Comparative studies of different input and control methods in a real time tabletop interaction context

A comparative analysis of Tangible User Interfaces and Multi-Touch within the context of a real-time tabletop video game

Giovanni Maria Troiano

UPF MASTER THESIS / 2011

Thesis Supervisor

Carles F. Julià (Music Technology Group -MTG)

Thesis Coordinator

Dr.Sergi Jordà Puig (Music Technology Group -MTG)

UNIVERSITAT POMPEU FABRA

Acknowledgements

I wish to thank my supervisors Carles F. Julià and my coordinator Dr.Sergi Jordà Puig from the MTG lab for their precious advices, their constant support, generous assistance and continuous encouragement. I'm also very grateful to Ulysses Bernardet from the SPECS lab for his thoughtful suggestions and his fundamental reflections and insights that helped me to find right solutions; to Jitesh Joshi who assisted me while programming the application used for my experiment; to Sebastian Mealla C. from the MTG lab and Juan Gabriel Tirado for their precious help and advices.

Finally I would like to thank all my classmates of the Interdisciplinary Master in Cognitive Systems and Interactive Media, for having shared with me a wonderful experience here in Barcelona.

Abstract

This report presents the results of an exploratory study of a tangible and a multi-touch input control methods in a real-time tabletop interaction context. The study investigated the effect of the interface on user's performances of *accuracy-precision* and *precision-rapidity* in target deflection and target interception tasks, performed in the context of a real-time tabletop video game. Participants played with two different "video game alike" applications using each input method on a classical ReacTable set up. The effect of both controls was investigated through quantitative performance measurement of the gameplay, as well as a qualitative analysis of participant's responses on post-test questionnaire and interview, regarding usability and user experience. Main findings revealed that tangible control better enabled for performance in both tasks, and that it generally created a sense of major control and comfortability with respect to multi-touch input method. The drawbacks of multi-touch were partially due to some issues related to the tracking technology used, some inherent limits of the application designed for the experiment and the uncomfortability created by the friction of the fingers on the surface when rapid sliding movements were performed. However, the qualitative analysis showed positive sensations of the users in favour of multi-touch control, in which it was perceived as a very intuitive and accessible control method. This result is partially in contrast with the findings of Lucchi et al. in [86], showing that, if less complicated actions are required in manipulation with the use of multi-touch control, the trend is to perceive it as more approachable and easy to acquire. The study contributes to understanding the implications of using of these two input control methods in a real-time interaction context, which is a domain that have not received great attention from the scientific community for this comparative studies.

Keywords: Input Devices, TUI, Tangible, Interactive Surface, Multi-Touch, Reactable, Real-Time, Interaction, Video Game, Tabletops, Control, Usability.

Table of Contents

1. INTRODUCTION

1.1 Problem Statement

Multi-touch, touch-screen and in general input control methods that encompass direct manipulation or multiple fingers interaction, are emerging as good high-degrees of freedom inputs. Despite presenting some specific problems with precise gestures [114], due to their ease of use and their intuitive affordances they are indeed among the most implemented controls nowadays, especially on portable devices, such as touch-sensitive tablets, touch-screen mobile technology, leptops and similar. For this reason, many researchers and developers in the field of HCI have tried to understand multi-touch and touch-screen benefits in terms of usability, effectiveness and efficiency, in order to improve its implementation [6, 10, 23, 22, 98, 120]. While bimanual interaction strategies with multi-touch and hybrid pen/multi-touch has been widely studied [6, 52, 53], the implications of using two or more fingers with the same hands have not been completely understood yet. Furthermore, outside the domain of mobile technology, when the multi-touch input is implemented in tabletops for the control and the manipulation of GUIs (Graphical User Interfaces) and virtual elements, his evaluation often crosses with the more robust TUI (Tangible User Interface). Due to their similar properties, interaction purposes and applications, this two control methods invite us to question whether they can be more suitable for some types of gesture and to what extent they can be complementary interacting in the same context. In cases like the ReacTable[1] [72] or Facet-Streams[2] [69] for instance, TUI and multi-touch have been smartly implemented together in the same interactive set up, in order to let the users take advantage of the specific characteristics of both control methods, to appropriately accomplish different types of actions and gestures in relation to their tasks. This examples clearly shows how multi-touch and tangible controls can be suitable for same devices or set ups, but at the same time they optimize user's experience when addressed to different type of uses.

One of the real challenges that has been undertaken by researches in the last few years, was to understand the natural placement and the gestural possibilities of these two control methods in relation with their physical affordances, adaptability, ease of use, intuitiveness, degrees of freedom, and relative limitations. Previous studies that have compared the use of tangibles and multi-touch [86, 114], produced results that confirm how tangible controls are still more robust and reliable with respect to the multi-touch when precise gestures are required. Recent studies on tangible and multi-touch interaction, suggest this comparative analysis has to be done away from the rhetoric of which control might be better and which worst [69]. However, while many experiments have been conducted on the use of tangible and multi-touch in non RTI (Real Time Interaction) context, so far, the community have not paid much attention in studying and investigating the use of this two control methods in a RTI context. Since the analysis of different input devices in HCI often made use of tasks involving precise physical actions and gestures [26, 65, 67, 89], such as pointing, dragging, acquiring target,

[1] http://www.reactable.com/
[2] http://www.youtube.com/watch?v=giDF9lKhCLc

intercepting moving targets, etc., Fitts's law [35] has generally shown to be a good predictor model on which shaping and evaluating user and input device performances. However, for a RTI context, such as a video game for instance, where user's action are generally less constrained to tapping back and forth with a stylus pen, or pointing with a laser from point A to point B, the pool of gestures that can be performed through the use of tangible and multi-touch controls may be too complex to be measured only with Fitts's law, therefore alternative solutions and strategies in evaluating input devices and user performances in this interaction context must be proposed and experimented.

1.2 Comparative analysis of different input control methods in HCI

1.2.1 Input devices classification and taxonomies

Several studies and researches have been conducted on different typologies, characteristics, use and implementation of input control methods in HCI and multimedia application [13, 38, 91]. Due to the very heterogeneous nature of the different tasks a system or a workstation can provide the user with, finding a universal control that can allow the accomplishment of any task in the appropriate way, would be practically impossible. According to Buxton, the choice of the right input technology to be used, whose qualities can better optimize user's performances, must be a "*trade-off"* between the physical characteristics of the input control method and the type of tasks that one has to perform [20]. Hence, a study about systematic knowledge and taxonomy of input control methods and technologies, helps out detecting the right position for this "*trade-off"*. Foley et al. [38] proposed a taxonomy of input technologies based on their relative graphical tasks (see Figure 1), but since single devices appear many times in the leaves of their trees, this taxonomy does not help a systematic detection of similarities and differences among input devices.

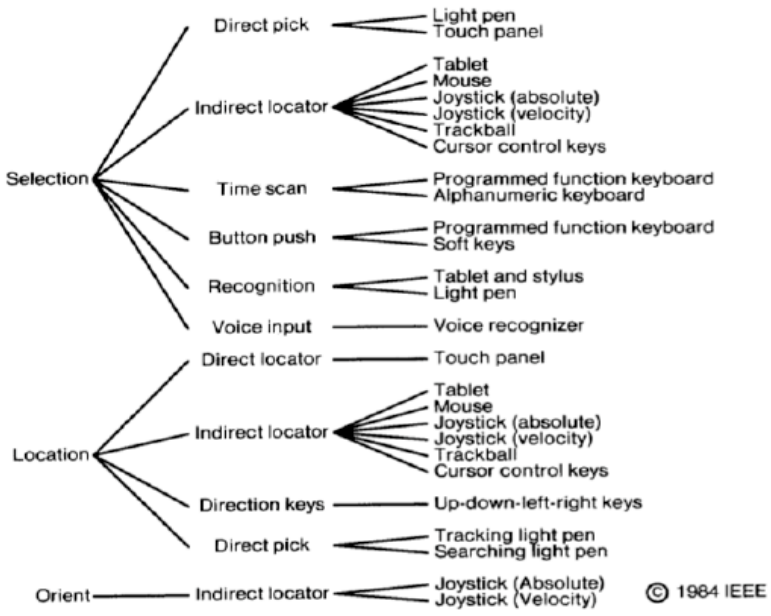

Figure 1: Foley et al. trees, on the left side the main three group representing some computer graphic input, in the center the action to be performed, on the right side the input devices mapped into this actions

The same attempt to categorize input technologies was made by Buxton in [13], but instead of starting from the action to be performed and then detecting the right input technology to use, he started from the physical characteristics and spatial dimension of the controls, trying to assess a basic design for input devices independently form graphical tasks (see Figure 2). However, this taxonomy takes into account only continuous physical devices and does not discuss the possibility of combining individual inputs into more complex types of controls.

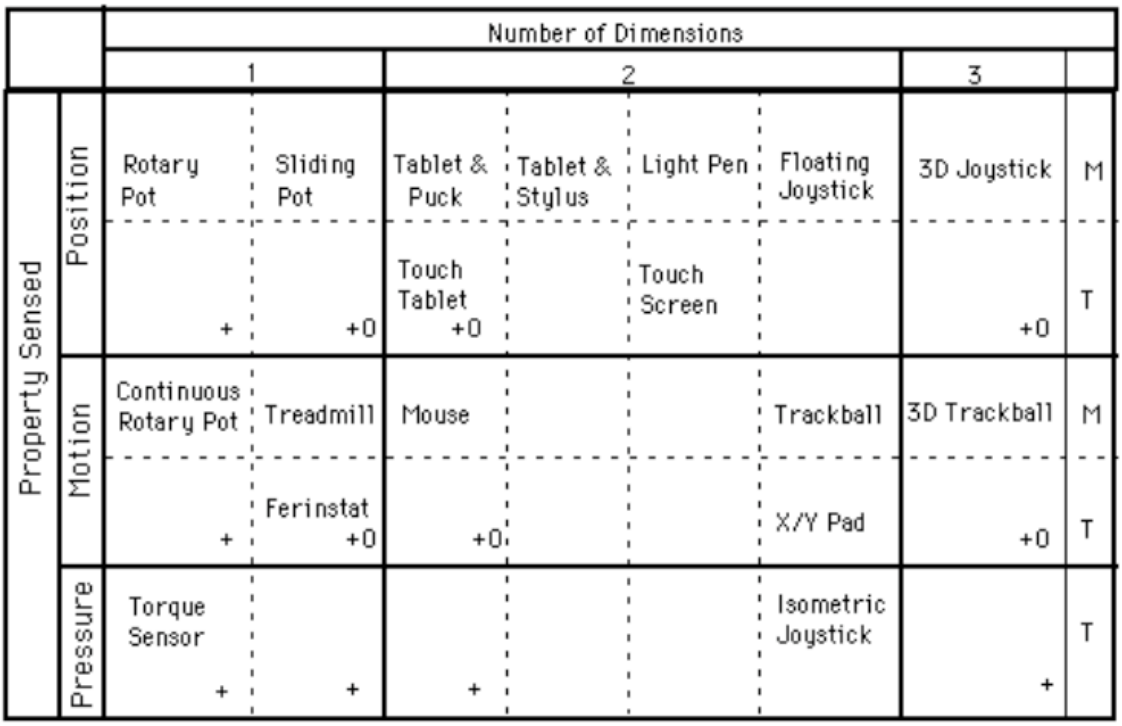

Figure 2: Bill Buxton's input taxonomy (1983)

A step forward was made by Card et al. in [91], where they reviewed Foley's and Buxton's taxonomies by reporting these models in their new paradigm (see Figure 3), as well as proposing their own approach. It is important to say that their study about input devices focused especially on the *expressiveness* of the input technologies, i.e. the ways in which their physical manipulation can communicate meaning to an application. Therefore, their work was mainly focused on classifying the devices according to three fundamental criteria: 1) the physical properties that an input device transduces, 2) how many DOF (Degrees Of Freedom) it can sense, 3) the measure of the input domain set (from discrete to continuous, i.e. from 1 to infinite). In this way, they were able to compensate the lack of Buxton's taxonomy, taking into account discrete inputs as well as continuous ones.

