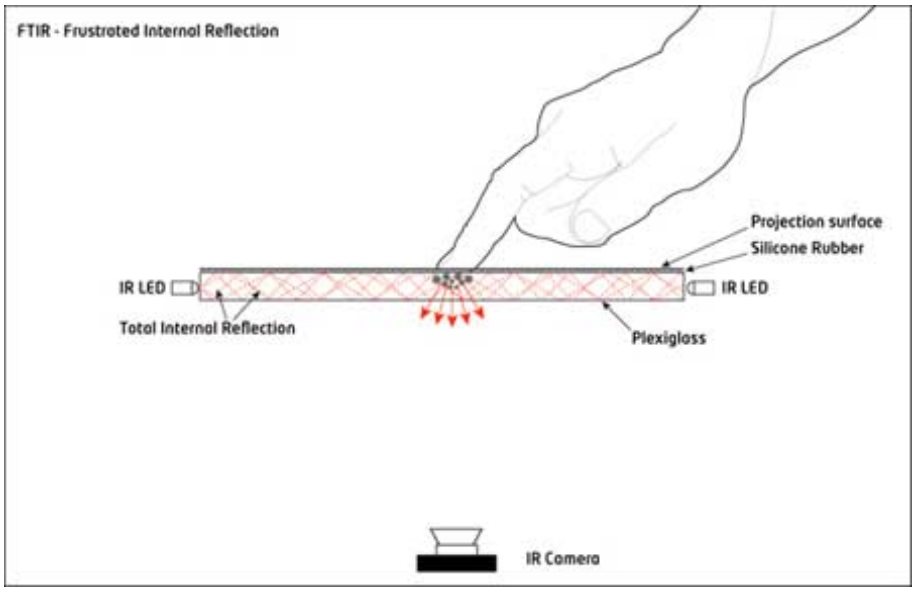

Figure 2.1: Light frustrated inside a material (image taken from [1])

Figure 2.2: Infrared Light that escapes FTIR and is captured by the camera (image taken from [1])

2.1.2 TUIO

TUIO is a protocol specifically designed to meet the requirements of table-top tangible user interfaces [22]. Its flexible design offers methods to select which information will be sent and it doesn't affect existing interfaces, or require re-implementation to maintain compatibility.

The protocol describes two main classes of messages, *Set* and *Alive*. *Set* messages transmit the object's state, such as position, orientation. *Alive* messages indicate the current set of objects on the surface using a list of session IDs. These messages are transmitted using UDP transport to provide low latency communication. Since with UDP transport there is a possibility to loose packets, TUIO uses redundant information to correct possible lost packets.

TUIO has been adopted by several projects, all related to tangible and multitouch interaction and numerous TUIO clients for various platforms and languages continue to

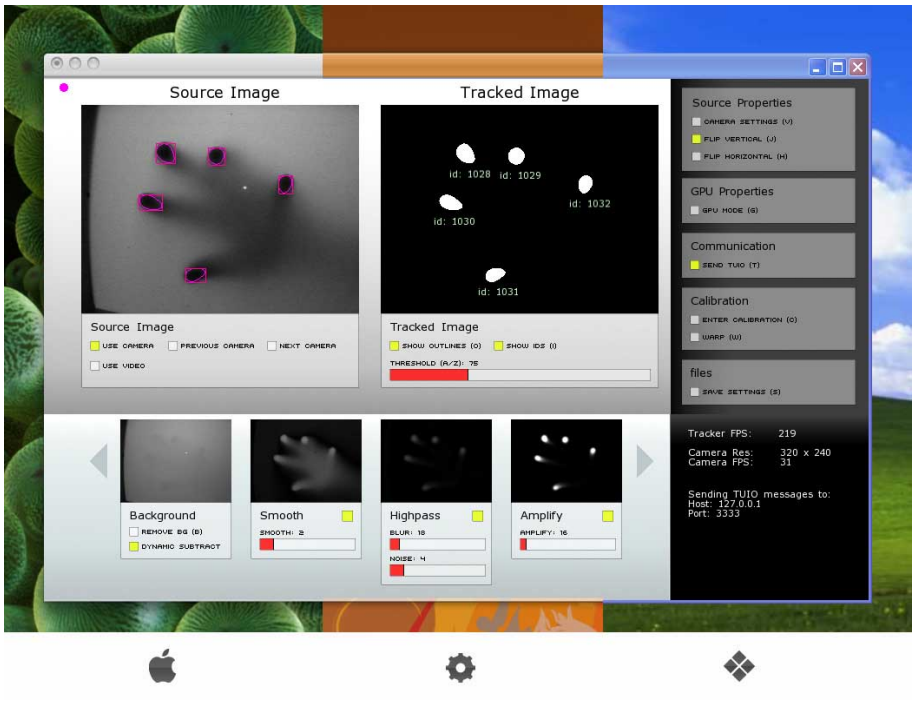

Figure 2.3: Community Core Vision software (image taken from [7])

surface to provide easy development of tabletop applications.

2.1.3 Community Core Vision

Community Core Vision is an open source/cross-platform software for computer vision and machine sensing. It can interface with various web cameras and other video devices to gather video input stream which is then output as tracking data (e.g. coordinates and blob size) and events (e.g. finger down, moved and released), as seen in Figure 2.3. This information can then be sent to client applications through TUIO protocol (section 2.1.2) as well as others.

CCV[1] is developed and maintaned by the NUI Group Community and supports many multitouch lighting techniques, such as FTIR (section 2.1.1), DI, DSI, and LLP.

2.1.4 *reacTIVision*

reacTIVision is an open source, cross-platform computer vision framework that tracks fiducial markers attached onto physical objects, as well as multitouch finger tracking. Developed by Martin Kaltenbrunner and Ross Bencina at the Music Technology Group at the Universitat Pompey Fabra in Barcelona, Spain, *reacTIVision* is a standalone application[2].

reacTIVision tracks fiducial markers (Figure 2.4) in a real time video stream. Through

[1] http://ccv.nuigroup.com/
[2] http://reactivision.sourceforge.net/

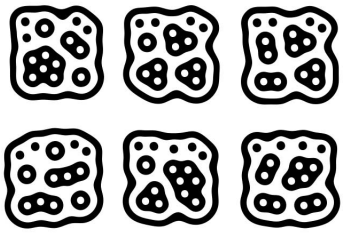

Figure 2.4: Fiducial Markers (image taken from [5])

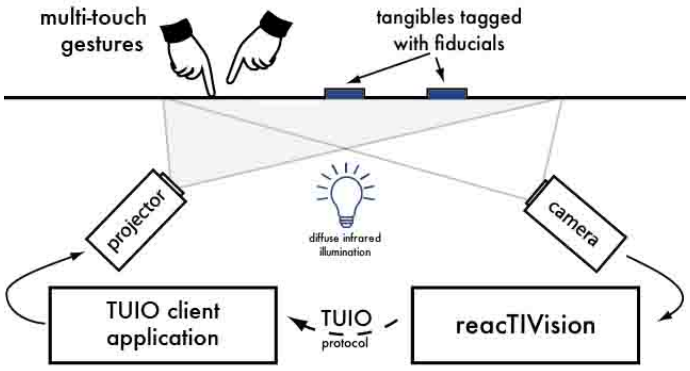

Figure 2.5: *reacTIVision* setup (image taken from [5])

TUIO messages via UDP port 3333, it sends messages to any TUIO client application (Figure 2.5).

This technology also allows for finger tracking by identifying small white blobs as finger tips on the surface, but since *reacTIVision* was initially designed for fiducial tracking it has been optimized for this task only, thus the finger tracking is not ideal and can be better achieved through different technologies, such as CCV (section 2.1.3).

2.1.5 ThreeGear

ThreeGear is a technology developed by 3Gear Systems that enables precise finger and hand tracking. It uses Kinect cameras to reconstruct a finger precise representation of what the hands are doing, which allows it to leverage small gestures, like pinching and wrist movements instead of traditional arm detection. It can provide milimiter-level precision of the user's hand using a camera mounted above the hand [4].

This system is coupled with a corresponding API[3] that allows writing of software applications based on its technology to explore this new level of precision. Although it is designed to fit on top of a traditional desktop setup, as seen in Figure 2.6, it can be scaled

[3] http://www.threegear.com/

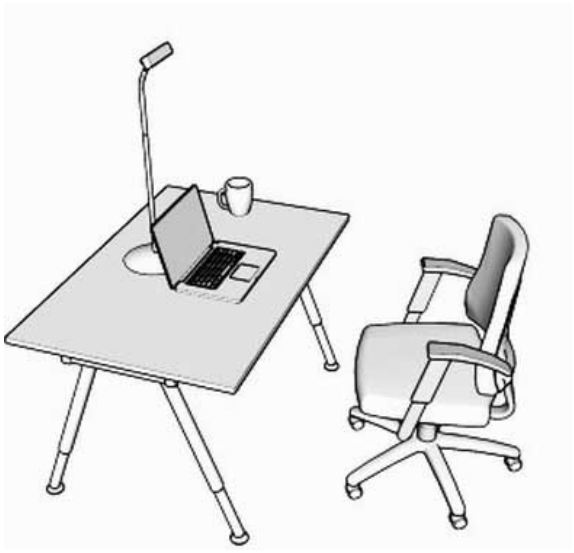

Figure 2.6: 3D camera mounted above the desktop setup (image taken from [4])

to other settings, like table-top, as long as the camera range is adjusted.

2.2 Setup description

Different technologies allow our set-up to detect user interactions throughout a continuous interaction space. The following sections describe how these technologies were deployed and what they "bring to the table".

2.2.1 On the surface

The assembled table measures 111 x 89 cm and has a height of 96 cm. Touch interactions on the surface are handled with *Frustrated Total Internal Reflection* (FTIR, section 2.1.1), which allows the detection of touch input through an array of infra-red light. We chose FTIR over other lighting technologies, since it proved to be the most effective and error free based on previous experiments with diffused lighting. To achieve this effect, a strip of infrared LEDs[4] was placed around an 58 by 76 cm acrylic with polished borders of 5mm thickness. A drafting paper with a silicon coating was placed on top of the acrylic to ensure a compliant touch surface. We inserted a *Playstation Eye* camera with a specific filter inside the setup to capture the LED strip's wave length and ignore any other source of light. The information captured by the camera is then interpreted by Community Core Vision (section 2.1.3) and translated into TUIO protocol (section 2.1.2) messages for client applications.

Object tracking on the surface is achieved through fiducial markers that are placed

[4]http://www.environmentallights.com/led-infrared-lights-and-multi-touch/infrared-led-strips/ir-led-strips.html

on physical objects, as seen in Figure 2.7. An HP web camera was introduced to capture these markers through normal visible light and send the information to *reacTIVision* (section 2.1.4), which translates it into TUIO (section 2.1.2) protocol messages for client applications.

Figure 2.7: Tangibles with fiducial markers

Both cameras are 74 cm below the table surface. A short throw projector with a 1280x720 pixel resolution is placed 74 cm below the table surface to project information on the table's surface.

2.2.2 Above the surface

A Microsoft Kinect camera was placed 89 cm above the surface, as seen in Figure 2.8 to capture hand and finger data. This data is handled through ThreeGear (section 2.1.5), which allows precise finger and hand tracking as well as the detection of small gestures, like pinching and wrist movements (Figure 2.9).

The ThreeGear API limits the hand detection to one pair and to the direction the camera is facing, which means that hand detection is only achieved on the side the camera is in. Additional hand pairs can be detected on other sides of the table by adding more Kinect cameras as will be detailed in Chapter 6.

2.3 Discussion

In this chapter we presented various technologies that are responsible for different types of tracking, as well as a description of how they were deployed towards building an augmented multitouch setup that supports touch and tangible interactions on and above the surface merging both into one continuous interaction space. By combining these different technologies this setup is capable of supporting natural user interactions with various modalities without interruptions when transiting from the surface to the area above it, and back.

Chapter 4 will present an API that is built to support communication between these different technologies and provide tools for easy development of applications in our setup.

Figure 2.8: Our multitouch set-up

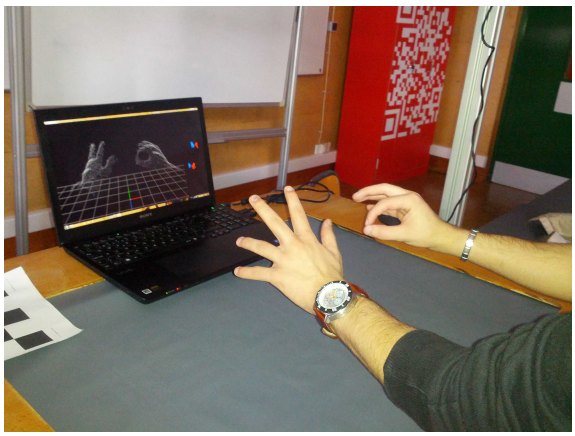

Figure 2.9: ThreeGear hand tracking

Chapter 3

Related Work

In this chapter we will discuss different setups and APIs for interaction with and above tabletop surfaces. Many different technologies have been deployed to allow interaction on and above the table. Each one has its own set of interactions as well as advantages and disadvantages. We will present a state of the art on how two different spaces can coexist and even collaborate to improve the user's ease of interaction.

3.1 The continuous Interaction Space

The rising popularity of digital surfaces has peaked the interest of researchers in the development of a broader set of interaction techniques.

Since most interactions fall into modalities such as direct touch and multitouch (by hand and by tangibles) directly on the surface, or hand gestures above the surface, they are limited in the fact that they ignore the interaction space between them. By merging all of the space into one interaction, a person can use touch, gestures and tangibles anywhere in the space. This, of course, brings out a new set of interactions thanks to the collaboration between modalities.

The continuous interaction space (Figure 3.1), is composed of the touch surface and the space above. Gestures don't necessarily need to be limited to interactions below one's hands. Thanks to the space above, the user's reach can be expanded beyond these physical limits [25], which means that a gesture that a person starts through direct touch can continue in the space above the surface. Normally a user would be able to grab an object through touch and drag the object along the surface, but now this action can be continued by lifting the hand into the 3D space, (as illustrated in Figure 3.2), [25]. This new dimension adds new ways to interact with elements in the table, and also new gestures.