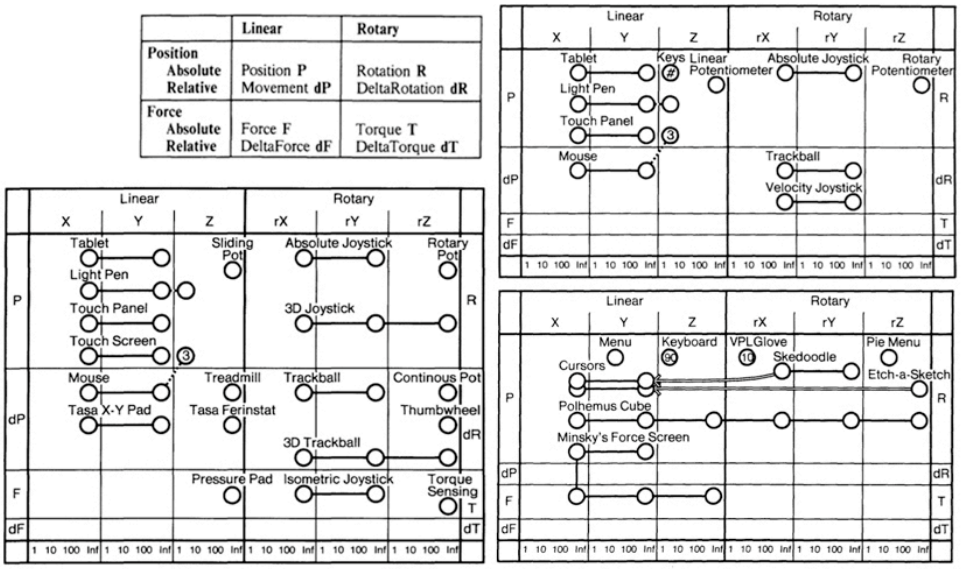

Figure 3: Various example of taxonomies by Card et al. (1990)

Graham et al. in [41] have introduced a model called *Dimension Space* (see Figure 4), on which categorize input devices through a new design approach, which transcends the principal that user interfaces and digital systems are two distinct parts of the interaction process, but they are rather combined into complete systems through the richly interaction of physical and virtual *"entities"*. In their explanation, richly interactive applications must be designed as *"complete systems involving mutual interaction of users, software and physical entities"* (Graham et al., 2000); therefore the description of an input device is not limited to the dimensions it can sense, how many DOF it has and what are its physical properties. Here an input device is described with respect to the space in which it is implemented and used, where different cognitive and physical qualities converge. Therefore, the understanding and evaluation of the control itself cannot be possible just from a technico-physical point of view, but it is rather extending to the degrees in which it is integrated in the system, the way it acts with respect to the system and how it is perceived from the user within the interactive experience.

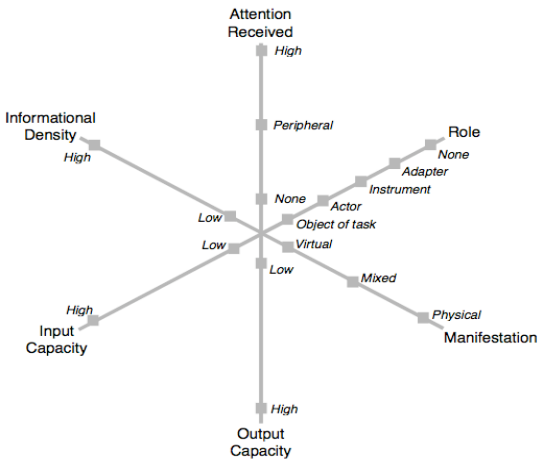

Figure 4: Graham et al. the Dimension Space, the plot is describing various characteristics of the "entities" (2000)

1.2.2 Implementation of controls

As described before, different input taxonomies have been proposed that tried to explain and optimize the way users can manage input controls, some of them more input device centered [33], other more focused on gestures and interaction [19, 53, 76, 119]. However, improving the quality of interaction *"tout court"* between users and systems is generally the final aim of these studies. In order to achieve refined control techniques and providing users with more possible ways to accomplish different tasks, it is therefore essential to know how to appropriately combining input's physical characteristics with different types of feedback (audio, video, tactile, etc.). As explained in related works [51], to rightly implement an input control method, designers have to take into account physical sensors, feedback presented to the user, ergonomic aspects, interaction techniques supported by a system, but also the cognitive modes in which humans are perceiving the interaction. Hunt and Kirk in [60] have detected two distinct modes of operation or thinking that human are using in controlling the elements of an interactive system, i.e. *analytic* and *holistic*. The former is regarded as essentially sequential and logically ordered, giving much importance to the single information and details perceived, whereas the latter gives more importance to the overall effect of the interaction. For instance, when interacting with a desktop or a laptop computer, users are generally provided with a set of multiple options, through which they perform navigation and single sequential choices using one specific device (i.e. mouse or keyboard); therefore this can be regarded as an *analytic* mode of interaction. On the contrary, when the system supports the use of more controls at the same time to modify different parameters, allowing the user to manage multiple contemporary choices, the interaction would be based on a *holistic* mode and perceived as a continuum rather than sequential. Due to the largely predominant WIMP (Window, Icon, Menu, Pointing) paradigm set by desktop interaction of PC (Personal Computer), it is in Hunt and Kirk opinion that nowadays *"the interaction of humans with computers and computer interfaces are unhealthily dominated by 'analytical' interaction"* (Hunt and Kirk, 2000). However, this distinction helps out understanding an important issue in control implementation and design, opened up by Buxton in [14], related to whether input devices can supply for *absolute* or *relative* values, and how the choice between those characteristics define the dimension in which the control method moves in, i.e. if it is *time* or *spatially multiplexed* (see Figure 5).

Figure 5: A classical example of a time multiplexed control (mouse) and a spatial multiplexed one (mixer). While with the former many parameters are modified sequentially in time with the use of one device, with the latter many parameters are modified at the same time through the use of many dedicated transducers for each parameter

"The example comes from process controls. There are (at least) two philosophies of design that can be followed in such applications. In the first, space multiplexing, there is a dedicated physical transducer for every parameter that is to be controlled. In the second, time multiplexing, there are fewer transducers than parameters. Such systems are designed so that a single device can be used to control different parameters at different stages of an operation" (Buxton, 1986). In other words, if the dimension of the input control method is *time-multiplexed*, we would use one control to do different sequential actions in time (as in the case of WIMP), therefore the system would be in charge of the dialogue and we would be using an analytical way of thinking in the interaction. In contrast, if the dimension of the input control method is *space-multiplexed*, we would use different controls to modify many parameters simultaneously; this would allow for a dialogue human-computer that has no fixed ordering, where the single modifications and processes would be perceived as a whole and continuous, leading us to the use of a holistic way of thinking. These dimensional, physical and cognitive factors are indeed affecting input control's implementation strategies and the way they are mapped[3] with respect to the logical interface. For instance, it is extensively explained in [36] how the use of multiple tangible controls[4] is more effective at spatial layout tasks, and how the implementation of these type of controls is correcting an inherent dissonance of traditional GUIs, where *"the display is often space-multiplexed while the input is time-multiplexed"* (Fitzmaurice et al., 1995). Hence, if we were to interact with a classical WIMP using multiple tangible controls, we could have argued that the way the input devices were implemented with respect to the system and mapped with the logical interface, would have allowed us for a more consistent and rapid use of the interface. However, it is not always the case that it is just enough modifying the mapping or implementing controls in a different way to improve the interaction between system and user and its performance. For this reason a comparative analysis of different input control methods, helps designer and researcher in understanding whether the potential improvement can rely on implementation, mapping, perceptual structure of tasks, ergonomic aspects, characteristics of the controls or physical actions and movements.

1.2.3 Evaluation methods and comparative analysis

The research and experimentation in the evaluation of existing input devices, as well as on the design of new ones, have seen the production of several studies and works in the literature of HCI. Most of these works are focused on defining representative tasks [16, 21, 26, 106, 125], analytical models for aimed movements [1, 2, 46, 88] and comparative analysis and evaluation of different devices [26, 63, 124, 127]. In [16] Buxton proposed a set tasks based on generic transactions performed by users while interacting with classical GUIs, i.e. pursuit tracking, target acquisition, freehand inking, tracing and digitizing, constrained circular and linear motion (see Figure 6).

[3] Mapping is generally defined as the operation of transforming some input data into other type of output data by virtue of specific synthesis ad expressions, the so called "I/O mapping". It is frequently used in different domains; in the case of input devices, mapping studies the logical connection between gestural parameters and output parameters (e.g. sounds, images, etc.)

[4] In this article Fitzmaurice, Ishii and Buxton have introduced the concept of Graspable User Interfaces, which will be later renamed by Ullmer and Ishii in [50] as TUI, i.e. Tangible User Interfaces.

Figure 6: Some representative tasks proposed by Bill Buxton. On the left side target acquiring, on the center freehand inking and on the right constrained circular motion

This set is based on the idiosyncratic character of different types of gestures and actions, and it's mainly focused on 2D graphical interaction. The creation of tasks that measure user and device performances, implies the quantification of these data to further validate evaluation models and methods. One of the most extensively used methodology in 1D and 2D task's quantification and evaluation is the Fitts's Law [35], which was originally proposed by Paul Fitts for measuring the information capacity of the human motor systems with the following formula:

$$MT = a + b \log_2 \left(\frac{A}{W} + 1 \right)$$

The original experiment entailed rapid back and forth displacement of a stylus pen between two targets of different sizes and placed at different distances (see Figure 7), subjects were instructed to perform this movement as quickly as possible. Fitts's Law formula has been widely reformulated for different experiments [32, 42, 44, 45, 64, 90, 97, 128], and the model showed a good fit for many other types of task, such as head movement [66, 107], distal pointing [79], dependence of target capturing time on information load and movement precision [126], MI (Motor Imagery) ability test [25], moving target interception and reaction time testing [68], pointing and dragging times in GUIs manipulation [87].

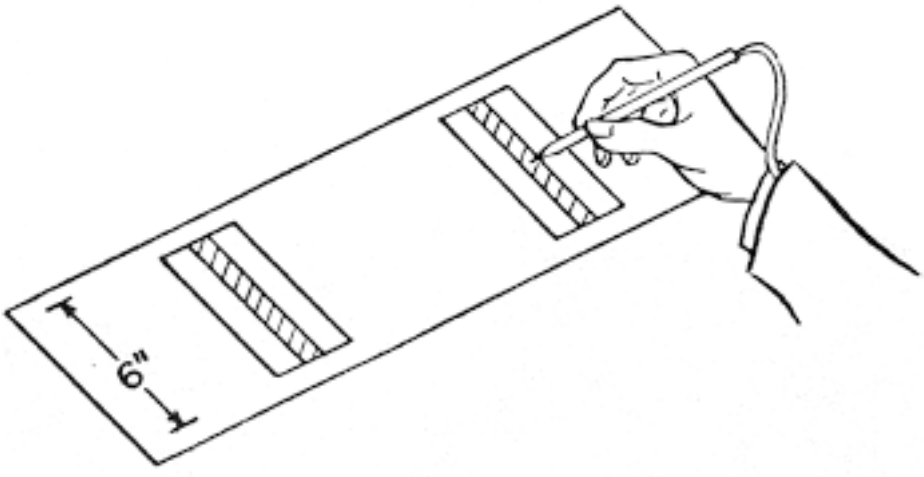

Figure 7: The stylus-tapping paradigm in which Fitts' law was first demonstrated for rapid spatially constrained movements (1954)

In comparative analysis of different input control methods, one of the most important experiments is indeed [26]. Card et al. have been using Fitts's Law in this experiment, to test and model performances of different controls, i.e. a mouse, an isometric joystick, step keys and text keys; experimental task entailed text selection on a CRT (Cathode Ray Tube) monitor, where the mouse was found to be the fastest device amongst the others and with the lower error rate. Despite being widely used for it's general applicability, the Fitts's law could not always model user performance in a successful way. It was the case of [3] for instance, where trajectory-based interaction needed the formulation of a more robust model to test actions like menus navigation, drawing curves and moving in 3D worlds, i.e. the "steering law[5]" (see Figure 8).

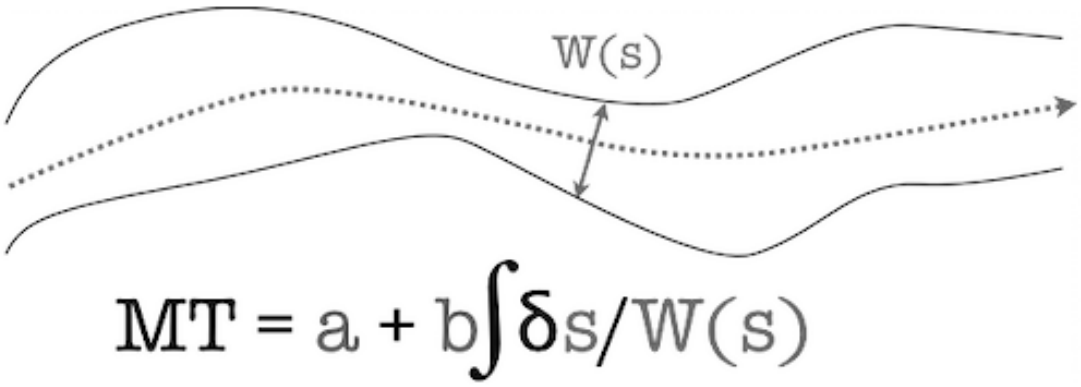

Figure 8: The Steering Law as formulated by Accot and Zhai (1997)

Other comparative analysis of input technologies have used methods other than pragmatic analysis of aimed movements, as in [63], where Jacob et al. showed that performance with integral and separable controls was superior in the condition where the device matched the task in integrality or separability. This experiment was mainly focused on proving how the perceptual structure of the task matches the control structure of the device; their conclusion suggested that a proper design of input technologies cannot be abstract from this principle: *"To design and select more effective input devices and interaction techniques, one needs to use a deeper understanding of task, device, and the interrelationship between task and device from the perspective of the user"* (Jacob et al., 1990). However, with the development of always more heterogeneous type of embodied technologies and devices, new and more specific researches questions are arising in the field of HCI, and evaluation methods have evolved too. Nowadays direct touch control such as multi-touch and tangible interaction are of great interest, and comparative studies are applying to this domains as well. Since multi-touch and tangible technologies are often implemented for tabletop interaction, studying these input methods in this domain may lead to interesting future outcomes.

[5] This model is so called because the experimental paradigm used to represent these type of trajectory-based tasks was the action of "steering through tunnels".

1.3 Tabletop interaction and input control methods

1.3.1 Historical account: early prototypes

With the rise of personal computers since 1980, real physical desktop operations performed on a horizontal tabletop, were turned into a relative metaphor for GUI on computer monitors (e.g Xerox Star, Macintosh 128K), thus switching the user's orientation from horizontally in front of a desk to vertically in front of a screen. This transition was encouraged by the new amount of possibilities that devices such Digital Assistants (PDAs) and new portable computers could provide to the user, i.e. high velocity of calculation, easy and quick digital data manipulation, appealing graphical interfaces, dynamic digital data storage, and so forth. Although fascinating and innovative, the so called WIMP paradigm have never really overcome the operational efficiency of a (using a Heideggerian term) "ready-to-hand" action on a tabletop or a desk; especially when having the need of performing spatial multiplexed actions (see point **1.2.2**), mouse and keyboards could not really substitute the far broader degree of control and gestural possibilities provided by human hands. In the early nineties, Pierre Wellner understood this problem as well as he realized that the passage toward the digital was indeed a crucial and an important point in evolutionary terms for human interaction. What he proposed in [122] with his DigitalDesk (see Figure 9), was a solution that tried to integrate both human natural actions and digital/virtual interface, trying to *"make the real desk like the workstation, instead of making the workstation like the real desk"* (Wellner, 1993).

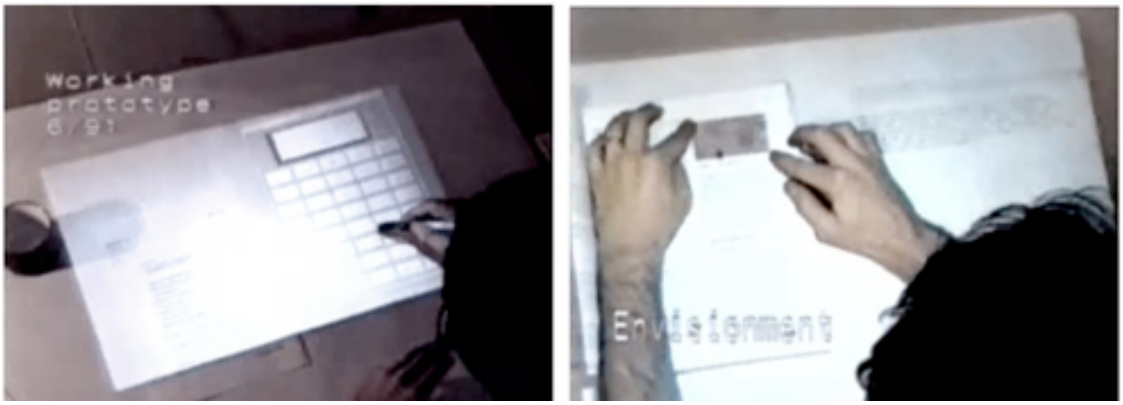

Figure 9: Pierre Wellner interacting with his DigitalDesk (1993)

This historical passage opened up to a new paradigm shift towards a more embodied digital interaction, taking into account the human gestural pool of bodily actions as a fundamental aspect, trying to improve the design of digital systems with which users interact with and manipulate data. The benefits of an approach towards a direct and embodied interaction with digital interfaces was discussed and evidenced prior to Wellner's contribution by Brown et al. in [10]. Here they wanted to exemplify how the logic working for partitions a single physical display into a number of different virtual displays, could be also useful when applied to input control; as stated in the introduction of their study *"a significant trend in user interface design is away from the discrete, serial nature of what we might call a digital approach, towards the continuous, spatial properties of an analogue approach"* (Brown et al., 1990). Ever since, investigation in digital desks and tabletop interaction has increased rapidly, especially in

view of the strong and advantaging features shown in the aforementioned studies, e.g. eliciting collaborative work and favoring spatial multiplexed control. It is in this direction that goes the ActiveDesk[6] created by Buxton and colleagues at Xerox PARC in the nineties. This tabletop was intended to be modeled as a traditional drafting table to support drawing activities, with which manipulating digital content retro-projected[7] on a transparent surface (see Figure 10), it was also used by Buxton, Fitzmaurice and Ishii for their study on Graspable User Interfaces in 1995 [36]. This study will be discussed more in details in point **1.4.1**.

Figure 10: Bill Buxton interacting with the Active Desk, University of Toronto (1992)

We can consider the aforementioned works of Wellner and Buxton as the pioneers of the field, and their research has been surely of fundamental importance for the experimental evolution that tabletops, multi-touch and tangible interaction had in the future. The metaDesk[8] introduced in 1997 bit MIT medialab researchers Ullmer and Ishii in [117] (see Figure 11) it's also another important piece of work in tabletop interaction, that extended the principals laid in the works of Buxton and Wellner, developing on top of them a new metaphor of interaction for a tabletop, through the use of physical tokens called Tangible User Interfaces (TUI). With the implementation of this physical tokens as a control to manipulate the virtual elements projected on the tabletop, Ullmer et al. were attempting to change the "painted bits" of GUI into the "tangible bits"of a TUI, by taking advantage of multiple senses and the multi modality of human interaction with the real world [61].

[6] http://www.billbuxton.com/ActiveDesk.html
[7] Compared with the top-projection of the DigitalDesk, the rear-projection of the ActiveDesk is favouring the sight of the user, since the physical presence of the user's body doesn't produce any shadow on the surface, therefore not interfering with the digital elements displayed by the projection.
[8] http://tangible.media.mit.edu/project.php?recid=81

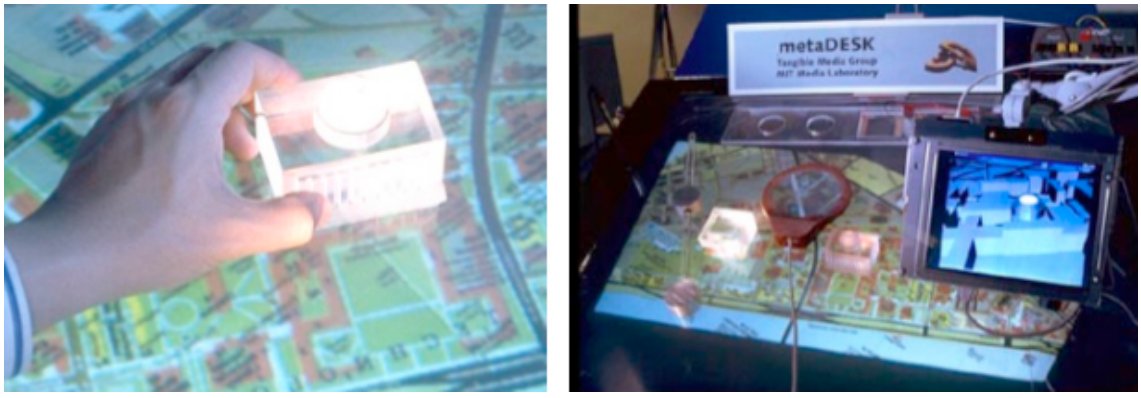

Figure 11: The metaDESK overview, on the right picture the Great Dome phicon (1997)

1.3.2 From BUILD-IT to Microsoft Second Light

As described in point **1.3.1**, Ullmer and Ishii's metaDesk [117] had extended the control input of an interactive tabletop with use of different physical tokens and other controls, like the passive lens, to interact with the surface and manipulate digital data. Furthermore, the system supported a 3D navigalbe model of the MIT campus buildings that could be visualized through the use of a LCD screen mounted on an arm support, with a 6DOF magnetic-field position sensor attached to the flat-panel display for tracking its spatial position and orientation. This features are actually transforming the space of a desk in an augmented reality experience, posing the user in a sequence of motor and cognitive tasks that leverages their sensory-motor experience, enabling them to conceive and understand different interaction metaphors. It is important also to notice that with the metaDesk was introduced the back-projection technique, where the projector is not place above the surface anymore, like in the DigitalDesk or the ActiveDesk, but at the bottom of the tabletop and the projected images are reflected by the use of two mirror on a semi-transparent surface (see Figure 12).