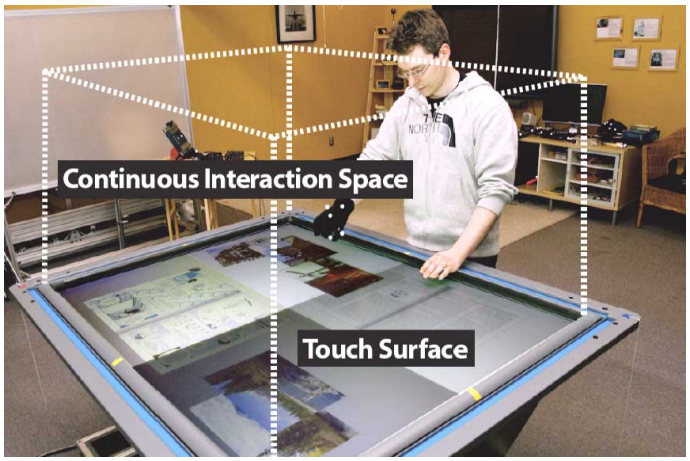

Figure 3.1: The continuous interaction space (image taken from [25])

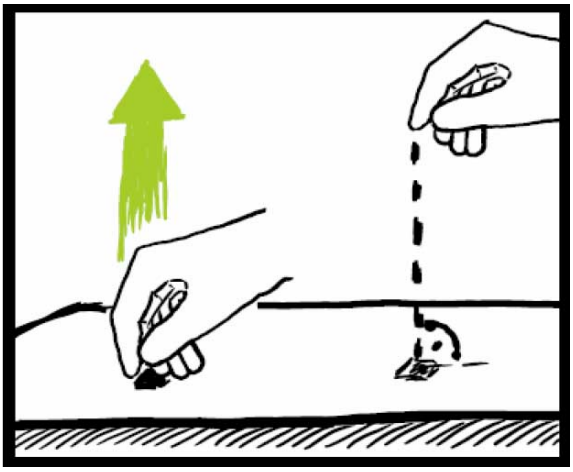

Figure 3.2: Interaction with touch and 3D space (image taken from [25])

3.2 Similar Setups

3.2.1 Medusa

In [8], Medusa, a proximity-aware multitouch tabletop is presented. Medusa uses 138 proximity sensors to detect a user's presence and location, determine body and arm locations, as well as distinguishing between right and left arms and map touch points to specific users and hands. Multiple proximity sensors have been used before in many works, for example [11], but Medusa stands out with its 138-sensor implementation.

The proximity sensors are arranged in three rings, as shown in Figure 3.3. The outward facing ring, is composed of 34 long-range sensors, spaced 3.3 cm apart and mounted at the top of each side of the table. Since this ring's sensors point outwards, a horizontal sensing plane that projects 80 cm from the side panels is created around the surface. Forty six long-range sensors are spaced 3.3 cm apart and pointing upwards, making up the outer ring of sensors, creating a vertical sensing plane wrapped around the perimeter of the

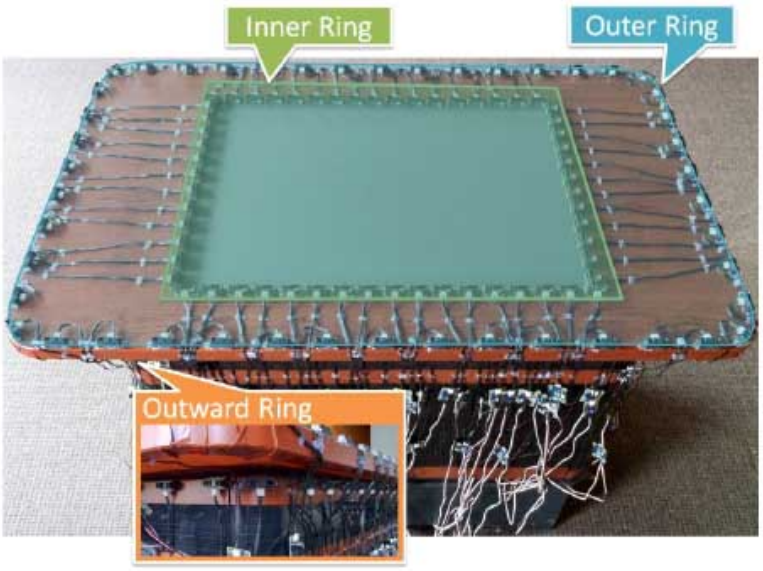

Figure 3.3: Medusa's sensors arranged in three rings [8]

tabletop. Finally 58 short-range sensors are spaced 0.8 cm apart and located around the touch area. These sensors point upwards to form an inner vertical sensing plane.

Medusa can provide the user's location, to explore this in a real setting. Different sides of the tabletop are assigned to different fidelities of a current prototype. So if a user is building a prototype application, the table can show a sketch when the user is standing on one side of the table, and change the sketch to a higher fidelity when the user walks over to an adjacent side of the table, as shown in Figure 3.4.

Medusa's technology allows for user logins, so in a multi-user scenario, if a user walks up to the tabletop and does not login, all of his interactions will automatically be blocked, since touch points are mapped to users. A "Do Not Disturb" mode was created to take advantage of this, providing users who are interacting with the system with a way of discouraging others from approaching, as seen in Figure 3.5. Although Medusa adds new interesting multi-use scenarios, it lacks any form of interaction above the table surface.

3.2.2 HandsDown

HandsDown is a technique that enables users to access personal data on a shared surface, associating objects with their identity and customizing appearance, content, or functionality of the user interface.

In [32], HandsDown is paired with a custom-build tabletop system, similar to Microsoft's Surface. Two image filter chains are applied on the hand image to extract finger touches and hand contours out of the same source. This makes hands appear as clear shadows in front of the surface, as shown in Figure 3.6(a). Then an infrared filter is used to remove visible light, and contours are extracted (Figure 3.6(b)).

As points with high curvature correspond to changes in contour direction, a filter is

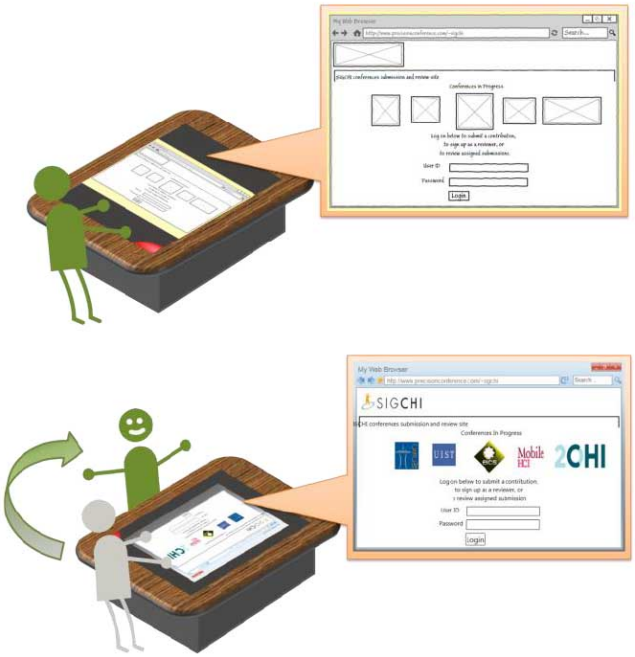

Figure 3.4: Low Fidelity Prototype being shown by default. Once a user walks to an adjacent side of the table, a high fidelity prototype is shown [8]

applied to select them and respective center points are selected as hand extremity candidates (Figure 3.6(c)). Lines connecting finger tips and center points between two adjacent finger valleys are extracted as the main axis and divided into six equally sized partitions (Figure 3.6(d)). A set of features is then selected to maintain a profile for that user's hand. A resulting example is shown in Figure 3.7.

HandsDown allows attaching identities to tangible objects. By placing an object and a registered hand on the surface next to each other the surface is able to establish an identity association between the two. In Figure 3.8 a user is attaching his identity to a mobile phone on an interactive surface. This technique enhances previous attempts at device and touch pairing, like BlueTable [37] and PhoneTouch [31]. It can even be further extended to access control.

Access control is explored in [32] as a tool to improve on some problems with collaboration around the table. Sometimes users can interfere with each other when using the same space, which causes discomfort. Access control allows users to protect their interactions in a variety of ways. A user can protect a document so that only his hand is allowed to access the file, which is a comfortable alternative to passwords, since in a collaborative environment passwords are subject to shoulder-surfing, which happens when a third person is able to peek at what a user is typing. There is also the possibility to lock workspaces. Much like the "log off user" function on personal computers, a user can minimize or lock his personal workspace while other users continue working with their workspaces.

Figure 3.5: Medusa on ”Do not disturb” mode. All logged out users are greeted with a ”prohibited” glowing red orb [8]

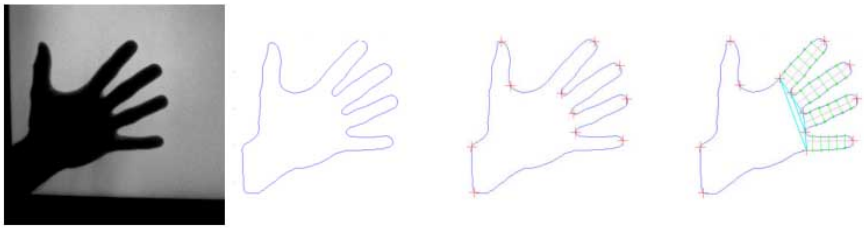

Figure 3.6: HandsDown extraction steps: (a) raw camera image, (b) extracted contours, (c) high curvature points, (d)extracted hand features [32]

3.2.3 LightSpace

LightSpace (Figure 3.9) is a small room installation designed to explore a variety of interactions and computational strategies related to interactive displays and the space that they inhabit. Cameras and projectors are calibrated to 3D coordinates allowing for projection of graphics correctly on any surface visible by both camera and projector.

The motivation behind LightSpace is to study how depth cameras enable new interactive experiences. Its goal is to enable interactivity and visualizations throughout everyday environments without the need to augment users and other objects in the room with sensors or markers [36].

This technology allows any normal table or surface to become an interactive display that allows users to use hand gestures and touch to manipulate projected content. This smart room configuration allows new combinations of interaction. Wilson and Benko [36] describe them as follows:

- *Through-Body Transitions Between Surfaces*

  It is possible to move objects between interactive surfaces through-body by touching the object and then touching the desired location (Figure 3.10). The system can

Figure 3.7: HandsDown shows users feedback when a hands is placed on the surface [32]

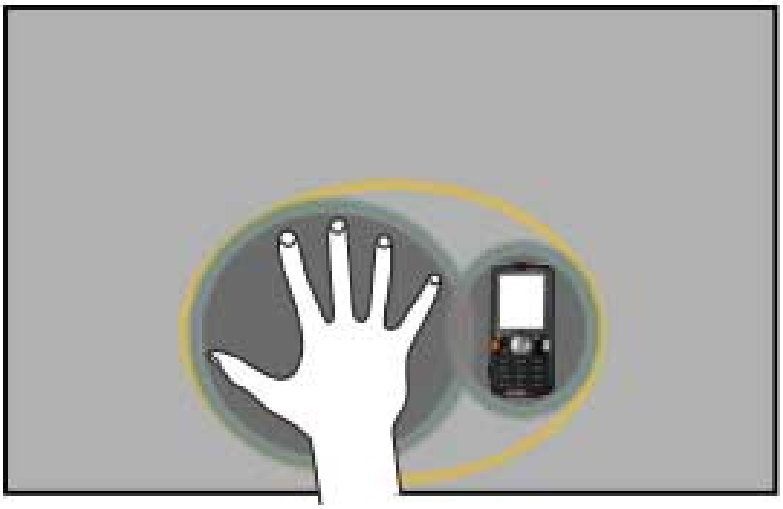

Figure 3.8: User's hand attached to tangible object through identification [32]

infer that both contacts were done by the same person, thus establishing a connection between the two surfaces.

- *Picking up Objects*

  A user can drag an object off an interactive surface and pick it up with their hand (Figure 3.11). Although the system does not track the hand (or any other part of the body), it gives a physics-like behaviour to each object. While the user is holding the object it can either touch an interactive surface, resulting in a through-body transition of the object to that surface or pass it around to others in the environment, and carry it between interactive surfaces.

- *Spatial Menus*

  The extra dimension in the user's position can be used to enable spatial interfaces (Figure 3.12). Spatial vertical menus are activated by placing one's hand in the vertical space above a projected menu marker. By moving the hand up and down it

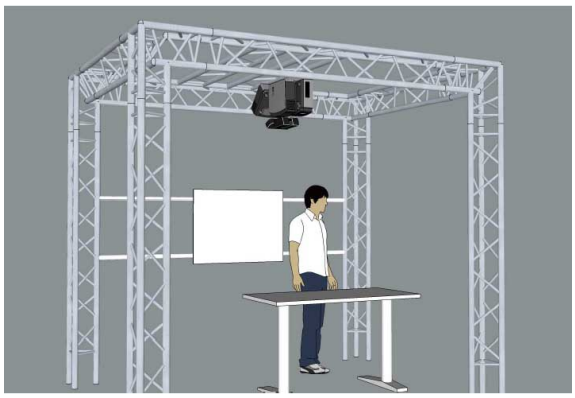

Figure 3.9: LightSpace configuration (image taken from [36])

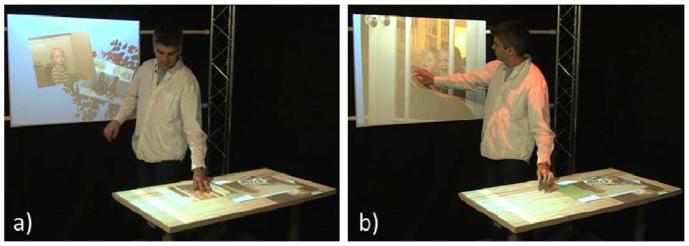

Figure 3.10: Through-body transition (image taken from [36])

is possible to scroll between the various menus options which are projected in the user's hand. It is possible to choose the option by staying in the selected option for more than 2 seconds.

LightSpace is not without its problems. Although it has no technical limit on the number of simultaneous users, six users was found to be the maximum, since beyond that, users were often too close together to be resolved individually.

LightSpace's smart room approach allows interaction with any surface, be it a wall or table, but that may also be one of its flaws, since there are added advantages to a smart room with actual interactive surfaces instead of simulated ones, even if it is less cost effective.

3.2.4 SecondLight

SecondLight is a surface technology which carries all the benefits of rear projection-vision systems while also allowing the extension of the interaction space beyond the surface. Its main feature is a special type of projection screen material which can be switched between two states under electronic control. SecondLight improves on tabletop setups since its ability to leverage the benefits of a diffuser and rear projection-vision for on surface interactions with the option to instantly switch to projecting and seeing through the surface provides the system with the "best of both worlds".

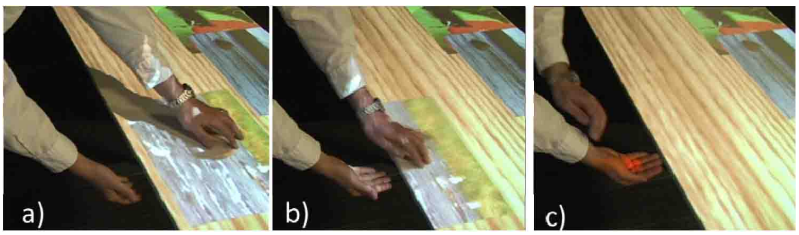

Figure 3.11: Picking up objects from the table (image taken from [36])

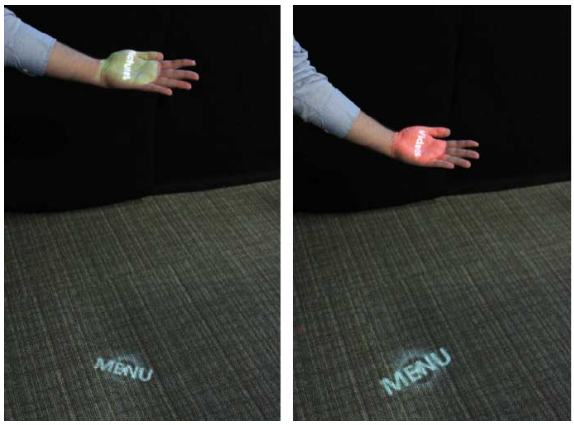

Figure 3.12: Spatial menu (image taken from [36])

The screen material used in SecondLight is described in [17] as an electronically controllable liquid crystal similar to the one used in ”privacy glass” which can switch between transparent and diffuse states as shown in Figure 3.13.