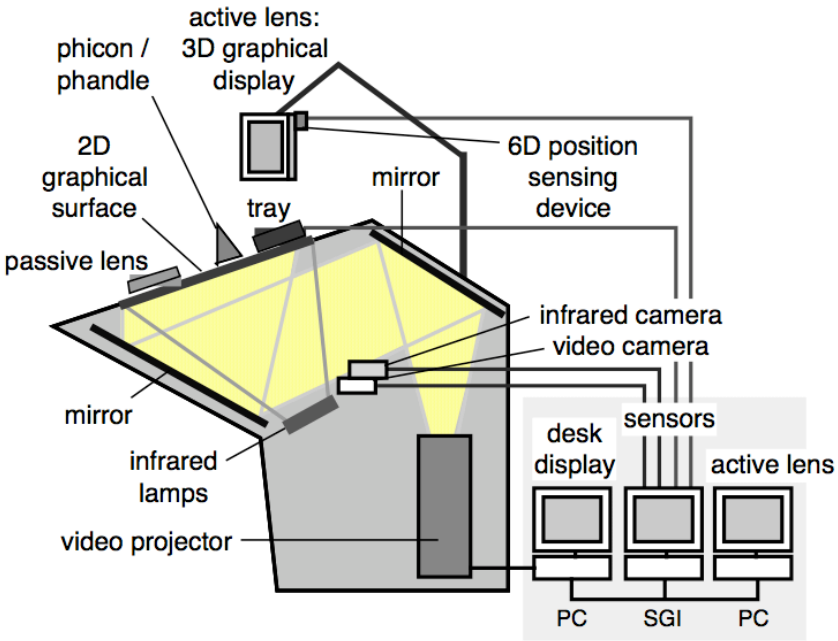

Figure 12: The metaDesk system architecture

A very similar paradigm is proposed by Fjeld et al. in 1998 with their system BUILD-IT[9], where their research question is regarding wether epistemic action can be considered to be a goal-driven or task related activity [37]. To test their hypothesis they develop this new paradigm called NUI (Natural User Interfaces), fusing real and virtual objects through the use of AR (Augmented Reality) as a metaphor for an interaction *"based on the real objects, augmented by computer characteristics"* (Fjeld et al., 1998). Hence, they developed a multi-modal interface made by two different surfaces (see Figure 13): 1) a horizontal surface with a static top projection, two menus and two areas, where users can manipulate physical pucks and interact with each other and the system; 2) a vertical surface with a virtual quasi 3D mensional world projected in it, where user can add and control the position of virtual object connected to the physical pucks. However, this system, unlike the metaDesk, is using a top-projection and a front-projection technique.

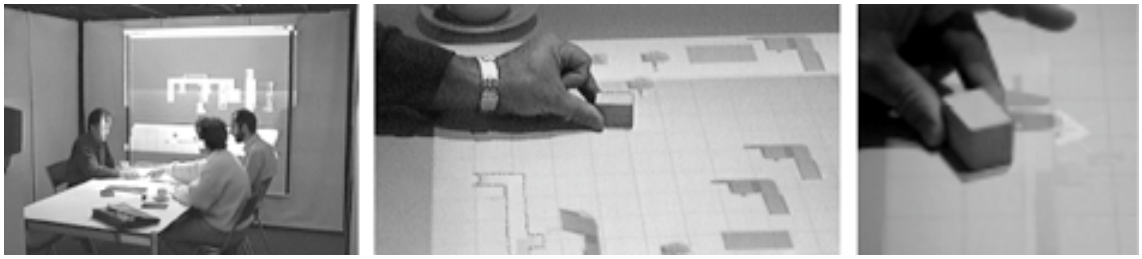

Figure 13: Users interacting with the BUILD-IT system (1998)

After metaDesk and BUILD-IT, the production of new interactive surfaces has drastically increased and several models have been proposed and experimented, trying to refine technological aspects and optimizing efficiency and usability of systems. It is important to underline that, since tabletop interaction is strongly dependent on the involved technology, the study and experimentation of tracking technologies, projection systems, and sophisticated interaction devices has been fundamentally important to understand how and where to improve system's performance and user satisfaction. Due to the parallel development of multi-touch and tangible input methods, some of the research have been more focused on investigating direct manipulation and touch technologies, whereas some have been more focused on tangible interaction, and some have tried to implement both input methods into the same technology. DiamondTouch[10] was introduced in 2001 by Dietz et al. in [29], it's main technological feature was the use of a tracking system integrated into the tabletop (see Figure 14). Infact, through a high-signal transmitted from the surface to the receivers placed in user's chair, touch location information was determined independently for each user, allowing each touch on the surface to be individually associated with a single user. Until today, this is still the only multi-touch technology there is able to discriminate between different users.

[9] http://www.fjeld.ch/hci/#paper_1
[10] http://www.circletwelve.com/

Figure 14: The DiamondTouch (2001)

Similar interactive tabletop technologies were introduced later, such as the Smartskin[11] [109] and the MagicTable[12] [7]. However, while the DiamondTouch was accepting only direct manipulation trough multi-touch controls, the Smartskin and the MagicTable were making use of multi-touch input method, as well as integrating particular kind of TUIs for specific actions and digital data handling. These interactive tabletops were still making use of top projection, which was not integrated in the table itself; therefore the problem of partially camouflaged projection by user's shadows still occurred. The top-projection technique was still used for the SenseTable[13] [101], and PlayAnywhere[14] [123] introduced projection on the side, which partially solved the problem of interface occlusion. PlayAnywhere was introduced by the Microsoft researcher Andy Wilson in 2005, it used an IR-based (Infra Red) tracking system located above the tabletop and it supported multi-touch interaction as well as tagged tangible objects detection. SenseTable was introduced by Patten et al. in 2001, the system used two WACOM Intuos tablet and a series of wireless TUIs, whose position and orientation was electromagnetically tracked by the system via inductive sensing. This system featured one of the first interactive tabletop musical performance applications called Audiopad[15] [102], *"an interface for musical performance that aims to combine the modularity of knob based controllers with the expressive character of multidimensional tracking interfaces"* (Patten et al., 2001). Later, in 2005 Sergi Jordà and other investigator from the MTG (Music Technology Group) of the Universitat Pompeu Fabra have introduced the ReacTable [72], a multi-user electro-acoustic musical instrument with a tabletop tangible user interface.

[11] http://ftp.csl.sony.co.jp/person/rekimoto/smartskin/
[12] http://iihm.imag.fr/fberard/
[13] http://tangible.media.mit.edu/projects/sensetable/
[14] http://research.microsoft.com/en-us/um/people/awilson/
[15] http://www.jamespatten.com/audiopad/

The ReacTable (see Figure 15) presented different technological innovations, such as the integration and location of all its components below the tabletop's surface, as well as an improved topological fiducial tracking system for TUIs detection [4], based on the same approach introduced by Costanza and Robinson in the d-touch system [27]. The GUIs are back-projected on the tabletop via mirror reflection; objects and touch gestures are recognized by a computer vision system which uses a camera positioned under the table. Since the semi-transparent surface of the ReacTable is back-illuminated with IR light, the system allows for unique and independent detection of TUIs tagged with fiducial markers and finger touch.

Figure 15: Some users interacting with TUIs on the ReacTable system

Other interactive tabletops have been introduced with years, proposing different technological solution and interactive paradigms, as in the case of Ortholumen[16] [104], which was introduced by Piazza et al. in 2007 and it used a light pen based interaction. In this system the light beam of the pen is tracked by a webcam positioned below the transparent surface of the tabletop, and the shape of the light informs the system about pen's orientation, position and direction; the system could also be expanded to multiple pen tracking, so as to enable for multi-pointer input and collaborative interaction. Another pen based interaction system was the InfracTables[17] [40], introduced by Ganser et al. in 2005. As aforementioned, this systems supported pen interaction, as well as different set of specific TUIs, such as ruler, eraser, calliper, ink dwell, etc. Device position on tabletop was determined via blob detection and ID was determined through the use of a bit code transmitted from the IR-LED mounted on the device to the SyncBox. FTIR[18] introduced in 2005 by Han in [47] was also making use of IR technology for device detection. This interactive digital tabletop was accepting only multi-touch input, presenting integration of component below the surface as the

[16] http://www.t2i.se/projects.php?project=ol
[17] http://www.icvr.ethz.ch/
[18] http://cs.nyu.edu/≈jhan/ftirtouch/

ReacTable, but unlike the work of Jorda et al., the surface was not back-illuminated with IR-light, whereas single IR leds were coupled into the acrylic overlay from the side, making part of the detection system directly integrated into the surface (see Figure 16).

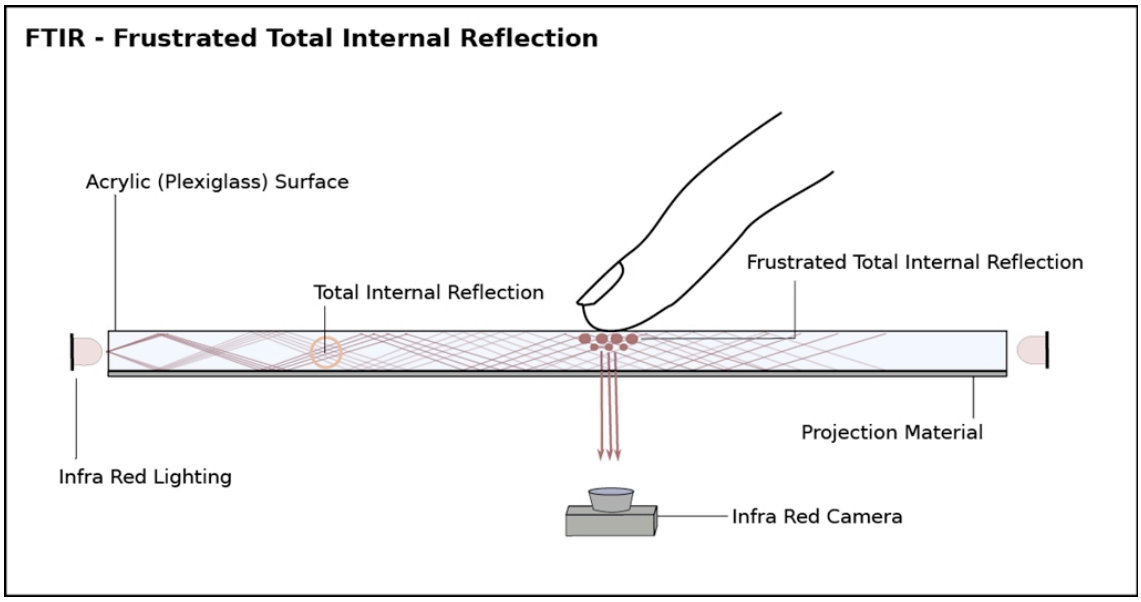

Figure 16: a graphical representation of the system used by Han et al. to build their multitouch surface

However, today's flat LCD screen are representing an appealing technology for tabletop research, in which they are becoming more economically affordable and, since they do not present any cameras and projectors in the way, they can improve ergonomic impact on the user, making it possible to actually sit at the table and interact with the system in a more comfortable way. It is in this direction that were going the works of Hofer et al., like the MightyTrace[19] [55], the FLATIR[20] [56] , or the TNT [54]. Another interesting work is the BendDesk[21], introduced in 2010 by Weiss et al. [121], this system presents an interesting concept of hybrid interactive desk, that combines features of horizontal and a vertical interaction on the same surface surface via a bended section. One step forward has been made by Microsoft with the SecondLight[22] introduced by Izadi et al. in [62], which is the first interactive tabletop that is able to extend the user's interaction beyond the display (see Figure 17). In fact, this system is based on a switchable projection screen, that can be made diffuse or clear under electronic control; therefore when the screen is in diffuse mode the projection is on the surface, whereas when it is in clear mode the projection goes through the surface. The system accepts multi-touch input, particular types of TUIs with internal prisms for on-object projection when the screen is in clear mode, an is also able to detect real physical objects such as real paint brushes or translucent sheet. Indeed, all these different tabletop technologies provide the users with many and diverse interaction options. Hence, a schema of the gestural possibilities and interaction frameworks related with digital tabletops, would

[19] http://www.icvr.ethz.ch/research/projects/closed/motiva/Technolgy/MightyTrace/index_EN
[20] http://www.youtube.com/watch?v=FrO-VnDA-24
[21] http://hci.rwth-aachen.de/benddesk
[22] http://research.microsoft.com/en-us/um/people/shahrami/

help understanding when to focus on the implementation of a particular control and how to properly integrating different input methods into the same system.