When in its diffused state, SecondLight behaves like a multitouch surface, allowing projection on the surface and detects fingers and tangible objects. When in the clear state its abilities are extended to projection through the surface into objects that have suitable surfaces resulting in an augmented projection. In [17] this augmented projection is explored in a way that relates both projections. For example, in Figure 3.14 a car is projected on the surface while objects above the table reveal the inner workings of the car projected on them through the surface.

It is also possible to track the users' hands from a distance allowing hand gestures and poses to be identified, as shown in Figure 3.14.

In [12] a fiducial method is proposed and built on top of SecondLight. Since SecondLight can switch between states, it can see fiducial markers through the surface, giving it the ability to track objects beyond the surface. These markers are closely related with the reacTIVision [21] markers, using the same mechanism for fiducial orientation and identification. Naturally marker sizes and range had to be taken into account for this new approach.

In [13] this technology is further explored to test new ways of interaction. A shadow feedback technique, as seen in Figure 3.15, helps users connect the user's hand in the real

Figure 3.13: SecondLight switchable screen. Clear state (left) and diffuse state (right). (image taken from [17])

Figure 3.14: Gesture-based interaction (left) Translucent sheets of diffused film being placed above a car to reveal its inner workings thanks to projection through the surface (right), (image taken from [17])

world with the virtual objects in the 3D scene. Shadows are cast for objects and the user's hand and can function as additional depth cues to help the user work around the Z-axis. As displayed in Figure 3.15 a virtual object picked up by the user gets more and more distant as the user lifts it until it turns into its own shadow.

3.2.5 DiamondTouch

DiamondTouch is a multi-user touch technology for tabletop front-projected displays enabling several people to use the same touch-surface simultaneously without interfering with each other as well as enabling the computer to identify which person is touching where.

In [10] research was made on collaborative workspaces in which multiple users work on the same data set. The environment consisted of a ceiling-mounted video projector displaying onto a white table around which the users sit. A single wireless mouse was passed around and it was proposed that collaboration would improve if the users could independently interact with the table thanks to multiple mice. Using different mice in a collaborative environment can be very problematic. Users are faced with the problem of keeping track of one pointer on a large surface with lots of activity. Users feel more eager to point at their virtual pointers to tell other users where they are.

To solve this problem a large touch-screen table surface was proposed and as such the

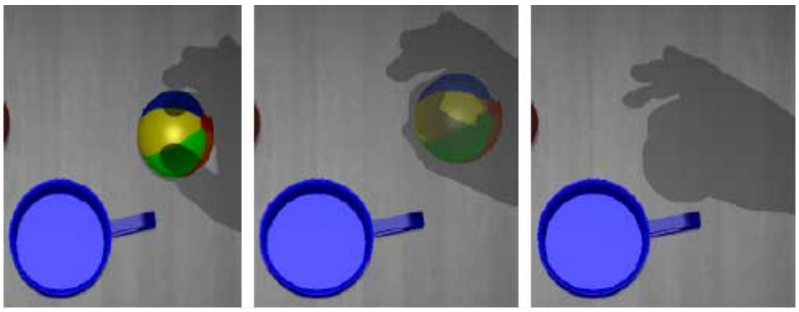

Figure 3.15: Objects and user's hand casting a shadow. [13]

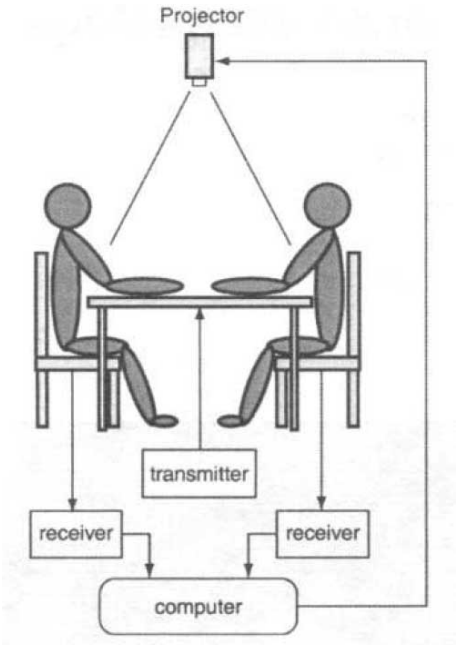

Figure 3.16: DiamondTouch setup (image taken from [10])

following characteristics were considered to be optimal:

1. Multipoint: Detects multiple, simultaneous touches
2. Identifying: Detects which user is touching each point
3. Debris Tolerant: Objects left on the surface do not interfere with normal operation
4. Durable: Able to withstand normal use without frequent repair or re-calibration
5. Unencumbering: No additional devices should be required for use - e.g. no special stylus, body transmitters, etc.
6. Inexpensive to manufacture

The DiamondTouch technology meets all of these requirements. It works by transmitting a different electrical signal to each part of the table surface that we want to identify.

Figure 3.17: ElectroTouch handoff technique (image taken from [20])

When a user touches the table a signal goes from directly beneath the touch point, through the user and into a receiver unit associated to that user. This allows the receiver to determine which part of the table was touched and who was the user that touched it.

This setup (Figure 3.16) has a very precise determination of which user is touching where, which makes it very useful and relevant, even though it is arguably less practical in the sense that a receiver for each user is required and may limit the user's natural movements around the table since they have to sit on the receiver.

3.2.6 *ElectroTouch*

ElectroTouch, seen in [20], provides an interaction technique and an accompanying hardware sensor for sensing handoffs that use physical touch above the table. It detects small electrical signals flowing through users' bodies when they make physical contact, by standing on wire antena pads to create a capacitive connection.

It builds upon the DiamondTouch table [10] but it differs in the sense that it is used to detect person-to-person touch. The premise is simple, users can pick up an object by tapping it on the table, touch hands above the table, and put the object back down by tapping again.

When people interact around a digital surface they often pass objects to others - this action is called "handoff" and it is initiated when a giver and a receiver are present. Since this action has been limited to surface-based interactions it can suffer from friction or even interference from other users [20].

A study was conducted to compare the performance of surface-only handoff techniques like Slide, Flick and surface-only Force-Field to above-the-surface Force-Field and *ElectroTouch* (Figure 3.17). Results showed that above-the-surface handoff techniques had shorter completion times and reduced errors when compared to surface-only techniques. It is suggested that this is due to friction and interference, since these two factors did not occur above-the-table. *ElectroTouch* proved to be the overall best technique since accidental handoffs rarely occurred and the positive tactile feedback that participants received when transferring an object by touching their partner's hand, made the handoff much easier to happen correctly.

3.3 Existing APIs and Applications

As shown in previous sections, in recent years we have seen a proliferation of research exploring the continuous interaction space consisting on interactive surfaces and the area above them. This growing interest has resulted in the development of APIs providing mechanisms to help developers in the creation of interactive applications. In this section we will present some of these APIs as well as other works that would have benefited from an existing API such as our proposed API described in chapter 4.

3.3.1 HapticTouch

HapticTouch framework [23] allows the creation of haptic tabletop applications. While computers typically handle feedback through visual and auditory modalities, haptic interfaces give tactile feedback to users. HapticTouch uses a component responsible for providing haptic feedback called Haptic Tabletop Puck (HTP) [26], which is a tangible device with a fiducial marker indicating its position on the table (Figure 3.18). It contains the following elements:

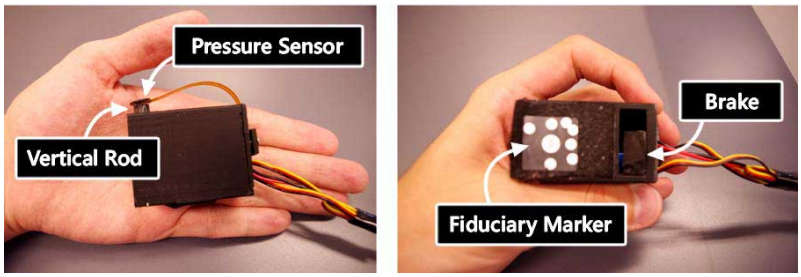

Figure 3.18: HTP's main components (image taken from [26])

- *Haptic output via a movable vertical rod.* A movable rod coming out of a small brick-shaped casing. A small servo motor hidden inside the case controls the up and down movement of the rod.
- *Haptic input, via the rod.* A pressure sensor on top of the rod measures users' finger pressure on it.
- *Friction.* A friction brake at the bottom of the HTP is implemented through a small rubber plate whose pressure against the surface is controlled by another servo motor.
- *Location.* The location of the HTP on the table is tracked through a fiducial marker on its bottom.

The HTP enables three main sources of haptic information:

- *Height*. The vertical movement of the rod represents irregular surfaces and different heights.

- *Malleability*. The feedback loop between applied pressure and the rod's height can simulate the dynamic force-feedback of different materials.

- *Friction*. The brake can modify the puck's resistance to movement in the horizontal plane.

The API's design aims to enable the creation of a wide range of application prototypes and sketches for haptic tabletop environments (HTP) without the need for programmers to understand 3D models or the physics of objects and materials, as well as providing a simple programming interface for haptics development.

The toolkit is layered to promote flexibility while reducing the programming burden for common tasks. The raw layer allows unconstrained hardware acces. The behaviour layer provides contact evens for HTP devices, as well as pre-defined haptic behaviours. A graphical haptics layer uses shapes, images and widgets to associate haptic behaviours to graphical objects. This three layer system gives developers the possibility to choose the layer most suitable for their particular needs.

3.3.2 Interactive space

Interactive space [24] is a framework that allows programmers to develop multitouch and gesture based applications. This framework does gesture recognition through a Microsoft Kinect above the interactive surface. It uses OmniTouch [**?**] as a solution to gesture recognition, which employs template matching of depth data to recognize fingers on a surface or in the space above.

This approach can generate false positives and also has directional limitations, such as fingers only being detected while in a vertical or horizontal position.

3.3.3 Panelrama

Cross-device sharing of data allows developers to create applications that share the user interface between multiple devices. In [38] Panelrama, a web-based framework, is presented to aid in the development of applications using distributed user interfaces (DUIs).

Panelrama provides three main features: easy division of UI into panels, panel state synchronization across multiple devices, and automatic distribution of panels to best-fit devices. It is designed to use existing technologies and facilitate code reuse so that users don't feel the need to re-learn or rewrite applications. This solution categorizes device properties and dynamically optimizes the user interface to these devices.

In [33] a new interaction style that expands across mobile devices and interaction surfaces is explored to support natural interactions. To illustrate this a number of applications are proposed, ranging from a word game that allows users to assemble letters on their phone and drop them onto the shared word board. A calendar application allows users to share their calendar by tapping the surface, while the application is open on the phone.

3.4 Discussion

In this chapter we presented the concept of continuous interaction space in section 3.1 which is the core idea behind our proposed setup and what we set out to achieve throughout our applications and studies. Section 3.2 presented a set of setups that vary from ours in different ways and aspects allowing different types of interactions, while lacking in some of the ones we aimed for. While *Medusa* and *DiamondTouch* have their own takes on multi-user environments identifying which user is responsible for each action, both lack the ability to track gestures and tangibles both on and above the surface, focusing only on touch interactions. *ElectroTouch* builds on top of *DiamondTouch* to study handoff techniques both on and above the table, but could benefit from studying the applications of physical object handoffs instead of just digital ones. *Handsdown*, on the other hand, does provide touch and tangible interaction, but lacks gesture interactions above the surface. *LightSpace* researches various forms of interaction with gestures on and above the surface, with the added advantage of working in any normal surface inside the room, but lacks actual physical object manipulation, furthermore, interactions provide less information to applications since it lacks an actual interactive surface. *SecondLight* comes very close to what we aim for in our setup. It allows touch and tangible interactions on and above the surface, exploring the various possibilities that it has to offer. However, it works through a switchable screen that alternates between a clear and diffused state, meaning that not only is it not very cost effective, due to the special properties of the hardware, it is also not possible to take advantage of both states and, by definition, both modalities at the same time.

Finally, in section 3.3 we present existing APIs that aid developers in creating applications for all of these different types of scenarios each in its own way. TACTIC captures many features from each one of these APIs and sets out to be more with its data merging and abstraction capabilities that allow it to not only be a tool for developers to create applications with all of these scenarios in mind, but also allow existing setups to support new ways of interaction, as described in chapter 4.

Chapter 4

TACTIC API

In this chapter, we present TACTIC, an API combining touch surfaces, tangibles, and the interaction space above the surface, in a way that allows developers to easily combine all these features, and distribute interfaces across multiple devices if required. Additionally, we present the results of a developer study showing how TACTIC is easy to learn and use.

4.1 Overview

TACTIC (Tangible and Tabletop Continuous Interaction) [1] is an API that supports the exchange of information between interactive surfaces, mid-air hand and object recognition and tracking services, and tangible interfaces. It was developed to be used in web applications, thus being accessible to what is probably the most pervasive environment currently.

TACTIC runs on a browser, which makes it easy to deploy in an interactive touch table or smartphone. TACTIC supports the abstraction of touch events, thus enabling the same code base to be used in interactive tables and mobile devices. It allows easily enabling digital objects with interactive behaviours and makes available gesture information, such as which hand and finger are being used, as part of touch and tangible events.

4.2 Arquitecture

TACTIC leverages the communication between client applications and various sources of input. The API's arquitecture is outlined in Figure 4.1. Touch and tangible information are sent through TUIO protocol (section 2.1.2) by Community Core Vision (section 2.1.3) and reacTIVision (section 2.1.4) respectively. The API has a built in component in its data manager to receive this information without the need for additional bridges. However, hand and gesture information, which are handled through the ThreeGear JAVA API (section 2.1.5) require an additional bridge to communicate with TACTIC. To solve this

[1] http://accessible-serv.lasige.di.fc.ul.pt/ tactic/

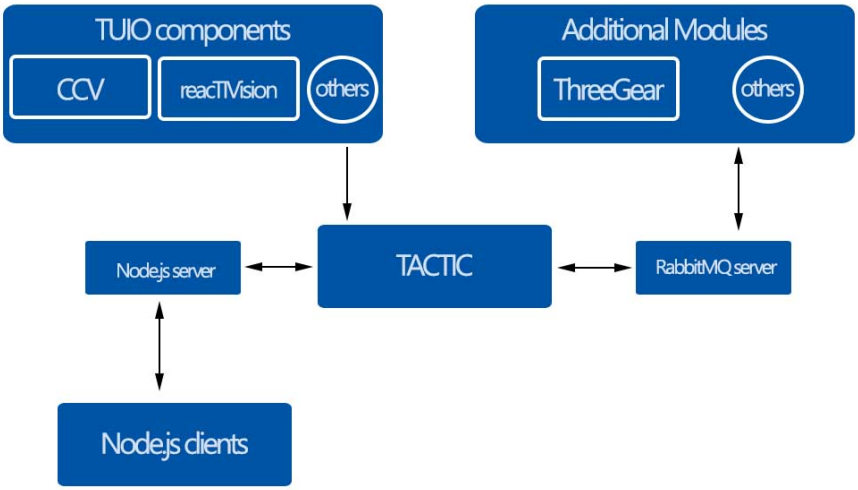

Figure 4.1: API underlying components

problem we deployed RabbitMQ messaging middleware [3] in our system architecture to allow seamless communication between any components. RabbitMQ allows components to publish and subscribe to events, easily bridging different technologies and languages, making our system highly modular since it is easy to add new components without making changes to previous configurations. A node.js module is included for communication between web-based applications, which enables easy development of distributed interfaces.

4.3 Documentation and Coding

This section presents the API's events and properties, as well as a description of the basics of coding with it.

4.3.1 Events

The API is fully implemented in JavaScript to support building HTML client applications. It does all the heavy lifting while providing users with abstract events that contain the information needed.

The following events relate to *on the surface* interactions and are always available as part of the API.

- **object_added, object_updated, object_removed** - Events triggered whenever an object **enters**, **moves** or **leaves** the surface.
- **object.added, object.updated, object.removed** - Events triggered whenever an object **enters**, **moves** or **leaves** an element that is expecting this event.

- **touch.press** - Event triggered whenever a **touch** is tracked **inside** an element that is expecting this event.
- **touch.update** - Event triggered whenever a **touch** already being tracked **moves** inside an element that is expecting this event.
- **touch.release** - Event triggered whenever a **touch** is **no longer** tracked inside an element that is expecting this event.

The following events relate to *above the surface* interactions and are only available when using a ThreeGear (section 2.1.5) based setup and the .jar file made available with the API[2]

- **object_hovering** Event triggered when an object is **lifted** from the surface and while **moving** above it.
- **hand_pinched** - Event triggered when a hand makes a **Pinch** gesture.
- **hand_unpinched** - Event triggered when a hand unmakes a **Pinch** gesture.
- **hand_moved** - Event triggered while hand is detected
- **fingers_moved** - Event triggered while fingers are detected

Events also have data associated to them. Different groups of events hold different sets of information:

- **Touch** events have the folowing data: *location (X, Y coordinates); touch ID*; *Hand* and *Finger* responsible for the touch.
- **Object** surface events have the following data: *location (X, Y coordinates); object ID; Angle*.
- ***object_hovering*** event details the following data: *location (X, Y, Z) coordinates*; *object ID*; *Hand* holding the object.
- ***hand_moved, fingers_moved, hand_pinched*** and ***hand_unpinched*** have the following data: *location (X, Y, Z coordinates)* of each finger and corresponding hand; *hand ID (left, right)*.

When TACTIC is used on a set-up that does not support above the table interactions, all Hand and Finger related data is returned as undefined allowing the API to continue to work without any problems.

[2]http://accessible-serv.lasige.di.fc.ul.pt/~tactic/

4.3.2 Element Properties

Some properties can be easily attached to HTML elements by adding the respective CSS class to them. This way the API saves the user the trouble of making extra calculations. Next we detail a set of classes that can be added to elements and the properties they receive:

- **movable** A movable element is automatically moved by the API whenever a touch is registered inside it and movement follows. When the touch is released the element stays in the new position.
- **touchable** A touchable element receives events related to touch inside the area that corresponds to it, and can then respond to those events (**touch.press**; **touch.update**; **touch.release**) in whatever way the user wishes.
- **object-aware** An object-aware element receives events related to object tracking inside the area that corresponds to it and can then respond to those events (**object.added**; **object.updated**; **object.removed**) in whatever way the user wishes.
- **resizable** A resizable element is automatically resized by the API when two touches are registered inside it at the same time, followed by movement from both touches causing a pinch gesture.
- **rotatable** A rotatable element is automatically rotated by the API when two touches are registered inside it at the same time, followed by a rotation gesture in any direction.
- **resizable_above** A resizable_above element is automatically resized when a Pinch gesture is tracked above it followed by an upward or downward motion.
- **rotatable_above** A rotatable_above element is automatically rotated when an open hand is detected above it followed by a rotation motion to the right or left.

4.3.3 How to use

Events can be bound to elements to add functions to specific situations throughout the code. For example, if the user wishes to make an element aware to touch events the only requirement is for the element to have the class *touchable*.

```
<div class= "button touchable"></div>
```

Elements that are *touchable* will receive **touch.press**, **touch.update** and **touch.release** events. These events can be handled by binding the element to the event and adding a function that works as the event handler.

The following code produces “Pressed at 300,200 with hand RIGHT and finger INDEX” when a user touches an element of class *button* with their right hand and index finger at the HTML window’s position *300,200*.

```
$('.button').bind('touch.press',
                  function(event, data) {
    alert("Pressed at "
          + data.x + "," + data.y
          + " with hand " + data.hand
          + " and finger " + data.finger);
});
```

The integrated node.js component allows users to send and receive messages between web applications easily without having to initiate any variables or messaging protocols. These messages can be sent in a network environment allowing applications to communicate in a cross-device setting. The following code shows how to subscribe to messages and how to send them.

```
socket.on('message', function(msg){
        console.log(msg);
});

socket.emit('message', "hello");
```

Finally, the RabbitMQ messaging framework allows communication between different technologies and languages making it easy to add new modules to an application. Users can send information back and forth from other languages such as Java or Python to their Web applications with the following commands.

```
// Subscribing to data (example)

MQ.queue("auto",{autoDelete:true}).bind(
        "handInfo", "*").callback(
   function(m) {
           console.log(m.data);
   });

// Publishing data (example)

MQ.topic('handInfo').publish({
        //...
        //place object here
        //...
}, 'app.finish');
```

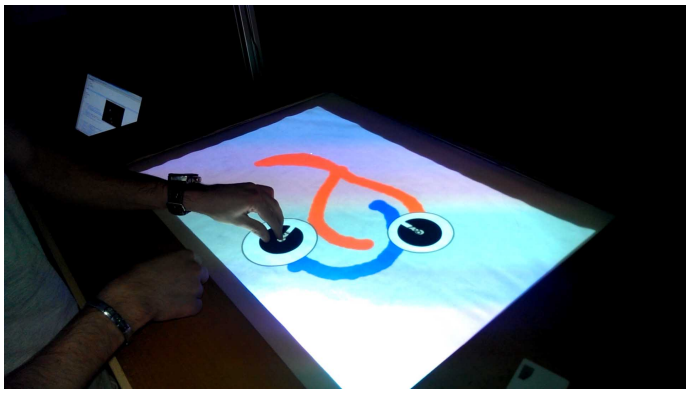

Figure 4.2: API controlling *object_followers* for tangibles

4.4 Implementation

This section describes how TACTIC can not only solve existing problems, but also allow new types of interaction.

4.4.1 Solving Occlusion

Since fiducial tracking is supported by a camera that captures visible light (section 2.2.1), surface projections can get in the way and cause missing fiducial markers in a tracking scenario. To solve this problem, we searched for a background color that would allow easy and full fiducial tracking on the surface and applied it to an *object_follower*. An *object_follower* is a circle that surrounds the fiducial when it is tracked for the first time and constantly moves below it, while maintaining itself above any other projection (Figure 4.2). Optionally, tangibles can be rotated to increase or decrease the radius of *object_followers*. This way it is guaranteed that the color below the fiducial will always be the desired color for tracking and significantly reduce the probability of miss-tracking a fiducial marker.

4.4.2 Merging information

TACTIC is not limited to only providing information from different sources of input. It is able to create new ways of interaction by merging and interpreting its pool of data.

Touch interfaces are not, traditionally, able to detect which finger or hand is responsible for each touch. However thanks to the possibility of merging our CCV and ThreeGear sources, TACTIC is able to provide touch events to users, detailing the hand and finger that is responsible for each single touch.

Fiducial markers, which are tracked by reacTIVision, are restrained to the table surface, since the camera can not detect markers that are not pressed against the acrylic. This limits tangibles to *on the surface* interactions. TACTIC is able to provide tangible interac-

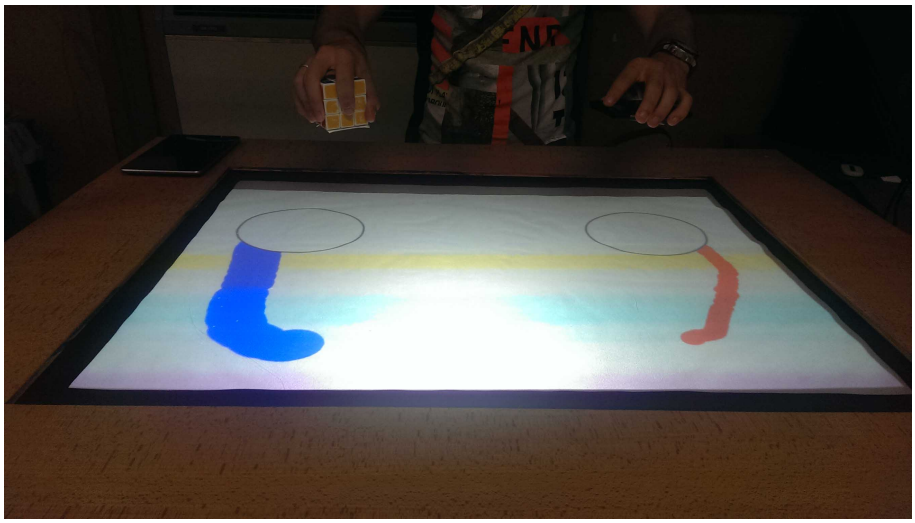

Figure 4.3: Tangible tracking above the surface, represented by white circles on the surface

tions *above the surface* contributing to the continuous interaction space effect. Tangible interactions above the surface are inferred through gesture recognition. After a tangible is placed on the surface, it is registered in the API's Data Manager. When the tangible leaves the surface TACTIC tracks the hand holding it at that moment. By continuously tracking a series of hand states (*position*, *closed*, *open*) it is also possible to detect when tangibles are exchanged from one hand to another above the surface, as well as when they leave the interaction area. This continuous tracking is confirmed in the form of a circle constantly moving below the object that is being held above the surface (Figure 4.3).

4.4.3 Backend processing

TACTIC processes a great deal of information from different sources to achieve the abstractions provided to users. In this section we look at how this information is gathered and processed.

4.4.4 Calibration

All Hand Tracking information is received from ThreeGear through RabbitMQ. The API has a calibration mode to calibrate ThreeGear incoming coordinates to any setup screen. This is done by running the calibration app in the API followed by a *Pinching* action on the top most, bottom most, left most and right most parts of the screen. This way all coordinates are calibrated to these bounds before being sent.

Events

Touch information from either TUIO or mobile browsers is treated in the same fashion. Any touch is mapped with a corresponding ID and, if available, information regarding *hand* and *finger* responsible for the touch. This is done by searching the current pool of hand tracking information for the hand and finger that are closest to the touch point, based on the surface's *X,Y axis*, and cross-referencing with other existing touches to avoid same finger matching, thus preventing inaccurate results. The results are very accurate even in cases where two hands or fingers are very close to the target. Finally the touch action is published as an event to all *touchable* elements that contain the area of the touch.

Tangible information *on the surface* is mapped with a corresponding ID, fiducial angle and, if available, information regarding the *hand* that is dragging the object. This is done, again, by searching the hand tracking information for the hand that is closest to the tangible's position and cross-referencing with other tracked tangibles that may already be held with that hand, thus avoiding possible tracking errors. Next an *object_added, object_updated* or *object_removed* event is published to all *object-aware* elements, while *object.added, object.updated* or *object.removed* events are published only to *object-aware* elements that contain the area the touchable is tracked in.

When tangibles enter the table for the first time they are registered in a list of *on the surface* tangibles that controls currently detected objects. This way when an object leaves the surface, if there is any hand tracking information available, instead of triggering an event to report the object removal, the last known position of the tangible is mapped to existing hand information to search for the closest hand that is not registered as holding any object. TACTIC, at this point, assumes that this nearby hand has to be holding the object at the instant of the surface removal. From then on, any hand information from that hand is followed by a publishing of the *object_hovering* event that details the ID of the object that is registered as being held, the hand's position, and the hand's ID. This ends when either the hand disappears from the interaction area, which treats the object as removed, or when an object with the same ID is tracked *on the surface*, which means that it was dropped on the table, followed by the appropriate *on the surface* tangible events.

Hand and finger information area treated in the same fashion. Both contain arrays of data with positions and IDs. Each of these positions are then scaled to the current HTML window's width and height, in order to achieve accurate and calibrated positions. Any hand and finger produce the *hand_moved* and *fingers_moved* events respectively. In case a *Pinch* or *Unpinch* event is received from the ThreeGear API, it is forwarded as *hand_pinched* or *hand_unpinched* with the corresponding hand information.

Element properties

- **Movable** - When a touch ID is tracked for the first time inside a *movable* element, that ID is registered as moving that element. From then on any *touch.updated* event

with the same ID automatically feeds the element's CSS properties to match it, causing the element to move with the touch in a dragging fashion.

- **resizable** - When two touch points are recognized at the same time inside a *resizable* element, the API begins to feed the two points distance relatively to the starting point as size to the element's CSS, causing a pinch effect, similar to what is seen in mobile environments.

- **resizable_above** - When a *hand_pinched* event is detected above a *resizable_above* element, the system begins to feed the corresponding hand's Z coordinate to the element's CSS, until a *hand_unpinched* event is detected. Consequently, the element expands when the hand is closer to the table and shrinks when it is farther from it.

- **rotatable** - When two fingers are tracked at the same time inside a *rotatable* element, the API starts keeping tracked of the angle that is formed between the initial state of both fingers and any followed positions updates, and feeding this angle information to the element's CSS, causing a rotation to take place.

- **rotatable_above** - When a hand is tracked on top of a *rotatable_above* element, the *thumb* and *pinky* fingers are registered as the two starting points for the rotation. From then on, the API keeps track of the angle formed between the initial state of both these fingers and any followed position updates, feeding this angle information to the element's CSS, causing a rotation to take place.

4.5 Validation

A developer study was conducted to investigate ease of learning and use of the TACTIC API. In this section we describe the study followed by a discussion of its results.

4.5.1 Participants

Five participants were chosen (1 female, 4 male) between ages of 22 and 27 to test our API by developing a test application. All participants were experienced web developers. From the pool of projects all participants were involved during the last year, a total of 7 projects dealt specifically with mobile web applications. Just a single project included touch interaction that did not directly relate with mobile devices. No project involved tangible interfaces.

4.5.2 Tasks

Participants were tasked with developing applications that would require knowledge of different aspects of the API as well as a few JavaScript and CSS basics. Our purpose

was to understand how easily and fast users could build complex applications using TACTIC. To achieve this, users were tasked with developing a painting application that would incrementally gain complexity as well as use more API functionalities.

Task 1 - Build an HTML page that displays 3 buttons representing the colors Red, Green and Blue and 3 buttons representing Small, Medium and Big brush size. This task was designed to get users to build a standard HTML page with no required API functionality, which will allow them to work on their own code through the next tasks.

Task 2 - Add touch functionality to the previous page to build a paint application. By touching the color buttons a new color is chosen, and by touching the size buttons the size of the brush is chosen. By touching any other area, the canvas is painted with the chosen brush color and size. The goal of this task is to understand how users adapt existing pages to the API and how they use its touch events.