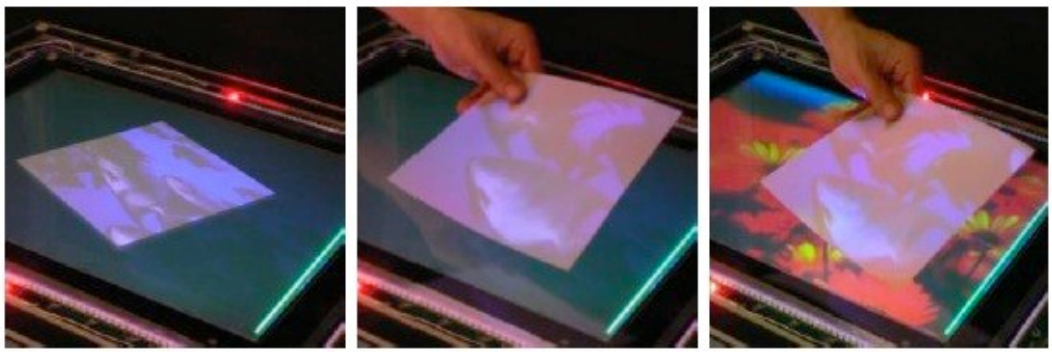

Figure 17: An example of the Microsoft Second Light in action. In the rightmost image is possible to see how this technology allows for simultaneous projectio on the surface and through the surface

1.3.3 Gestures and interaction

Tabletop interactive technologies can be considered as dichotomous systems, in which they are both graphical displays and direct input device. For this reason they provide the user with digital manipulation via natural hand gestures, as well as the control of virtual objects with various type of physical tokens. With the evolution of technology, nowadays digital tabletops can accept without problems multiple input devices, and detect them simultaneously and independently from how many users are interacting with the system. For this reasons, co-located and co-present collaboration are easily supported by interactive tabletops, which have shown great potential in terms of CSCW (Compute Supported Collaborative Work), and the benefits of hybrid input control methods applied to tabletop interaction has been discussed in various studies [8, 50, 53, 57]. Kunz and Fjeld in [82] have proposed a small classification of the possible types of devices and interaction related with a tabletop system (see Figure 18). TUIs, multi-touch and pen-based interaction, seem to be still predominant input control methods used for tabletop interactive technology. Since our work was focused on comparative analysis of TUIs and multi-touch, the next section will be describing these input control methods more into detail.

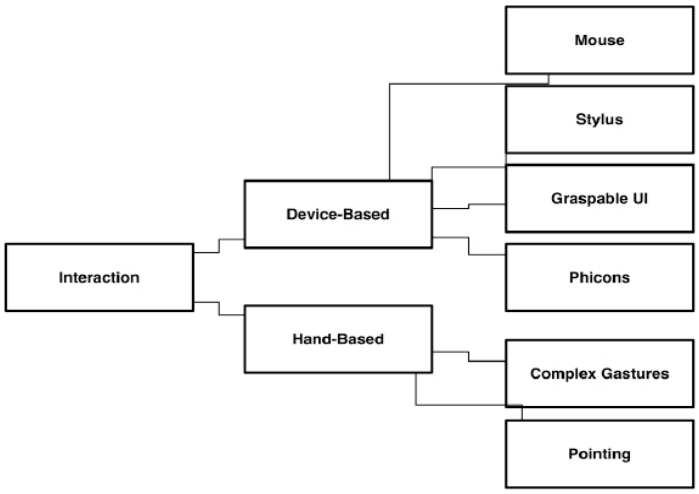

Figure 18: Readapted classification of tabletop interaction by Kunz and Fjeld

1.4 Tabletop Input Control Methods

1.4.1 Tangible User Interfaces (TUI)

The concept of TUI was introduced in 1997 by Ulmer and Ishii in [61], and then also extended by the same authors in 2000 in [116]; these controls where firstly implemented in the metaDesk (see point **1.3.1**) to augment tabletop interaction; through the use of physical objects called phicons, users could manage and manipulate digital contents retro-projected on the tabletop surface as if they were physically extended beyond the virtual application. However, the very first groundwork for the discussion on this new interfaces was laid in 1995 by Fitzmaurice, Ishii and Buxton in [36], their premises were to represent an emerging post-WIMP paradigm, concerned with providing tangible representations to digital information and controls, then allowing the users to actually "grasp" digital data directly with their hands (see Figure 19).

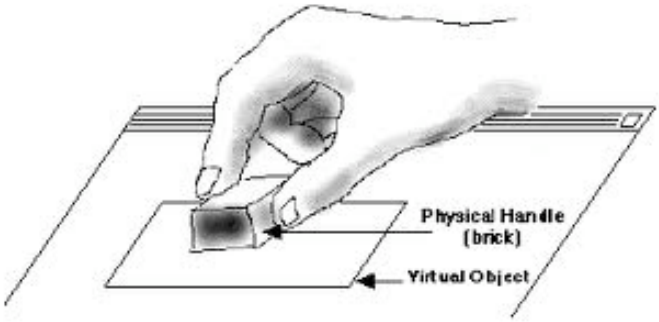

Figure 19: A graspable user interface as represented in [36]

These works where indeed inspired by Durrell Bishop, who designed a conceptual sketch of a Marble Answering Machine[23] [110] while studying at the Royal College of Art in 1992, where incoming calls were represented by tangible objects in shape of little colored spheres of marble. Once the answering machine had registered the incoming message, the little sphere rolled into a bowl embedded in the machine, than the user could pick up the sphere and place it in the identation where the message(s) could eventually be played back (see Figure 20). In this case, the mapping of digital contents into a physical object semantically changes the meaning of the latter, where it actually becomes an augmented object whit a meaning other than a simple sphere, but the sphere actually "is" the information contained in it. The Bishop's answering machine, the graspables of Fitzmaurice, Ishii and Buxton, and the tangible bits of Ulmer and Ishii, can all be considered as pioneering and seminal works in the field of tangible user interfaces, and with them they have surely exemplified two important aspects of this technology: 1) how TUIs are potentially not limited to the visual and aural senses, providing the user with additional haptic feedback, and then extending the WIMP interaction paradigm of digital data manipulation to a more natural and embodied interaction, 2) how TUIs take advantage of the human predisposition to be active and creative with one's hands, providing a better way of interacting with computational applications that leverage user's knowledge and skills of interaction which rely on affordances of everyday, non-digital, world and its objects. This concept is reinforced by Paul Dourish in [31], who says that the positive outcome towards this embodied interaction paradigm, also reinforces the idea of integrating computation into

[23] http://vimeo.com/19930744

our everyday world, in which *"tangible and social computing both capitalize upon our familiarity with everyday world, a world of social and physical interaction"* (Dourish, 2004). Another important benefit of TUIs is that they encourage two handed and bimanual interaction [43, 73]. As previously discussed in point **1.2.2**, TUIs have also shown to improve performances at spatial layouts when the handling of different data at the same time is required; the reasons of this benefit are extensively explained in [36].

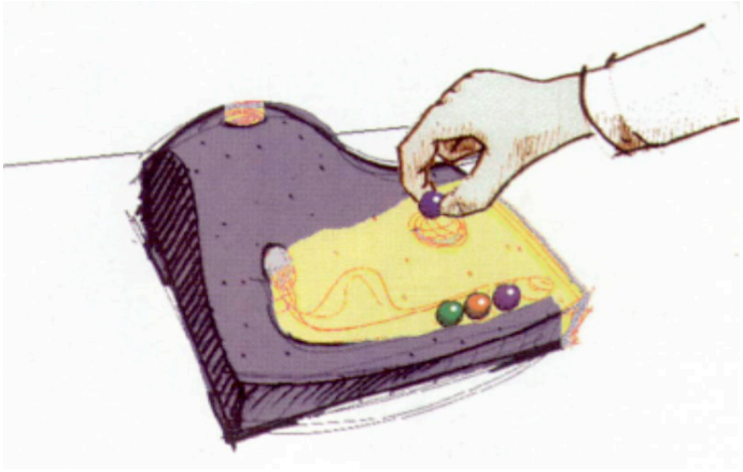

Figure 20: A sketch of the Durrell Bishop's Marble Answer Machine (1992)

However, since tangible interfaces have shown to be a promising technology in HCI, many implementations have been done in different systems and in different interaction domains, and different taxonomies have been proposed [34, 36, 58, 115]. John Frazer can be considered on of the pioneers in implementing tangible controls for the manipulation of digital data. In fact, even before GUIs or TUIs, i.e. at the end of 1970, he had already foreseen the future application of tangible controls, by using physical models as input devices for CAD systems in collaborative design discussions on architecture [39]. Thinking about the technological state of that time, this system was quite advanced, since the single cubes used as input devices where able to sense the positioning of their surrounding neighbors. Only thirty years later, David Merrill's Siftables[24] from MIT medialab, has shown a similar concept [96]. A similar approach to Frazer was used in 1999 by Underkoffler and Ishii in [118]; with Urp they have introduced a system that supported collaborative discussions on urban planning, by mean of using physical building models and various pucks for digital data manipulation on a luminous interactive tabletop. TUIs have been successfully used in the design and construction of toys for children, as in the case of Topobo[25], which is a robotic toy made by different parts that can be joint together to make different and personalized combinations. Furthermore, this toy is provided with kinetic memory, which makes possible to record and playback physical motions and actions without having to perform any programming with the computer; Topobo has also been proven to teach advanced physics concepts to children as young as 5. Many TUIs are also used for AR application, in order to connect physical objects to virtual contents that can be visualized in an external monitor or digital screen. MADO is an example of this type of applications, where users can build physical models by combining electrical pieces, then connecting this model to the MADO interface to watch and explore the physical model

[24] http://alumni.media.mit.edu/≈dmerrill/siftables.html
[25] http://www.topobo.com/

in the virtual world [92]. Another example of TUIs implementation for AR is the work of Huang et al. Easigami[26], which is a system intended to help children to develop their visualization and spatial recognition skills, through the use of augmented reconfigurable set of flat polygons (see Figure 21), electronically connected to a computer for 3D model visualization in real-time [59].