Task 3 - Add tangible functionality to the previous page so that all previous interactions can be done with objects as well. We wanted to study how users used tangible events and how the API promotes code recycling.

Task 4 - Add above the table interactions, requiring painting to be done above the table exclusively. This allowed us to study how users used above the table events and further understand patterns of code reuse.

Task 5 - Add cross-device functionalities building a new mobile application. When a color is chosen, it is sent to the smartphone page changing its background color and painting above is only done while touching the smartphone's screen. We wanted to study how users employed the node.js communication component to build cross-device applications, as well as the abstraction of touch events for both mobile and tabletop settings.

4.5.3 Procedure

Trials started with a profile questionnaire. After the questionnaire, a brief overview of the API would follow explaining the basics of the documentation and how everything worked. Next users were asked to do each one of the tasks while task duration and written code were stored.

When all tasks were completed, another questionnaire would follow to let us know what users thought of the TACTIC API. Users were asked to express how easy the event, classes, communication, cross-device and tangible functionalities were to understand on a Likert scale of 0 to 9, with 0 being terribly hard and 9 being perfectly easy.

4.5.4 Results

During the trials we collected the time to complete each task and snapshots of the code at the end of each task. Developers took an average of 14.3 minutes to complete the first task (SD=2.8 minutes). The second task, the first one requiring the use of the API, was

Feature	Average (Standard Deviation)
Events	8.2 (0.84)
Classes	8.4 (0.55)
Communication	8.6 (0.55)
Cross-platform	8.2 (0.84)
Tangibles	8.2 (0.84)

Table 4.1: Average and standard deviation of the questionnaire results.

completed on average in 8.3 minutes (SD=1.7 minutes). The third task was quicker, being completed in 2.9 minutes on average (SD=30.8 seconds). The fourth task was the quickest one, completed in 37.7 seconds on average (SD=23.6 seconds). The fifth and final task, which introduced a mobile device to the application, was completed in 6.4 minutes on average (SD=1.8 minutes).

In what regards the analysis of the code written, we have analyzed how many lines were changed and how many new lines were written between each task. The analysis considers all HTML, CSS and JavaScript files produced. For the initial task, developers wrote on average 63 lines of code. The second task asked developers to include the API in their page, and to perform the painting through touch. This resulted, on average, on 76 new and 6 changed lines of code. The new lines were mainly responsible for performing the painting. The changed lines introduce the touchable behaviour in existing page elements. For the third task, developers were required to introduce tangibles for painting. All developers changed exactly 14 lines of code in this task, without any further changes. In the fourth task, the painting needed to be performed through gestures above the table instead of touching. This was achieved by all developers with the introduction of a single line of code, and a change in another line of code. Finally, the last task introduced a mobile device to control the painting. This led developers to write an average of 9 new lines of code for the page that was being displayed on the interactive table, and a page to be displayed on the mobile device with 45 lines of code on average.

After completing the tasks, trial participants completed a questionnaire about how easy it was to understand and use the TACTIC API. We asked them to classify the API's events and classes, and the API's support for communication, cross-platform development and tangible interaction. The results are summarized in table 4.1.

As can be seen from the results, the developer's impressions are overwhelmingly positive. Additionally we collected their opinions after the trials, which support these results. All developers expressed their happiness with how much they were able to achieve in such a short time (the longest session - D3 - took less than 40 minutes). Additionally, all participants pointed out as a positive point the fact that to use the API they were not required to learn something specific regarding the technology used to support touch, object recognition and tracking, or tangibles, and they could rely on their existing JavaScript and jQuery knowledge. Other positive factors mentioned include the intuitiveness of the

names of events and classes (D2), the ease of use of the API (D2, D4 and D5) and the small amount of time required to learn it (D4).

4.5.5 Results' Analysis

Tasks 1 and 2 comprised building the main blocks of the painting application, which justifies their being the longest ones. In these tasks, developers had to build the elements to select brush color and size, and the code to do the painting. The API was used solely in task 2 to endow the HTML canvas and button elements with touchable behaviour.

From task 3 onwards, we expected to see the benefits of the API. In task 3, developers were asked to replace usage of touch by interaction through tangibles. Results for this task demonstrate that developers understood how to use the framework. All developers did the same thing in this step, which was simply to change the event and binding that was attached to each interactive element (6 buttons and the canvas), resulting in 14 changes to their code. This reveals they understood the way of using the framework, and furthermore, it did not take them too long to do so. On average they took under three minutes.

Task 4 asked to replace the interaction on the tabletop with interaction above the table. The fact that the longest it took one participant to complete it was 64 seconds, reveals that by this point all developers had fully grasped the API grounding concepts, and could very easily and quickly apply them. Once more, they all performed the same actions: introduced one line to start using the ability to track objects above the table, and changed one event processing line to begin responding to object hovering events.

The final task included distributing the application to another device. A mobile device was now used to control when the painting was performed, which meant the operation of the mobile device had to be communicated to the table application responsible for painting on the interactive surface. This task implied changes to the HTML page loaded on the table's browser, and the creation of a new HTML page to be loaded on the mobile's browser. Changes to the table's page were small (under 10 lines of code on average). All the changes and new code were completed in about 6 minutes, which is another sign that the API's way of operating had been well understood by this point.

Overall, from the analysis of the code and the questionnaires, the API proved to be easy to understand and use. It promoted code reuse, and enabled developers without experience with tangible interaction, distributed interfaces and object tracking, to incorporate those features in their Web applications.

4.6 Discussion

In this chapter we presented TACTIC, an API supporting the development of web applications aware of touch events, object recognition and tracking on and above interactive surfaces, gesture recognition and cross-device communications. The API's architecture

and documentation were thoroughly described coupled by a brief tutorial on its usage. The API's implementation was detailed, showcasing how data is combined at various levels providing additional information to events.

We have demonstrated TACTIC's ease of learning and use through a study with five experienced web developers, who found TACTIC's concepts easy to grasp and use.

Chapter 5 will present a user study on gesture performance conducted on our existing setup (chapter 2) using TACTIC. Chapter 6 will describe how TACTIC was used to build applications that can create new situations and allow new types of studies in the accessibility field.

Chapter 5

Gestures

This chapter presents a user study, comparing the performance of zoom and rotation tasks on and above an interactive surface.

5.1 Methodology and Setup

It is becoming easier to integrate gestures made above an interactive surface with traditional multitouch interaction on the tabletop. While the latter has been the subject of several studies, the first is still lacking studies with the same level of detail. Furthermore, comparisons of the performance of similar gestures on and above tabletops are also missing.

5.1.1 Objectives

This study aims at comparing user performance, when performing gestures on and above interactive surfaces. In this fashion, the study contributes to the knowledge about interactive gestures, complementing existing characterization of gestures either on tabletop surfaces, or in mid-air. This is the first study that compares the same actions both on tabletop and above the surface.

5.1.2 Gesture Characterization

Zoom and Rotate were the two actions selected for this study. *Pinch* and *Rotation* gestures are commonly used to perform these actions in smartphones and tablets. Consequently these were the gestures employed in this study. On the tabletop, the gestures were performed by touching the surface with two fingers and pinching (Figure 5.1 A) or rotating (Figure 5.1 C) the fingers, as usual. Given that above the surface there is no surface where to rest the fingers, the gestures used where slightly modified. To zoom, users were requested to pinch their fingers (Figure 5.1 B), and could then control the zoom level by moving the hand closer (zoom in) or further (zoom out) from the table. To rotate, users

were requested to place their hand open above the table and rotate the hand on the plane parallel to the table's surface (Figure 5.1 D). While both *Rotation* gestures are very similar in their execution, the *Pinch* gestures could not be as similar because it is impossible to guarantee, with the used setup, tracking of both the thumb and index fingers without occlusion. Therefore, pinching selects the action, while the hand distance to the table controls the amount.

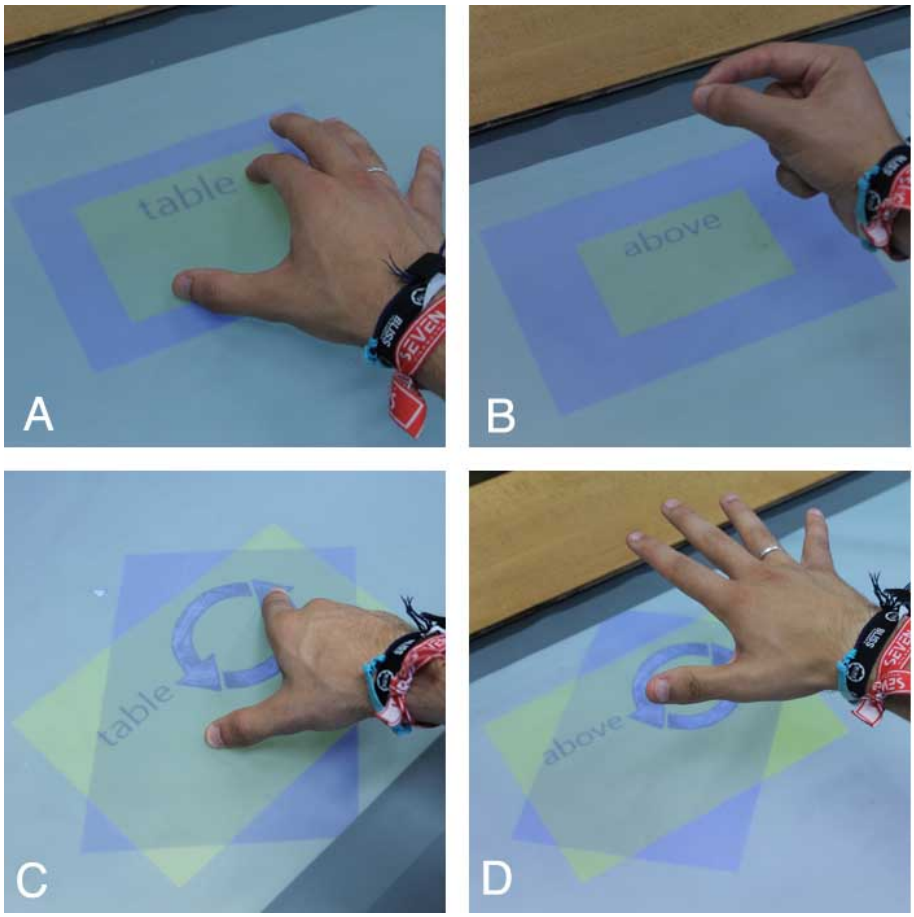

Figure 5.1: (A) Pinch gesture on the table; (B) Pinch Gesture Above the table; (C) Rotation gesture on the table; (D) Rotation gesture above the table. In yellow the initial object, in Blue the target placement.

5.1.3 Experimental Setup

Tasks were performed on our proposed setup described in chapter 2.

Before the full study, a pre-study was conducted to validate that the different technologies used on and above the table did not alter the performance of the study participants and ensure the same level of responsiveness. After performance improvements to the above the table setup, the same level of responsiveness was achieved.

5.1.4 Participants

Sixteen participants were chosen between an age range of 21 to 27 years (6 female, 10 male). All but one participants were right handed. None have any type of impairment regarding vision or dexterity.

5.1.5 Tasks

Participants were asked to perform a series of zoom and rotation tasks. For both tasks an object was displayed on the table's surface, together with a zoomed or rotated version of the same object, represented in a different color. Participants were asked to match the object with the target.

5.1.6 Independent Variables

The main factor in the study is the Area where the action is performed: on the table, or above the table. For Zoom tasks the following additional factors were controlled: *Direction* (zoom in, zoom out); *Distance* (small – 91 pixels, big – 170 pixels). For *Rotation* tasks the controlled factors were: Starting Angle (0°, 60°, 120°); *Rotation Amount* (45°, 90°); *Direction* (clockwise, counter-clockwise). The choice of factors was influenced by the experiments in [14] and [15].

5.1.7 Dependent Variables

For each task, data regarding total number of *movements* (gestures before completion), *error* (distance to target size or angle) and *duration* was collected. For each trial we collected the participant's preference for touch interaction or above the table interaction in a 7 point Likert scale.

5.1.8 Procedure

The experiment followed a within-subjects design and was divided into two trials (*Zoom* and *Rotation*). *Pinch* trials included 96 tasks (*Area* (2) x *Direction* (2) x *Distance* (2) x repetitions (12)). *Rotation* trials had participants perform 288 tasks (*Area* (2) x *Starting Angle* (3) x *Rotation Amount* (2) x *Direction* (2) x repetitions (12)). In order to eliminate possible positioning effects, targets were randomized between 6 positions on a 3x2 grid. Each participant matched a target twice in each position, resulting in 12 repetitions in each trial. A total of 6144 tasks were completed. At the start of each trial, participants had the chance to practice with a number of targets to learn how tasks would be presented and how to complete them. Trial order was alternated between participants. Tasks were distributed randomly in a combination of all factors. Each task began with an object in one of the combinations, as well as a target size, or rotation to achieve. The task would

end when the participant signaled to be satisfied with the object placement. However, for *Duration* measurements we considered the period from the participant's first gesture until completion of the last gesture. After the end of a trial, participant's preference was collected.

5.1.9 Analysis

Performance measures were analysed with a repeated measures analysis of variance (ANOVA). Post-hoc tests were made for the analysis of significant main effects for factors with more than 2 levels. All significant results reported consider $p < 0.01$.

5.2 Findings

This section begins with an overview of the effects of the main factor of the study, *Area*, before proceeding with the report of the effects of the other factors on the performance of *Zoom* and *Rotation* tasks.

5.2.1 Area Effects

Area proved to impact several performance measures, both on *Zoom* and *Rotation* tasks. Overall, participant performance was better when conducting tasks on the surface. Zoom tasks were faster ($F(1.0, 1.191) = 151.71$) and required less movements to complete ($F(1.0, 1.191) = 16.01$) on the surface. For zoom tasks, *Area* had no effect on the error made. Rotation tasks made on the surface were faster ($F(1.0, 1.191) = 376.54$), needed less movements ($F(1.0, 1.191) = 76.51$) and resulted in smaller errors ($F(1.0, 1.191) = 204.94$). The participant satisfaction with both interaction modes followed the same tendency. For zoom tasks, a Wilcoxon signed-rank test showed that participants prefer the interaction to take place on the table ($M = 6.25$, $SD = .68$) instead of above the table ($M = 4.69$, $SD = .95$), $Z = 3.54$, $p < 0.01$. For rotation tasks, the Wilcoxon signed-rank test showed that participants also prefer the interaction to take place on the table ($M = 5.56$; $SD = .814$) instead of above the table ($M = 3.38$, $SD = 1.26$), $Z = 3.57$, $p < 0.01$.

5.2.2 Zoom Tasks

Duration

An analysis of *Direction* showed that zoom out tasks were performed faster ($F(1.0, 1.1191) = 34.9$) than zoom in tasks. Additionally, the interaction between *Area* and *Direction* showed that this effect is more pronounced when performing the zoom out tasks above the table ($F(1.0, 1.191) = 43.86$).

Movements

Zoom *Direction* also impacted the number of movements required to complete the task. Zoom out tasks required less movements (F(1.0, 1.91) = 8.87). The interaction between *Area* and *Direction* showed, once again, that this effect is more pronouced above the table (F(1.0, 1.191) = 26.67).

Error

The only factor to impact the error made by participants in *Zoom* tasks was *Distance* (F(1.0, 1.919) = 7.58). Smaller zoom tasks lead to smaller errors when resizing the objects.

5.2.3 Rotation Tasks

Duration

Direction showed that clockwise rotations were faster (F(1.0, 1.191) = 7.84). As could be expected, *Rotation Amount* showed that 45° rotations were faster (F(1.0, 1.191) = 254.48). The interaction between *Area* and *Rotation* showed that this effect was more pronounced above the table (F(1.0, 1.191) = 84.25).

Movements

Once again, as expected, *Rotation Amount* of 45° required less movements (F(1.0, 1.191) = 432.22). An interaction between *Area* and *Rotation Amount* showed this effect to be more pronounced on the table (F(1.0; 1,191) = 23.49). Another interaction, this time between *Direction* and *Starting Angle* showed the effect of hand positioning, as described by the starting angle, has on rotation movements in different directions (F(1.0, 1.191) = 12.03). Different starting positions impact the movement in contrasting ways depending on the direction of the movement, as can be seen in Figure 5.2.

Error

Contrarily to what happens in the *Zoom* task, when participants performed rotations, several factors contribute to the error made. In addition to already mentioned *Area* effect, an analysis of *Direction* revealed that counter-clockwise rotations resulted in larger errors (F(1.0, 1.191) = 10.08). Also the *Rotation Amount* revealed that 90° rotations resulted in larger errors (F(1.0, 1.191) = 7.37). Interaction between *Area* and *Direction* (F(1.0, 1.191) = 9.35) and Area and *Rotation Amount* (F(1.0, 1.191) = 7.12) showed once more that the effects are more pronounced above the table.

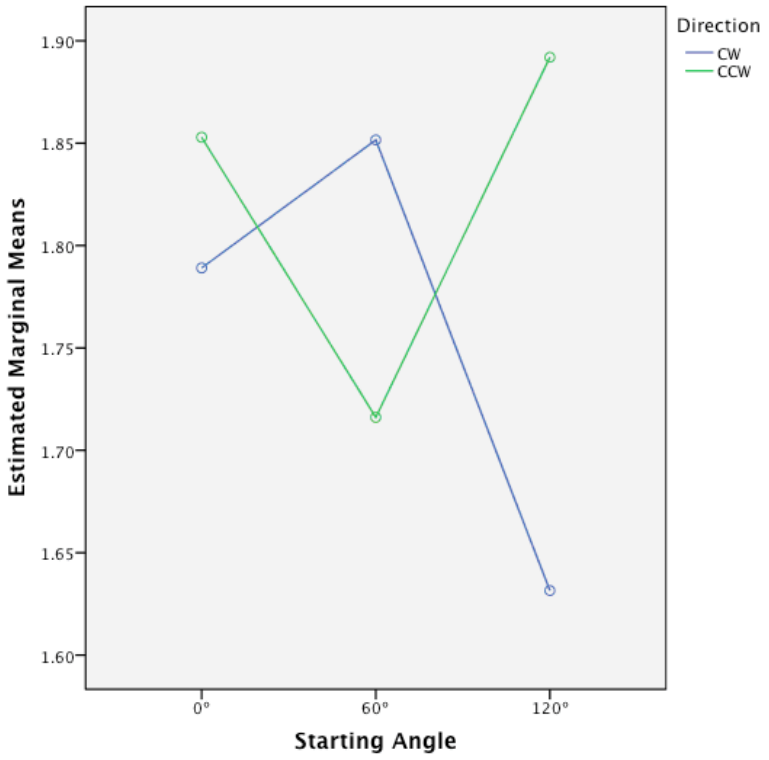

Figure 5.2: Average number of movements for the rotation tasks, by direction of movement and starting angle of the object.

5.3 Result Analysis

The analysis of the main factor being studied, *Area*, showed that tasks on the table's surface are performed faster, with less movements and resulting in smaller errors, when compared to tasks performed above the table. There are several reasons contributing to explain this result. Gestures on mid-air require managing an extra degree of freedom. Even tough the gestures used in the experiment tried to minimize this fact (with the zoom gesture being performed in a plane perpendicular to the table, and the rotation gesture in a parallel plane), user's still have to manage the extra degree that performing gestures on mid-air presents and makes them more tiring. Additionally, mid-air gestures do not afford the direct manipulation of the objects displayed on the interactive surface. Finally, trial participants were representative of current owners of smartphones and tablets that are acquainted with touch interaction, thus being familiar with on the table interaction.

Accordingly, *Area* showed to have a significant effect on all the performance factors analyzed. Both *Zoom* and *Rotation* tasks were performed faster on the table. The same occurred with the number of movements. Finally, the error for *Rotation* tasks was higher above the table, with the error in *Zoom* tasks being the only measure that failed to be significantly influenced by the *Area* where it was performed.

While this finding is not surprising, there are other interesting findings in this study.

It was possible to observe from the interactions between *Area* and other factors, more detailed nuances of its impact. One common observation across different measures is that performance effects are more pronounced when tasks are executed above the table. This shows that, not only performance on the table is superior to performance above the table, but also that when performing tasks above the table, changes in the within-factors consistently result in larger differences in performance factors. From an interaction design perspective, this recommends avoiding the use of the same configuration for interaction techniques in a continuous interaction space comprising the table and the space above it, because their effects vary from surface to space above. For example, the zoom factor associated with the table surface should be different from the one used above it (in this study, the zoom factor was the same in both areas). It will be interesting to study if this behaviour occurs consistently with other interaction mechanisms – e.g. scroll.

Another important consideration for the design of an interactive system with these characteristics relates object placement, action design and human ergonomics. One finding from the study is that direction of rotation (action design) together with the starting angle (object placement and orientation) led participants to perform more movements to complete the desired task. This is a consequence of more or less awkward hand and wrist positions that participants had to adopt. While in a purely virtual environment, users would adapt their starting hand position, thus manipulating the starting angle, this might not be the case if the system design requires the user to rotate an object using specific points of the object. Additionally, when considering a system that uses tangibles, it might not be possible for the user to reorient their hand and wrist while grasping an object.

It is also possible to analyze the performance of the suggested gestures for the actions executed above the table. Although the rotation gesture above the table was more similar (rotation on a parallel plane) to the one on the table than the zoom gesture (translation on a perpendicular plane), the zoom gesture performance was arguably better, with the zoom out action in particular reaching levels of performance similar on and above the table. This interesting finding suggests that adding an above the table zoom (or zoom out only) gesture to a multitouch interactive setup might lead to increased performance. One possible explanation for this derives from the possible occlusion that a hand touching the projected surface creates. By having the hand above the surface instead of on top of it, this occlusion would become a smaller factor (even smaller when zooming out, since the hand is moving away from the surface, which would contribute to explaining why zoom out performance was better that zoom in).

Finally, a comparison between this study and previous studies findings regarding specific aspects of zoom and rotation gestures is made. The better performance in zoom out tasks (requiring less time and movements) compared to zoom in tasks is in accordance to what was found in [14], where contracting pinch gestures (i.e. *Zoom out*) were found to be faster to complete and ergonomically easier. In what concerns rotation tasks, the

expected result that smaller rotation angles lead to faster movements, smaller errors and require less movements is in accordance to [15], where it was found that duration and ergonomic failure rate increase with rotation diameter. Curiously, this study found that clockwise rotations are faster. This is contrary to what Hoggan et al. [15] report, where clockwise rotations took generally longer. Furthermore, in this study, counter-clockwise rotation tasks required less movements to complete, but resulted in bigger errors.

5.4 Discussion

This chapter presented a user study (N = 16) where the performance on *Zoom* and *Rotation* tasks was compared in two settings: an interactive tabletop surface and the also interactive space above the surface. The study compared the effects of the area where the gestures were being made, as well as other factors. This was the first study on the performance of gestures above a tabletop, and the first performance comparison of the same task done on and above a tabletop.

The study confirmed that performance on the surface is greater than performance on the space above it. Other findings from the study cover the different impact that the area where the gesture is performed has on its outcome; the relation between task mechanics and human ergonomics; and the benefits that can arise from endowing multitouch surfaces with the ability to recognize above the surface gestures. These results are useful for informing the design of applications that explore a continuous interaction space, comprising an interactive tabletop surface and the area above, a trend that is becoming more common given the increased availability of the technology to do it.

Chapter 6

TACTIC applications

In this chapter we present a set of applications showcasing TACTIC's core features, followed by a contribution towards a user study on blind people and the possibility of expanding our setup to support collaborative scenarios

6.1 Showcasing TACTIC

In this section we present a set of applications in order to display some of TACTIC's capabilities ranging from multiple types of interactions and interfaces. These applications were tested in our existing setup.

6.1.1 Touch

Our first application shows how to use TACTIC's classes and events to quickly build a functional touch application. The page's body is made of a single DIV element. By adding the **touchable** class, this element is able to receive touch events. By binding a function that changes the color of the element to yellow when receiving **touch.press** and another to change it back to red when receiving **touch.release**, the application gives us actual touch feedback for touches, as seen in Figure 6.1. We also add other classes to the DIV to make it react to gestures. By adding **movable**, **resizable** and **rotatable** to the DIV class, it will automatically respond to single touch dragging by moving with the finger, as well as Pinch gestures for resizing and Rotation gestures for rotating.

6.1.2 Tangibles

For our second sample application we wanted to demonstrate some of TACTIC's tangible events. We devolped an object scanner for tangibles, showing feedback for each object as it is identified. A single DIV is placed on the page and the **object-aware** class is added to it so that it can receive tangible events. We bind a function that displays the object ID on the screen to the **object.added** event. The object ID number is matched with a

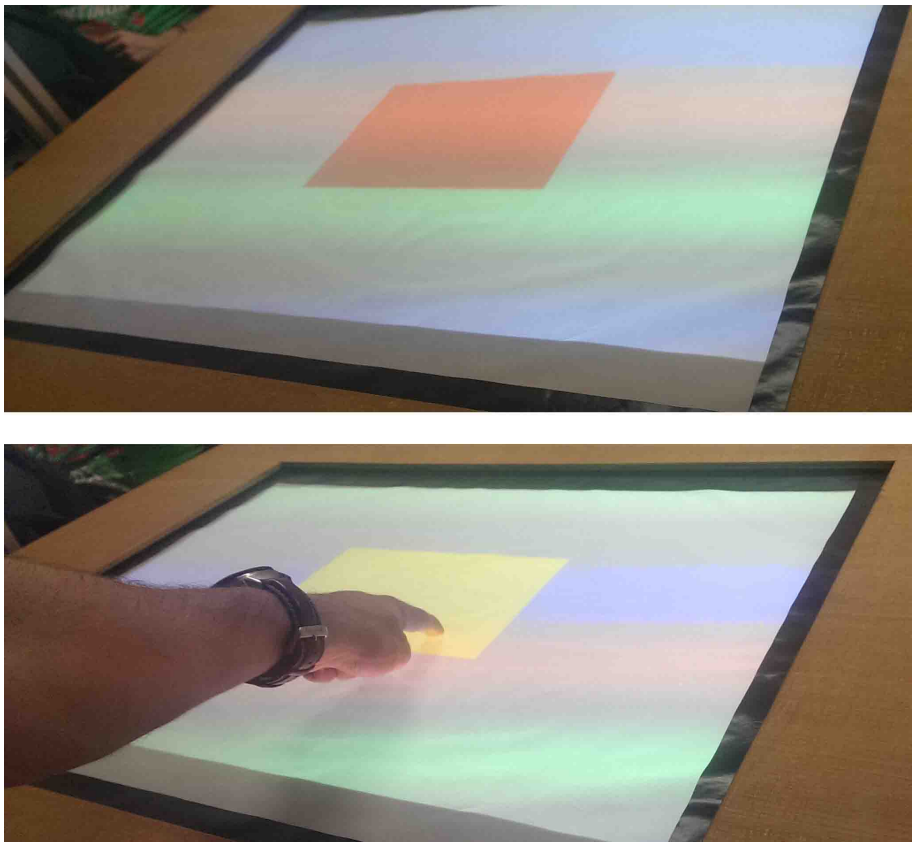

Figure 6.1: Upper image showing touchable element; lower image showing touchable element being pressed

previously set list of real objects to provide a textual feedback of the object. We also bind the **object.updated** event to a function that increases or decreases the text size of the screen depending on the object's angle of rotation, causing it to change in real time as the object rotates, as shown in Figure 6.2.

6.1.3 Above the surface

In order to show how TACTIC keeps track of objects above the surface, our third application allows objects to paint while hovering, with a specific color assigned to each object and a specific size brush assigned to each hand. To achieve this, a canvas is created in the page's body. The canvas is bound to the **object_hovering** event, which triggers from the moment an object is lifted from the table surface and continues to trigger until it leaves the area or returns to the surface. This event contains information regarding which hand is holding which object and it can detect when the hand holding the object changes. In the bound function a set of clauses determines that if the hand holding the object is the right one, then the paint brush will be thick and if not then it will be thin. Objects are assigned colors, making the current color red for the phone and blue for the cube (Figure 6.3).

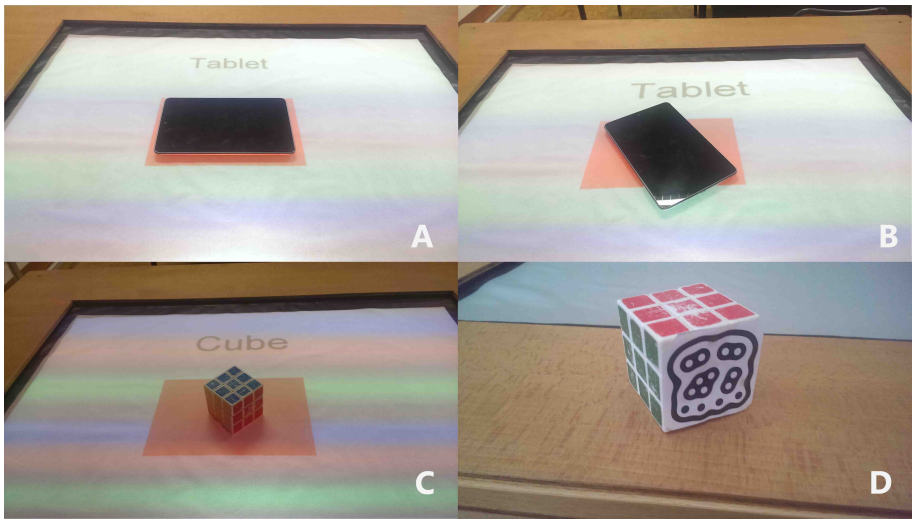

Figure 6.2: (A) Tablet recognized in application; (B) Tablet rotation increases text; (C) Cube being recognized in application; (D) Fiducial marker on Cube

6.1.4 Device communication

To demonstrate TACTIC's communication capabilities and easy cross device coding, we developed a table application that has random objects spread out on the surface. Mobile devices can drop on top of them and "steal" them into their screen. The user can then drop them again on the table surface, while hovering, with a touch on the phone screen. A second smart device can also come near the other device and "steal" the element from it. Three elements are randomly generated and all have the class **object-aware** in order to receive the **object.added** event when an object enters them. When this occurs the element information is sent to the page associated with the tangible ID which will be waiting for messages on the smart device. When the smart device has an element inside it, it starts to scout for other nearby objects using the **object_hovering** event. If another object gets close to it, the element is sent to that smart device's page since it is also waiting for messages. At any given time a hovering smart device can be touched, triggering a **touch.press** event to send an element inside it to the hovering position on the table (Figure 6.4).