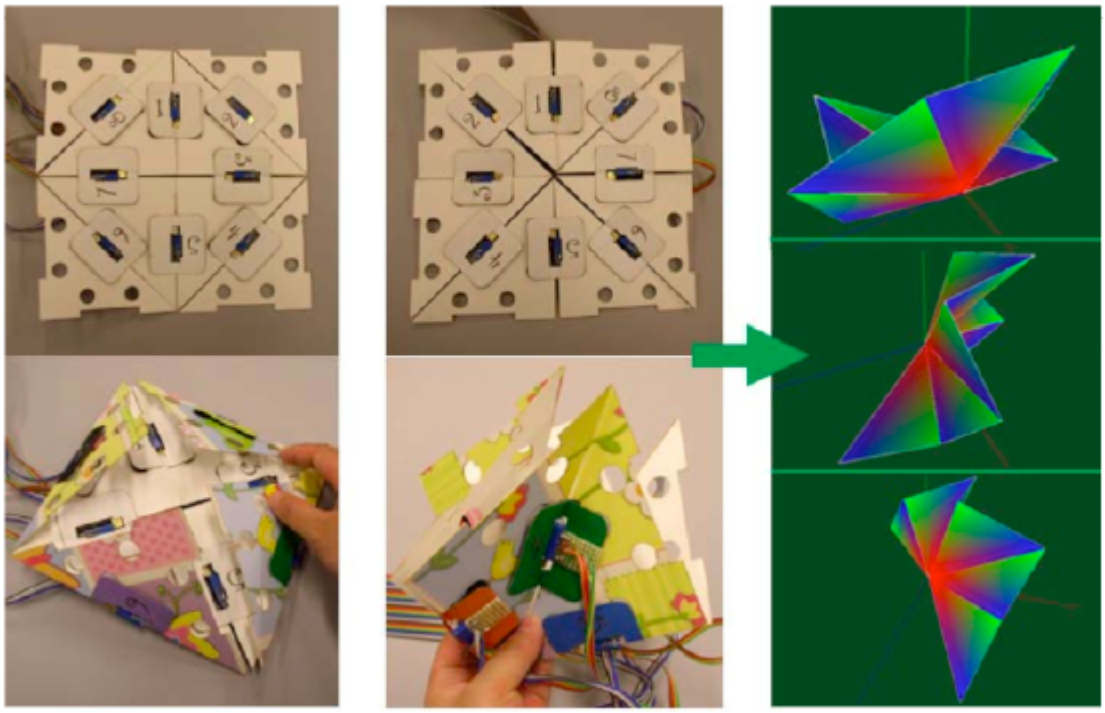

Figure 21: An example of foldable TUI and their digital representation, in the work of Huang et al. Easigami (2009)

Tangible interaction has been often used, especially with children, to improve and facilitate learning processes. It is the case of AlgoBlock, where TUIs have been used by Suzuki and Kato in a tangible programming language as a computer-based educational tool to facilitate interactions among young learners [112]. The premises of this work were to discuss computer-based education from the point of view of situated learning theory [11, 83] and the idea proposed by Lave et al. of LPP, i.e. legitimate peripheral participation [83]. A survey recently conducted by Marshall et al. on embodied interaction, have entailed experiments with children making use of both TUIs and multi-touch controls for collaborative problem solving, producing interesting results in favor of tangible and embodied interaction [94]. Their analysis have shown how material qualities of the digital interface and physical objects can affect behavioral strategies adopted by children, where using tangible devices in lieu of multi-touch led to less aggressive strategy for objects control. Albeit Marshall et al. research is not unashamedly referring to a comparison of TUIs and multi-touch input methods, the final results are inevitably putting these two on different positions, showing inherent limits and benefits of these different technologies applied to same or different interaction paradigm. The next session will give an overview of multi-touch technology.

[26] http://www.youtube.com/watch?v=QteE8F4n7is

1.4.2 Touch-screen and Multi-Touch

Touch screen and multi-touch control methods are having nowadays great rise and interest amongst emergent novel technologies, and always more often they are implemented in mobile devices and digital tabletops, such as Apple iPhone[27] and iPad[28], or Microsoft Surface[29] for instance. Some of the early and recent application of touch and multi-touch on interactive tabletops have already been described in point **1.3.2**. However, despite seeming a very young phenomena, the creation of this technologies and techniques is going quite far back, i.e. at least more than 40 years ago for touch screen technology and about 25 years ago for multi-touch. It is certainly sobering how long this technology took from inception to ubiquity; an article wrote by Bill Buxton in 2008 helps understanding this issue [24]. Perhaps, we can consider one of the earliest touch sensitive devices ever implemented for manipulation and control of digital signals, the Electronic Sackbut[30] developed by Hugh Le Caine between 1945–1973. This instrument featured an innovative control of waveforms and timbre through a sensitive touch pad, that was mounted on top of a regular organ keyboard and controlled by the left hand. Between the mid 1960s and the early 1970s the IBM and the University of Illinois started to develop and disclose the early touch technology prototypes making use of touch input device to interact with computers; it is the case of PLATO IV[31] system (see Figure 22), a touch sensitive computer used in grade-school classrooms for education assisting.

Figure 22: PLATO IV terminal with touch-screen and plasma display (1952-74)

By the late 1970s Chris Herot and Guy Weinzapfel from the Architecture Machine Group at MIT, realized one of the first touch sensitive screen that could not just sense positioning of fingers [49], but through the use of a mounted touch-screen overlay on strain gauges, the device was able to sense vector information in six dimension, i.e. force in x, y and z as well as torque force on the same axis. More example of early touch-screen technology and touch sensitive devices are illustrated by Bill Buxton in [12].

[27] http://www.youtube.com/watch?v=grTCDFbhKMU
[28] http://www.youtube.com/watch?v=y2Hz8dhQw8Q
[29] http://www.microsoft.com/surface/en/us/default.aspx
[30] http://www.hughlecaine.com/en/sackbut.html
[31] http://en.wikipedia.org/wiki/Plato_computer

However, the aforementioned technologies were partially limited by the fact that they were only able to sense one point of contact per time. As mentioned in [12], the very beginning of research on multi-touch technology goes back to the study of tactile sensors for robotics. In 1986 Paul McAvinney introduced an optical sensing device called the Sensor Cube, which used cameras placed across the surface of the display to detect touch location and angle approach of up to three fingers [95]. Before that, in 1984, the same year the first Machintosh computer was introduced, Buxton et al. developed a touch tablet capable of sensing an arbitrary number of simultaneous touch inputs, reporting both location and degree of touch for each [84]. Due to the interactive potential shown by this first prototypes, a great interest and advanced researches born towards the various possibilities that could be expressed by multiple finger's gestural pool, in relation with the use of multi-touch input control method for the manipulation of various types of parameters. Although not using direct contact on a surface, some of this potential was already shown by Myron Krueger in 1983 in his works Video/Place[32] and Video/Desk[33]; this pioneering works [80, 81] were really pushing the boundaries of capturing human gestures, including a widely implemented type of gesture in modern touch-screen sensitive devices, such as the pinch gesture to scale and translate objects (see Figure 23).

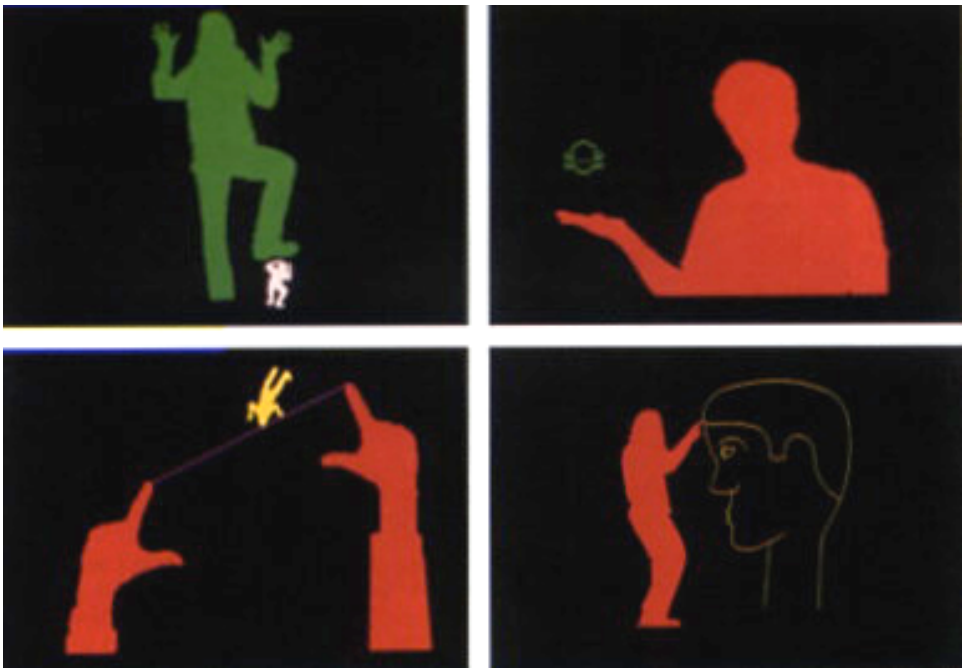

Figure 23: Myron Krueger's pioneering work VideoPlace, early work using rich interaction gestures (1974)

Many studies on hand gestures and bimanual input control applied to interactive systems and multi-touch devices had been carried out by Bill Buxton and different authors [17, 22, 23, 85], to investigate the benefits of interaction paradigm that could advantage the use of human gestural capabilities. In particular [17] highlights how the combination between a conventional device used for GUIs direct manipulation (e.g. a keyboard or a mouse) and a multi-touch control device, could eventually improve and

[32] http://www.youtube.com/watch?v=dqZyZrN3Pl0
[33] http://www.youtube.com/watch?v=d4DUIeXSEpk

increase the range of direct manipulation and actions a user is able to perform. This was achieved on purpose by creating a keyboard which has a multi-touch panel integrated into the bottom, which a user could simply "flip" to have access to the multi-touch pad, and take advantage of it's features when richer and more complex simultaneous finger gestures were required from the application, as in the case of a virtual mixer for example (see Figure 24).

Figure 24: The Tactex's Flip Keyboard

This work shows the importance of having a versatile control, that might eventually allow a user to chose more naturally and more efficiently the way he or she wants to interact with a system. However, according to Buxton [18], when we are designing or choosing a control method, a fundamental axiom that we have to bear in mind is: *"everything is best for something and worst for something else"* (Buxton, 2008). Hence, we will see in the next section, how in tabletop interaction, multi-touch control compared with TUI and other input devices, has shown to be a good input control method for some actions and bad for some others. Especially when very precise gestures were needed, multi-touch has been found to be more error prone with respect to the other types of control. Of course, this results were not only dependent on physical characteristic of the controls, but they were also dependent on the design of the application and the type of tasks that user was required to accomplish.

1.5 Comparative analysis of TUI and Multi-touch

1.5.1 TUI and Multi-touch differences and similarities

As seen in previous studies [26, 63, 89], mouse input device was used as a good reference for comparative studies on different input control methods, due to his large implementation and use in interacting with GUIs. However, since the tendency has changed in the last years, different input devices have become more diffuse and used in HCI and the focus of research has been shifting from conventional to novel technologies, such as TUIs and multi-touch. As input control methods, tangibles and multi-touch present many similarities (see Table 1), and since both devices support bimanual interactions and spatial multiplexed, it is natural to inquire whether one technology can present the same advantages and quality of the other, and to what extent they can be transferable.

Table 1: Differences and similarities between TUI and Multi-Touch

	TUI	Multi-touch
Tactile Feedback	Yes	No
Manipulation space	3D	2D
Specialized device	Yes	N/A
Space-multiplexed	Yes	Yes
Support two-handed action	Yes	Yes

Generally, two issues become relevant when the case is analyzed from the perspective of multi-touch: 1) since multi-touch control does not have a specific physical form, it cannot be categorized as a *specialized* or *non-specialized* device, 2) can multi-touch enable the same motor-cognitive activity elicit by TUIs? Two major differences shown by this technologies are partially answering this question:

- When user is interacting with a system using multi-touch control, he or she cannot actually lift or hold objects with their hands, making the manipulation space 2D and not 3D as in the case of TUIs; we see, for example, how this affected grade of aggressiveness in interaction strategies among children in [94].

- Multi-touch controls are not providing tactile feedback, then they are relying more on visual feedback than TUIs; some implications of this issue are discussed by Buxton in [18].

1.5.2 Previous works

Tuddenham et al. compared user's performance using TUIs, multi-touch, and a mouse-and-puck input devices for a task implying precise shape acquiring and matching [114]. Their quantitative analysis demonstrated that TUIs were easier and faster to acquire, and more accurate to manipulate with respect to the other two interfaces. One phenomena pertaining multi-touch interfaces emerged as problematic issue when users had to cease the contact with the surface to submit their performance, i.e. the so called "exit error", which was principally caused by the slight shift of a contact point, such as lifting the fingers off the screen. For this reason, the multi-touch interfaces have shown to be less effective for fine-grained controls than TUIs. Lucchi et al. compared tangible

and multi-touch interfaces for virtual/physical small-scale walls and shelves sorting [86]. Their quantitative results have shown that generally tangibles are faster and more accurate for spatial layout tasks, but also that multi-touch could perform better under certain condition, especially when users could take advantage of some features not provided by tangible objects, such as “undo”, “select all” or “lazo selection”, that could simplify and speed up user actions under certain circumstances (see Figure 25).

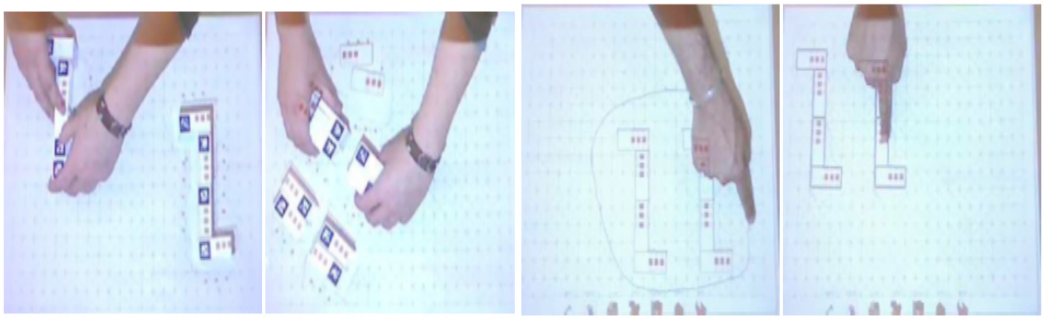

Figure 25: Some gestures performed from subjects during the experiment of Lucchi et al., image on the right side shows the "lazo selection" option provided with multi-touch control

Furthermore, quantitative analysis confirmed how tangible interfaces were easier to acquire due to their familiarity, but authors have proposed the design of application requiring less complex gestures for multi-touch control to reduce this gap between these two control methods. Kirk et al. in [78] conducted case studies of the design of two hybrid interactive surface technologies: VPlay and Family Archive. They have investigated important aspects in designing such applications: 1) How to detect and rationalize key features of digital and physical controls in relation to specific applications? 2) Once rationalized, why choosing one or the other? 3) How the physical world should be emulated into the digital domain? What they have suggested is that TUIs are particularly advantaging in applications where eyes-free control and rich or accurate control are needed. However, multi-touch shows indeed some advantaging features proper of digital domain, that simply cannot be achieved with tangible objects for obvious physical reasons, such as dynamically copying or deleting an object. Creating interaction paradigm that implies manipulating emulation of physical objects in 3D world on a 2D surface can be challenging, but designers have to be prepared to even break the law of physics to provide users with an effective interaction. In this sense, their work takes an important step for future research in tangible and multi-touch interfaces. Terrenghi et al. have compared in [113] the use of physical and multi-touch interfaces for puzzle solving and photo sorting tasks. Their findings are showing fundamental differences between this two control methods, in which, even though they have designed the digital tabletop to facilitate bimanual interaction, participants were actually more prone to use bimanual interaction with physical objects than with digital ones. Also, physical objects have shown to elicit more frequent use of the non dominant hand, providing the frame of reference to the dominant hand. However, not many experiments in this field have investigated the use of TUIs and multi-touch for the control of real-time video games, or more generally real-time applications, where new interesting results and implications may arise.

1.6. RTI in HCI and video games

1.6.1 Defining RTI

The application used for the experiment described in this thesis was developed in the context of real-time interaction video games. For this reason the next paragraphs will try to explain the concept of RTI and mark its differences with respect to other types of interaction. Since this concept is not well defined yet, the basic assumption on which it is intended to be built, will use a mixture of studies on interactivity and video gaming as a reference for the construction of the theoretical framework.

1.6.2 Main concepts and ideas

While a broad research has been carried out regarding the meaning and the implication of interactivity and interaction in different fileds, like communication [77, 108], HCI [30, 32], virtual reality [100, 111], humanities and arts [103], digital music [70, 71], less attention has been paid on how and when in time this interaction should happen, as well as the importance of the role of time in the interactive process. First studies trying to frame the importance of time relationship between user and computer in HCI appeared in the eraly ninties [48]; althought being mainly focused on technical issues related to usability and design problems, this study raised the important question of user's asynchronous actions in relation with the sequential nature of actions imposed by the "old style interfaces" of computers. What emerges form this reflection is that, computer time processes and human temporal pattern of actions has to be naturally related to each other in order to produce a better or more "realistic" outcome in the interactive process. The notion of RTI (Real-Time Interaction) tries to delineate a theoretical framework, on which trying to analyze and improve the natural process of interactivity between users and computers. Since our actions in the world follows real time rules and constraints, i.e. we cannot go back in time for instance (even though Mallet[34] would not agree on this point [93]), it would be resonable under certain circumstances to interact with systems that can generate informations following the same rules. It is important to understand prior of everything that the concept of RTI does not take into account velocity as a key issue; in real-time generation of contents and data, what matters is not that they are generated in the fastest way possible, instead the word real-time in this case referes to a simulation that proceeds at a rate that matches that of the real process that it is intended to be simulated. An example that can help explaining the notion of real-time within interaction, is related to VR (virtual reality), in fact according to Steuer, amongst one of the basic requirements of VR systems is the one of having a computer capable of generating animation and data in real-time [111]. The capacity of a system to generate contents "on the fly" it's indeed an important quality to achieve a condition of RTI; for example, while interacting with a VE (Virtual Environment) in a VR experience, the user would expect visual contents and perspectives to change dynamically, according to his head movements and changes in point of view. However, dynamic generation and modify of contents in RT do not have to be necessarily as fast or as quick as possible.

[34] Ronald Mallet was a professor of physics at the University of Connecticut since 1975. He is best known for his scientific position on the possibility of time travel.

1.6.3 RTI in video games

For instance if the VR system is emulating a real-life tennis match (see Figure 26), one would expect his virtual oppenent and the virtual object represented in the virtual environment to respond according to time that resembles the one of real-life or similar. In this case, having a virtual opponent or a tennis ball that goes much faster than one's natural times of reaction, would produce a stressful interaction and non plausible simulation. Steuer also defines immediacy of response in RTI as a very important characteristic to generate the conditions of interaction, describing a sort of criteria that define grades of interactivity of media, based on their capacity to be as more sumiltaneous and immediate as possible [111]. According with the aforementioned case of the simulated tennis match, this criteria does not hold anymore.

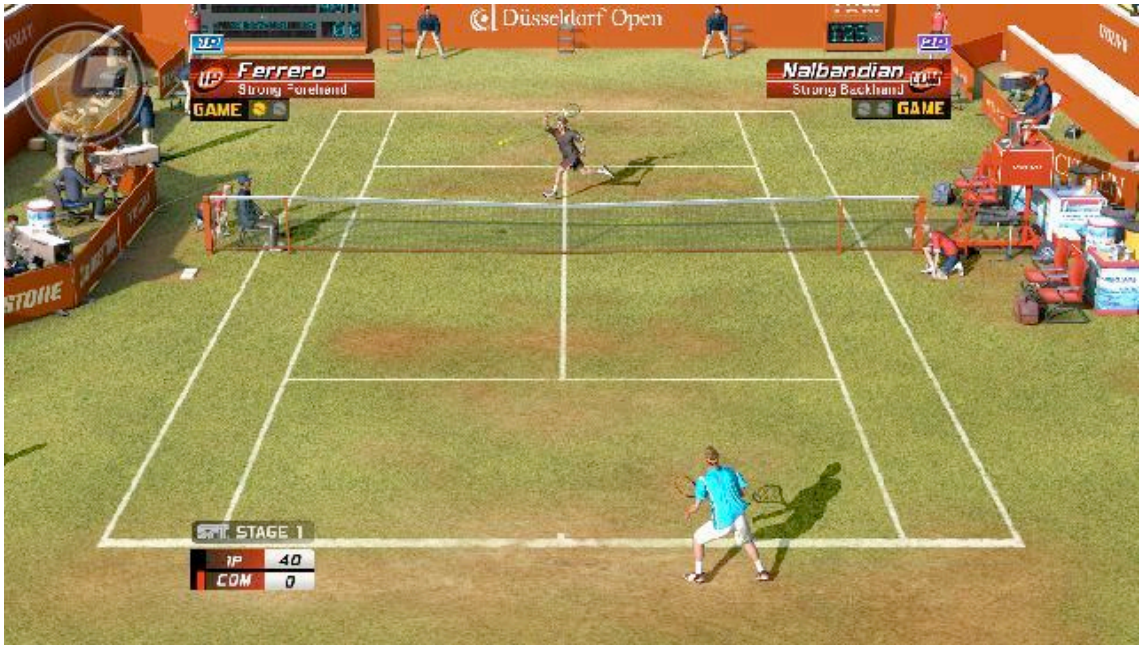

Figure 26: Virtua Tennis, an example of real- time videogame sport simulator (Copyright of SEGA Corporation)

One of the main characteristics of real-time video games is that time progresses continuously according to the game clock. Players do not perform actions in a sequential fashion, instead they perform various actions simultaniousely and continousely in time; for instance, in RTI it's not possible to "undo" an action. Furthermore, objects and opponents present in the game are behaving according to real-time conditions, therefore they may act or react to player's moves at any moment. Time management and physical coordination are fundamental in this type of interaction. Real-time gameplay is often used in simulation video games for entertainment or training, e.g. flight simulators. For instance, the notion of RTI perfectly fits to the concept of flight simulators, since what is intended to be achieved in this type of video games, is the reproduction of real flying conditions as more realisticly as possible. To achieve this particular conditions, it is of fundamental importance that all actions occurs in real-time, in an environment that resembles time and physical constraints of a real life environment. However, not all the video games or appliaction that are designed following a RTI approach, must compulsorily match the condition of real life. For

instance, online RPG such as “World of Warcraft”[35] (see Figure 27) and many other real-time strategy video games, do not necessarily follow the real physical and gravitationl rules that hold in the real world. However, the interaction between players in this type of video games happens in real-time, since time progress cannot be stopped and the virtual environment behaves and changes indepentently if the players are playing the game or not.

Figure 27: Worl of Warcraft RPG game online (Blizzard Entertainment)

Due to the sequential nature of computing, game time in video games is actually subdivided into discrete units or intervals, but these are typically so small to end up being imperceptible to the player. Therefore, even though the actions in game time are happening in a sequential and discrete fashion, at high rates they are perceived as a continuum. The notion of RTI might be confused with RTC (Real Time Computing), where the concept of "real-time constraint” is associated with operational deadlines from event to system response. However, the realtionship between agents and time constraints in RTI is similar to the one we find in RTS (Real Time Systems), which are often OO (Object Oriented) modeled and operate under real-time constraints; in 1998 Nielsen et al. proposed a RTS made by two separate but complementary levels, i.e. *functional-level* where a collection of objects defines the system’s structure and behavior, and a *constraint-level* where a set of interaction constraints define the way these objects may interact with each other [99].