6.2 Accessibility

Our proposed setup and TACTIC can come together to create new possibilities in the accessibility field as well. As such, we contributed to a study aiming at understanding how blind people interact with tabletops using a touch based exploration. This section presents a brief description of this study, focusing on our setup and API's contributions.

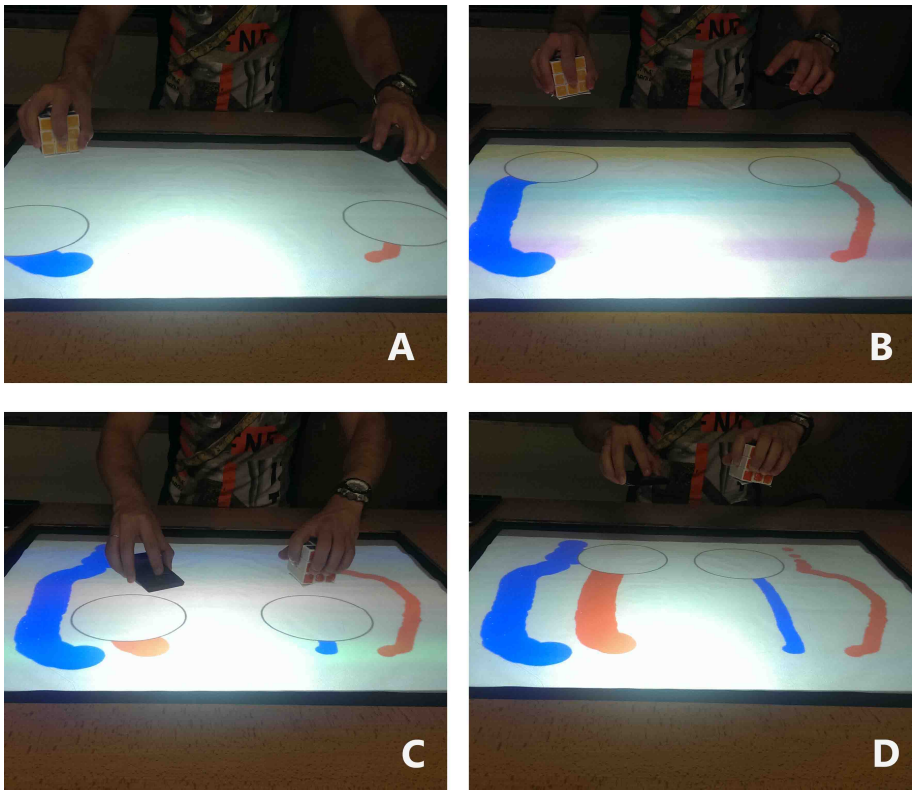

Figure 6.3: (A) Cube and Phone being tracked above; (B) Cube paints with blue, phone paints with red; (C) When changing hands, color is switched accordingly; (D) Brush size remains correct for each hand, big for right hand, small for left hand

6.2.1 Motivation

Interactions with tabletops and large surfaces is still a relatively infant domain, when looking at the accessibility solutions on offer for blind users. Smaller mobile and tablet counterparts are shipped with built-in accessibility features, enabling non-visual exploration of linearized screen content. It is unknown how well these solutions will perform in large tabletop environments, with more complex spatial content layouts. A study was conducted where 14 participants performed common tabletop interactions using Explore by touch, the common method of non-visual access to touchscreens; and SpatialTouch, our proposed interface to support simultaneous two-hand exploration.

6.2.2 Application design

This study takes advantage of our setup described in chapter 2 and TACTIC's ability to allow easy integration of gesture and touch information when interacting with the table surface.

When a user touches the table surface TACTIC is responsible for tracking which hand and finger are responsible for that touch at all times. Furthermore, the design of the experiment required an audio component to provide feedback through over-ear headphones

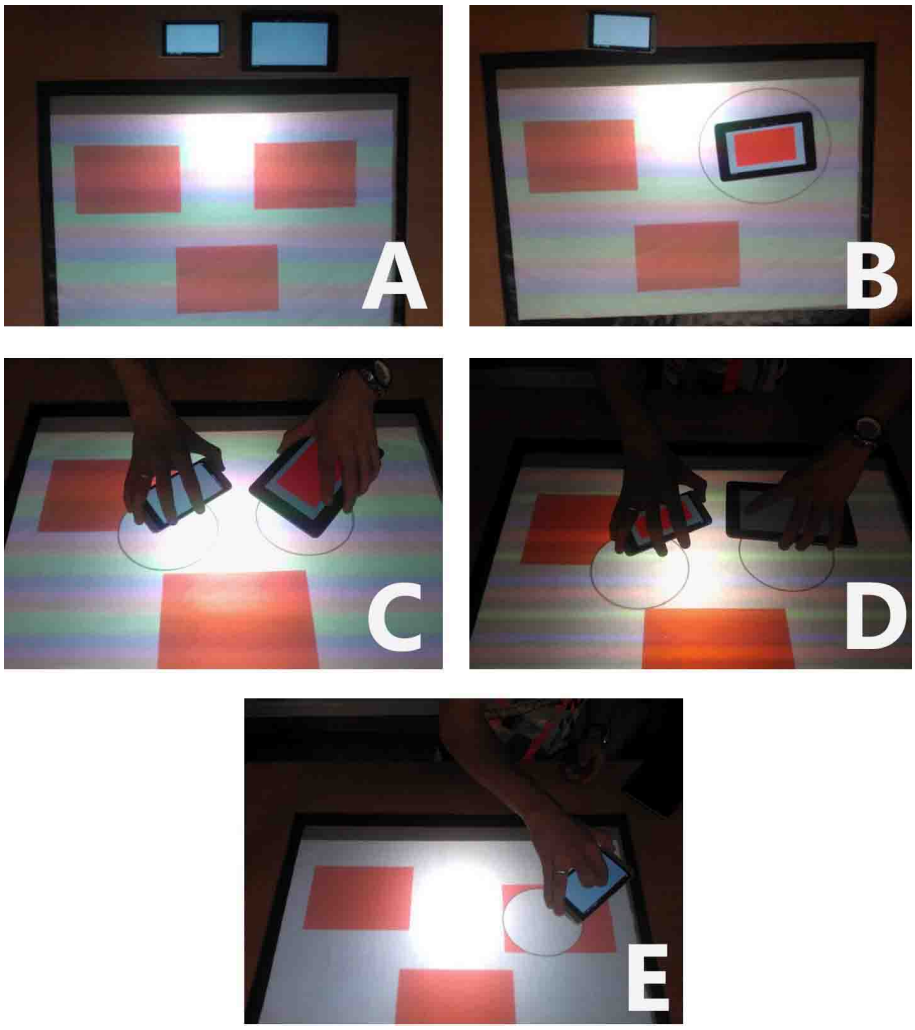

Figure 6.4: (A) Application with 3 elements; (B) Tablet is placed on top of element capturing it; (C) Tablet and phone being tracked above; (D) Tablet and Phone proximity caused the element to be sent from one to the other; (E) Phone being tracked above is touched, causing the object to return to the table below it

to users. Different voices were assigned to each hand, as well as different sounds for each touch element. This already existing component was easily connected to the touch application through RabbitMQ, resulting in no code being changed in the audio component. Additionally a logging keeper was also easily connected to the touch application to record touch points at all times, retaining *location (x, y), timestamp, touch-state (begin, move, end), hand and finger of interaction, target id*.

A stimulus application was developed to generate tabletop interfaces within a 9 x 12 grid (108 areas). Target grids were randomly generated for each new task. Figure 6.5 shows targets randomly distributed in the form of text.

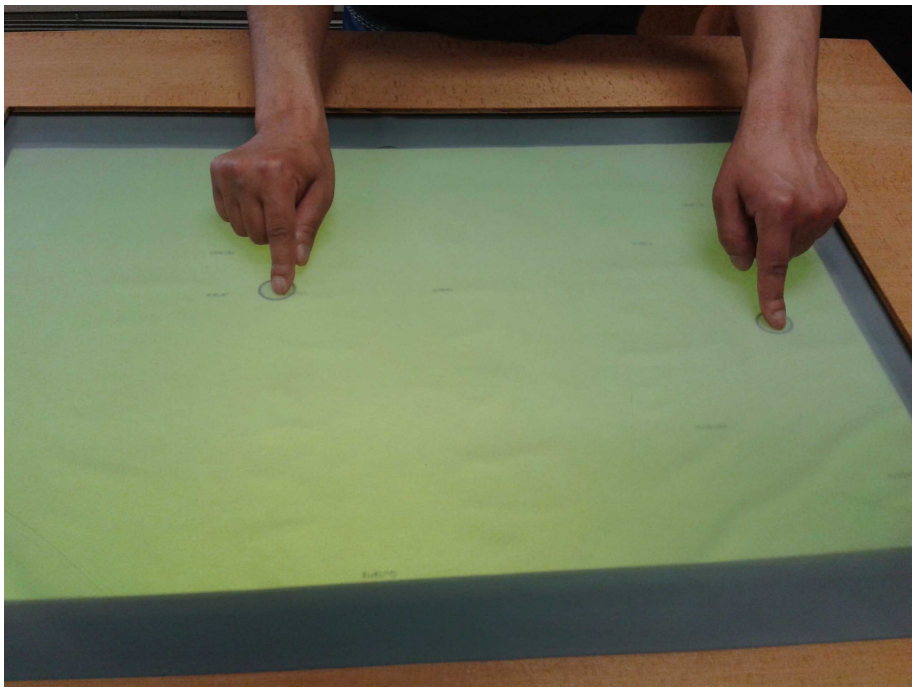

Figure 6.5: Participant using our setup with *SpatialTouch* exploration

6.2.3 Participants

Fourteen participants with visual impairments, eleven males and three females, took part in our user study. Participants' age ranged from 23 to 62 (M=44.5, SD=12.1) years old. They were recruited from a local social institution and all participants were legally registered blind. None of the participants reported having sever motor or hearing impairments (Figure 6.6).

6.2.4 Exploration methods

The following two interface methods were used to collect user interactions:

Explore By Touch. Participants could interact by dragging their finger around the screen, and the system would read aloud the name of the object underneath. A "click" sound was performed when a participant's finger exited an object. Targets could be selected by performing a double tap gesture anywhere on the surface, the last interacted with was then selected.

SpatialTouch. An extension of the basic functionality of *Explore by touch*, to provide support for simultaneous bimanual interactions. *SpatialTouch* leverages multiple sound sources, and off screen tracking to identify which hand corresponds to the touch *pointer id* captured by the touchscreen, and provides independent feedback for each hand. Interactions were restricted to just one finger per hand. To aid the distinction between multiple sound sources, each hand is mapped to a specific voice (male or female) and assigned

a specific location. We used the *Text-to-Speeches* [1] spatial audio framework to map the voices to either the left or the right ear, depending on the hand used to interact with the object.

6.2.5 Conclusions

Exploring the screen of a smartphone or a tablet is a common task for a great amount of blind people.

Large surfaces bring novel challenges to the exploration task. Not only the exploration surface is larger but the interfaces built for these (e.g., interactive kiosks or maps) are not as tidy as the ones in commodity devices.

This work looked at how blind people deal with large touch surfaces to locate, relocate, and relate targets, along with how well they are able to acquire a good spatial model of the screen contents. TACTIC was a major factor towards developing an application that was capable of capturing the necessary information to reach accurate results and take the appropriate conclusions.

Results revealed the users employ different strategies depending on their goal but that these coping mechanisms are still ineffective. We also investigated if using two hands to explore the screen improved their abilities to effectively do so. This showed to be beneficial in some tasks, particularly to relate targets, and to provide better structure in the exploration task.

6.3 User collaboration

Everyday interactions with physical tables are, generally, not restricted to one individual. It is common for collaborative scenarios to take place in everyday actions, like passing objects from one individual to another. We set out to expand on our existing setup to support these collaborative scenarios.

As described in chapter 2, there is a Microsoft Kinect camera above the surface, which is responsible for providing hand tracking information to the ThreeGear API (section 2.1.5). As described in (section 2.2.2) the ThreeGear API limits hand detection to one pair and to the direction the camera is facing. In order to solve this problem, we attached a second Kinect camera to the opposite side of the table (Figure 6.7) and connected it to a second computer running the ThreeGear API. This way each computer can handle one pair of hands located on the side of their respective cameras. Since ThreeGear maps hand shapes to create their virtual representations it is impossible to track any hand that is directly opposite to the direction the camera is facing, which means that the fidelity of hand tracking for a user on one side of the table is not compromised by the hands the

[1] bitbucket.org/sound/texttospeeches

other user is showing on the opposite side. Thanks to TACTIC's modular architecture, it is easy to add the new pair of hands as a new module. A RabbitMQ (section 4.2) cluster is created connecting computer A to computer B, merging both hand inputs into one single pool of information. Computer A is running a client application, which will be described later on, as well as the TACTIC API (chapter 4). TACTIC receives gesture information from RabbitMQ, which in turn fetches it from the ThreeGear component in this computer as well as computer B's ThreeGear component.

To prove this concept we built an application of top of the one proposed in section 6.1.4, which allows smartphones and tablets to capture elements from the surface and either return them again while on the surface, or while hovering above it. This application also allows devices to exchange their elements when near each other above the surface. This feature works, thanks to TACTIC's ability to monitor hand gestures above the surface, therefore, by adding the new hand information from computer B the application continues to work with almost no changes to the existing code. As seen in Figure 6.8, two objects are being tracked on opposite sides of the table. Each pair of hands are being tracked by separate kinect cameras, but being treated as one single pool of information in the application running on one of the computers.

Figure 6.9 shows how both phones exchange elements when nearby. This was possible without any changes to previous application's code. Finally Figure 6.10 shows the user who received the element droping it on the surface with a touch on the phone's screen.

With this experiment we were able to provide a collaborative scenario with ease thanks to TACTIC's modular architecture. Additionally it is possible to expand this scenario up to four users by simply adding two kinects to the remaining sides of the table coupled with two additional machines, since its easy to connect them to RabbitMQ's pool of information.

6.4 Discussion

In this chapter we have presented TACTIC's core features in different settings and fields. We set out to explore different interaction scenarios with a set of applications, as well as expanding our setup to allow more than one user to interact at the same time. A user study conducted on blind people proves TACTIC's relevance towards the field of accessibility by providing new forms of interaction.

Its important to state how easy it was to develop all of the applications presents in this chapter thanks to TACTIC. All of the applications presented in sections 6.1 and 6.3 took less than one hour to develop, since most of their interactions take advantage of TACTIC's already existing element properties and events that are easy to implement. In the case of the user study conducted in section 6.2, the used application was easy to connect to an existing Java audio component and a logging keeper thanks to RabbitMQ.

This is testament to TACTIC's impact on the creation of applications, doing all of the heavy lifting while providing abstractions and detailed events to developers.

Figure 6.6: Participant using our setup during a trial

Figure 6.7: Two kinect cameras facing opposite sides of the table

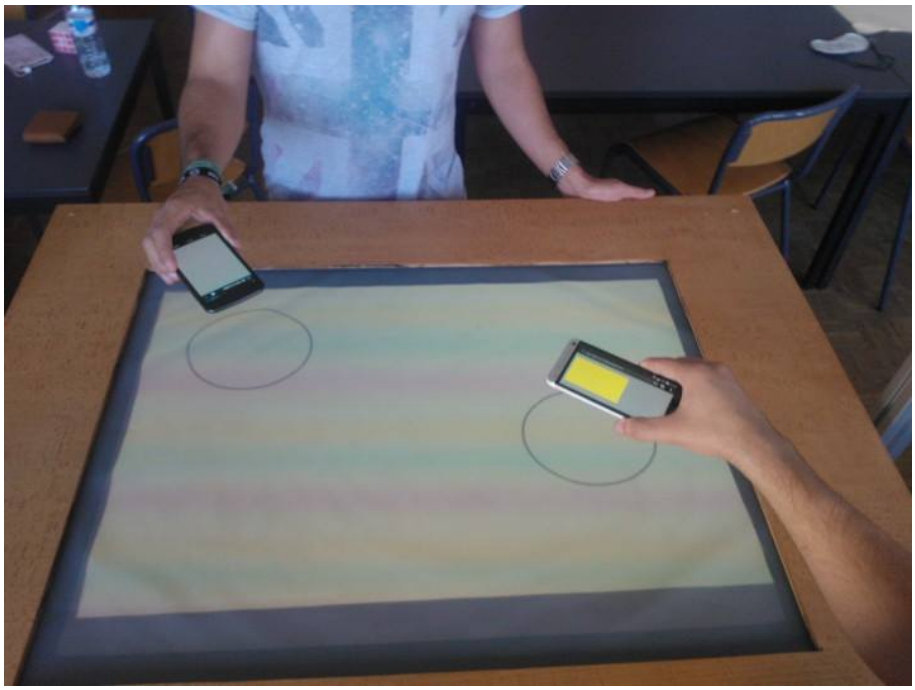

Figure 6.8: Two objects being tracked on opposite sides of the table

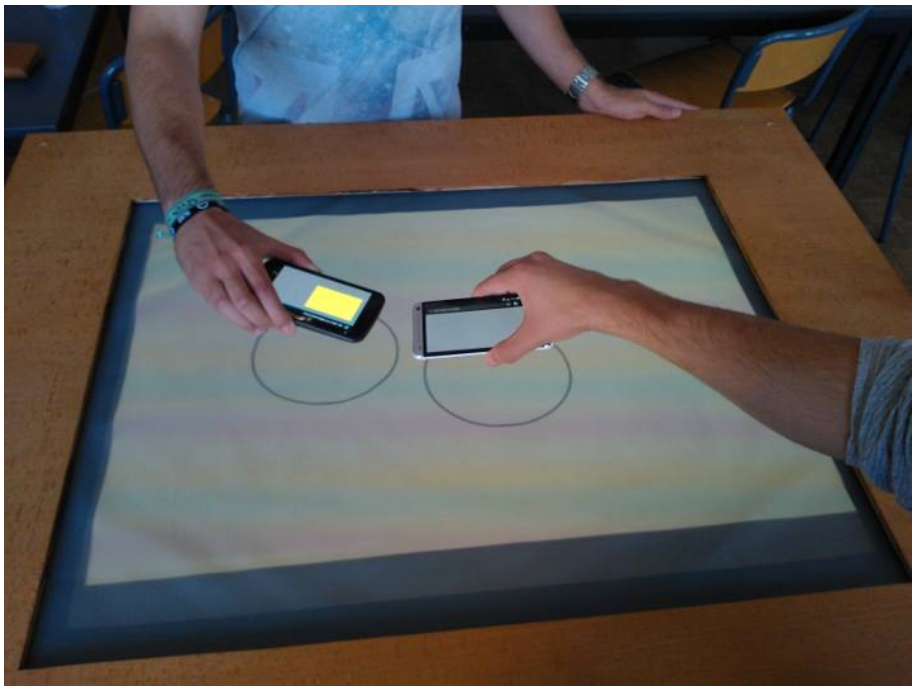

Figure 6.9: Element exchanging from one user to the other

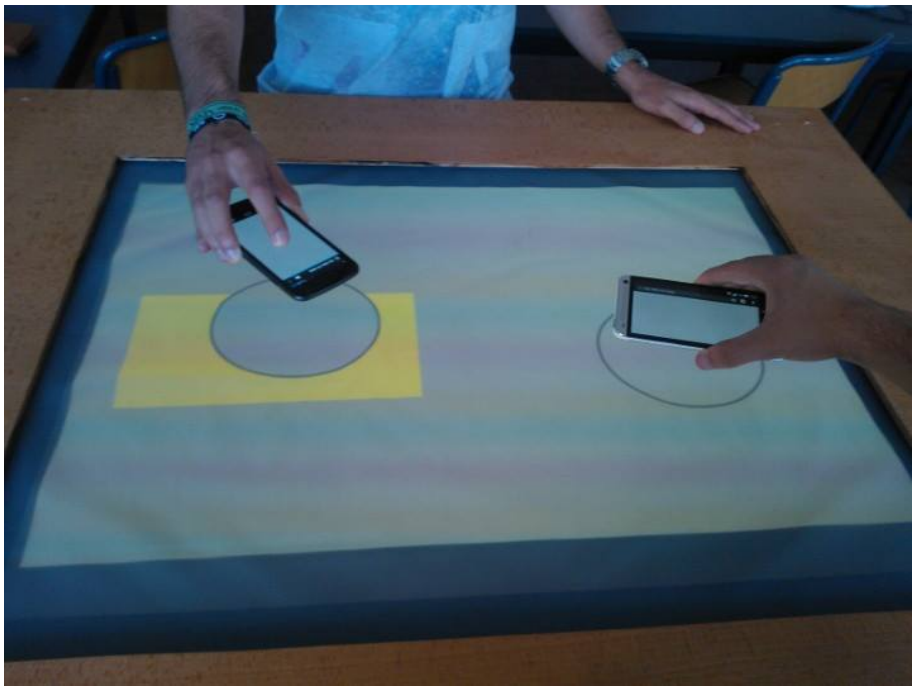

Figure 6.10: Opposite user's phone dropping element on the surface

Chapter 7

Conclusion

In this chapter we present our conclusions on this thesis and our thoughts on future work.

We set out to augment our existing multitouch setup to support tangible interactions on and above the surface. Although *on the surface* interactions were relatively easy to include in our system, the same can not be said for *above the surface* interactions, which were only made possible thanks to TACTIC, our developed API. TACTIC handles tangible interactions in the air by constantly monitoring hand tracking states and *on the surface* tangible interactions, which allows it to accurately predict which tangibles are being held, as well as tangible exchanges in mid air.

Our API expanded to include more functionalities with time, as we felt the need to draw more forms of interaction from our setup. Since tangibles can come in the form of smart objects, such as smartphones or tablets, our next step was to include a cross-device communication service to allow easy development of applications that communicate with the devices they are interacting with. We conducted a developer study for validation purposes, which showed that users found TACTIC to be easy to understand and use. Developers were able to build applications with touch and tangible interaction on and above the surface as well as distributed interfaces, in a very short period of time thanks to its promotion of code reuse.

All of TACTIC's features were showcased through a set of applications aiming to create new ways of interaction from our combination of technologies, ranging from simple touch interfaces to cross-device scenarios. Thanks to our API's modular architecture we were, again, able to expand our setup to support collaborative scenarios, which was coupled with an application showcasing how users can share elements around a multitouch table with smart objects above the surface.

We conducted a study on gesture performance on and above the surface for Pinch and Rotation gestures to complement existing characterizations of gestures both on tabletop surfaces and in mid-air. Furthermore, we had the opportunity to test our setup and API's contribution on the field of accessibility by conducting a user study on blind users' exploration methods on one and two handed scenarios.

For our future work, we wish to continue to grow our API to give more power to developers by providing more events and properties as well as improving our setup to support more collaborative scenarios with up to 4 users. We would also like to add more sources of information that are different from the ones we already have so that our API can support even more existing technologies.

Bibliography

[1] Frustrated total internal reflection. http://wiki.nuigroup.com/FTIR#Links:, Nov 2013.

[2] It goes where no device has gone before. https://www.leapmotion.com/product, Nov 2013.

[3] Rabbitmq. http://www.rabbitmq.com, Dec 2013.

[4] Threegear systems. http://www.threegear.com/technology.html, Nov 2013.

[5] A toolkit for tangible multi-touch surfaces. http://reactivision.sourceforge.net/, Nov 2013.

[6] Tuio 1.1 protocol specification. http://www.tuio.org/?specification, Nov 2013.

[7] Community core vision. http://www.nuigroup.com/, Sept 2014.

[8] Michelle Annett, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. Medusa: a proximity-aware multi-touch tabletop. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 337–346, New York, NY, USA, 2011. ACM.

[9] Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. Menuoptimizer: interactive optimization of menu systems. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, UIST '13, pages 331–342, New York, NY, USA, 2013. ACM.

[10] Paul Dietz and Darren Leigh. Diamondtouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 219–226, New York, NY, USA, 2001. ACM.

[11] Benjamin W. Franks, Lasse Schwarten, Jens Teichert, Markus Krause, and Marc Herrlich. User Detection for a Multi-touch Table via Proximity Sensors. In *IEEE Tabletops and Interactive Surfaces 2008*. IEEE Computer Society, 2008.

[12] Daniel Gallardo and Sergi Jordà. Slfiducials: 6dof markers for tabletop interaction. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces*, ITS '13, pages 401–404, New York, NY, USA, 2013. ACM.

[13] Otmar Hilliges, Shahram Izadi, Andrew D. Wilson, Steve Hodges, Armando Garcia-Mendoza, and Andreas Butz. Interactions in the air: adding further depth to interactive tabletops. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 139–148, New York, NY, USA, 2009. ACM.

[14] Eve Hoggan, Miguel Nacenta, Per Ola Kristensson, John Williamson, Antti Oulasvirta, and Anu Lehtiö. Multi-touch pinch gestures: Performance and ergonomics. In *Proceedings of ITS'13*, pages 219–222. ACM, 2013.

[15] Eve Hoggan, John Williamson, Antti Oulasvirta, Miguel Nacenta, Per Ola Kristensson, and Anu Lehtiö. Multi-touch rotation gestures: Performance and ergonomics. In *Proceedings of CHI'13*, pages 3047–3050. ACM, 2013.

[16] Christian Holz and Patrick Baudisch. Fiberio: a touchscreen that senses fingerprints. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, UIST '13, pages 41–50, New York, NY, USA, 2013. ACM.

[17] Shahram Izadi, Steve Hodges, Stuart Taylor, Dan Rosenfeld, Nicolas Villar, Alex Butler, and Jonathan Westhues. Going beyond the display: a surface technology with an electronically switchable diffuser. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 269–278, New York, NY, USA, 2008. ACM.

[18] Daniel Jackson, Tom Bartindale, and Patrick Olivier. Fiberboard: compact multi-touch display using channeled light. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 25–28, New York, NY, USA, 2009. ACM.

[19] S. Jordà. The reactable: tangible and tabletop music performance. In *Proc. CHI EA '10*, pages 2989–2994. ACM Press, 2010.

[20] Liu Jun, D. Pinelle, C. Gutwin, and S. Subramanian. Improving digital handoff in shared tabletop workspaces. In *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems (TABLETOP 2008)*, pages 9–16, 2008.

[21] Martin Kaltenbrunner and Ross Bencina. reactivision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, TEI '07, pages 69–74, New York, NY, USA, 2007. ACM.

[22] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza. TUIO: A protocol for table-top tangible user interfaces. In *Proc. of the The 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005.

[23] D. et al Ledo. The haptictouch toolkit: enabling exploration of haptic interactions. In *Proc. TEI 2012*, pages 115–122. ACM Press, 2012.

[24] Weibel N. Liu, Y. and Holland J. Interactive space: a prototyping framework for touch and gesture on and above the desktop. In *Proc. CHI EA '13*, pages 1233–1238. ACM Press, 2013.

[25] Nicolai Marquardt, Ricardo Jota, Saul Greenberg, and Joaquim A. Jorge. The continuous interaction space: interaction techniques unifying touch and gesture on and above a digital surface. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part III*, INTERACT'11, pages 461–476, Berlin, Heidelberg, 2011. Springer-Verlag.

[26] Nicolai Marquardt, Miguel A. Nacenta, James E. Young, Sheelagh Carpendale, Saul Greenberg, and Ehud Sharlin. The haptic tabletop puck: Tactile feedback for interactive tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 85–92, New York, NY, USA, 2009. ACM.

[27] E. LingNau A. McCrindle C., Hornecker and J. Rick. The design of t-vote: a tangible tabletop application supporting children's decision making. In *Proc. IDC '11*, pages 181–184. ACM Press, 2011.

[28] Rafael Nunes, Nikolay Stanchenko, and Carlos Duarte. Combining multi-touch surfaces and tangible interaction towards a continuous interaction space. In *Proceedings of the 3rd Workshop on Interacting with Smart Objects*, page 11–14, 2014.

[29] Hamilton P. and Wigdor D. Conductor: enabling and understanding cross-device interaction. In *Proc. CHI '14*, pages 2773–2782. ACM Press, 2014.

[30] Sangbong Park. Implementation of new gestures on the multi-touch table. In *ICT Convergence (ICTC), 2012 International Conference on*, pages 168–169, 2012.

[31] Dominik Schmidt, Fadi Chehimi, Enrico Rukzio, and Hans Gellersen. Phonetouch: a technique for direct phone interaction on surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 13–16, New York, NY, USA, 2010. ACM.

[32] Dominik Schmidt, Ming Ki Chong, and Hans Gellersen. Handsdown: hand-contour-based user identification for interactive surfaces. In *Proceedings of the 6th Nordic*

Conference on Human-Computer Interaction: Extending Boundaries, NordiCHI '10, pages 432–441, New York, NY, USA, 2010. ACM.

[33] Seifert J. Rukzio E. Schmidt, D. and H. Gellersen. A cross-device interaction style for mobiles and surfaces. In *Proc. DIS '12*, pages 318–327. ACM Press, 2012.

[34] Johannes Schöning, Jonathan Hook, Tom Bartindale, Dominik Schmidt, Patrick Oliver, Florian Echtler, Nima Motamedi, Peter Brandl, and Ulrich Zadow. Building interactive multi-touch surfaces. *Tabletops-Horizontal Interactive Displays*, pages 27–49, 2010.

[35] Diane Watson, Mark Hancock, Regan L. Mandryk, and Max Birk. Deconstructing the touch experience. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces*, ITS '13, pages 199–208, New York, NY, USA, 2013. ACM.

[36] Andrew D. Wilson and Hrvoje Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 273–282, New York, NY, USA, 2010. ACM.

[37] Andrew D. Wilson and Raman Sarin. Bluetable: connecting wireless mobile devices on interactive surfaces using vision-based handshaking. In *Proceedings of Graphics Interface 2007*, GI '07, pages 119–125, New York, NY, USA, 2007. ACM.

[38] J. Yang and Wigdor D. Panelrama: enabling easy specification of cross-device web applications. In *Proc. CHI '14*, pages 2783–2792. ACM Press, 2014.