[35] http://eu.battle.net/wow/en/

2. SYSTEM DESIGN

2.1 RTI Application in Tabletop Interaction

2.1.1 Design Goals

As shown in previous studies [78, 86, 94, 113, 116], interactive digital tabletops are particularly suitable to perform studies on comparative analysis of TUIs and multi-touch, in which they can easily afford and implement both interfaces on the same surface. As previously said, most of the comparative analysis of these control methods have been done using non RTI applications, and little attention has been paid by the scientific community towards the possibility of testing them in a RTI context. For this reason, this study expressly focuses on understanding the implications of comparing TUIs and multi-touch controls in a tabletop video game interaction context, under RTI rules and constraints. In order to analyze both controls performance and answering to our research questions, two "video game style" RTI applications have been designed and programmed, and subsequently implemented in an interactive tabletop system. To keep the graphical background very simple and clear for the subjects that would have been taking part to the experiment, the graphical style chosen to design the application was based on the simple aesthetic character of the 1970s, 1980s arcade video games. Since the experiment was build to test both tangible and multi-touch control in a video gaming performance, through parameters such as accuracy, precision and rapidity (see point **3.4**), two classical arcade video games have been regarded as the most appropriate model on top of which developing our application, i.e. the classical Atari Pong[36] and the Arkanoid[37] (see Figure 28).

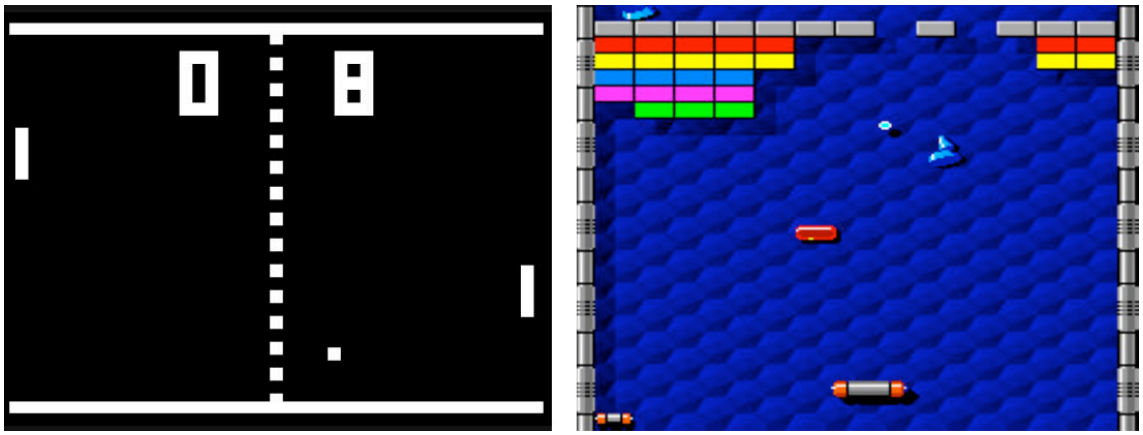

Figure 28: Two screenshot from the Atari Pong (left) and the Arkanoid (right) video games

Generally, video games of this kind are requiring the user to intercept or deflecting moving targets by performing rapid ballistic movements and accurate gestures; therefore the behavior and the difficulty of the game have been modelled according to the Fitts's law, for the creation of an index of difficulty [35], and to the experimental results on target interception and reaction time illustrated by Brenner et al. in [28, 9] and Georgopoulos et al. in [105], taking into account issues of speed coupling, stimulus

[36] http://www.pongmuseum.com/
[37] http://www.arkanoid.com/

dependent reaction time, time planning, visuo-motor delay, compensation for latency, ipsilateral/contralateral area and speed-accuracy tradeoff. Two applications were then programmed for two different task to be performed in the experiment:

- application for Task1 was designed and programmed for target deflection and hit, to test performance as the product of accuracy and precision (see Appendix C, 1.1, 1.2);

- application for Task 2 was designed and programmed for target interception, to test performance as the product of precision and rapidity (se Appendix C, 1.3, 1.4).

Once programmed, the applications were then implemented into the ReacTable hardware and integrated into the reacTIVision software as described in the next points.

2.1.2 Hardware

The hardware setup used in this experiment was the classical ReacTable set up (see Figure 29), consisting of a round table with a luminous acrylic transparent surface, measuring 90 cm in height with a diameter of 95cm. The whole structure was made of resistant aluminium covered with black plastic on the lateral perimeter to avoid environmental light penetrating inside.

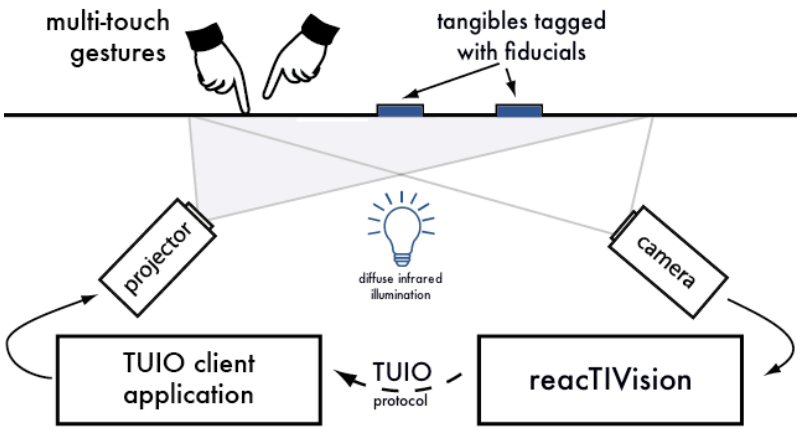

Figure 29: The ReacTable system overview

The electronic part of the system consisted of two IR lights, projector and a camera to be connected to the video I/O of a computer. The projector used for this experiment was a BenQ MX710 using DLP technology, with a native resolution XGA of 1024 x 768 and a noise level of ≈30.0 db; all the projected images were reflected on the back of the surface through the use of a mirror positioned in front of the projector. The camera used was an Allied Guppy[38] F-033B/C fireware camera, with a Sony ICX424 CCD sensor and a frame resolution up to 60 fps. Two small IR light were

[38] http://www.alliedvisiontec.com/us/products/cameras/firewire/guppy.html

placed at the bottom of the structure to permit the camera to capture fingers and fiducial images from the transparent surface. A pair of regular studio speakers were used to provide sound feedback. One piece of acrylic tagged with a fiducial marker was used to implement the tangible control into the system; the fiducial marker placed at the bottom of the object was part of the basic “amoeba” set provided with reacTIVision software (see Figure 30). Detailed implementation of controls and software specification are described in the next point.

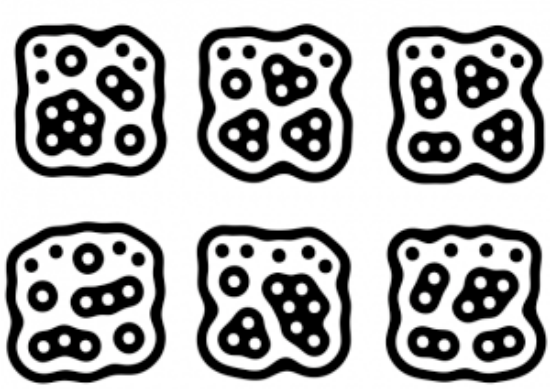

Figura 30: A basic "amoeba" fiducial set provided with reacTIVision software

2.1.3 Software

The fingers and object tracking software, as well as the two applications programmed in Processing 1.2.1[39], run on MacOSX Leopard version 10.5.8. Finger and fiducial marker detection was done using the reacTIVision[40] tracking library [4, 74, 75], which analyzes black and white image, returning a list of fingers identified as “f” and a list of markers each one identified with its single ID (see Figure 31).

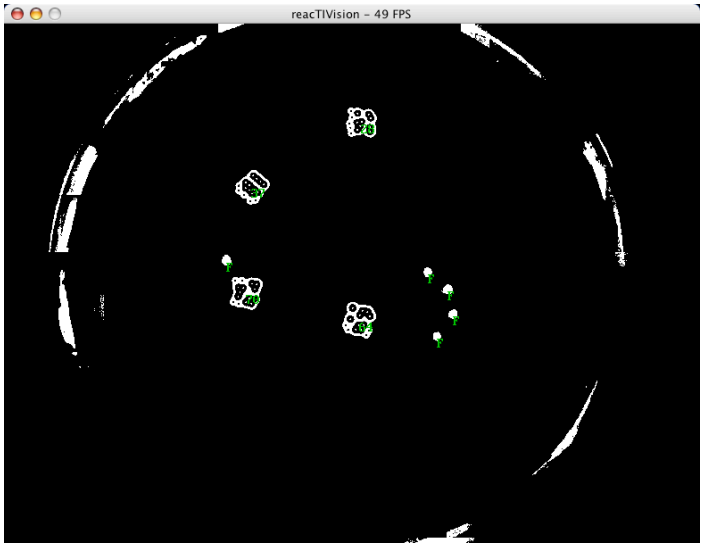

Figura 31: A visualization of the camera tracking system of the reacTIVision software

[39] http://processing.org/
[40] http://reactivision.sourceforge.net/

Information about position, angle, speed and acceleration were extracted via TUIO[41] library in Processing for both fingers and fiducial markers (see Appendix C), then mapped for the control of virtual objects in the following way (see Figure 32):

- TUI: direct mapping into virtual object of x, y and angle parameters extracted from fiducial markers

- Multi-touch: determining the center of the virtual object by calculating the euclidean distance between the two fingers; angle calculated in radians using the center of the virtual object as the origin point

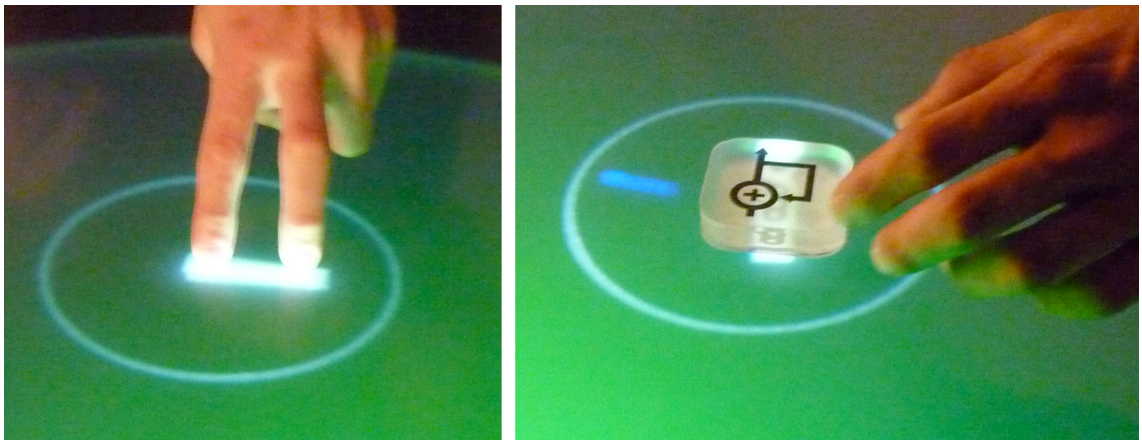

Figura 32: A screenshot from the experiment showing the mapping strategy for the two diferent controls

The two RT applications were programmed in Processing 1.2.1 with the integrated use of the following libraries (see Appendix C):

- TUIO library for finger and fiducial tracking

- Processing OpenGL and Javamediax[42] for graphic rendering

- Fisica[43] library for Processing 1.2.1, was used to create the physical models; the library was created by Ricard Marxer for Processing programming environment

- Minim[44] audio library for Processing 1.2.1 was used for the generation of sound effects in the game environment

[41] http://www.tuio.org/?processing
[42] http://processing.org/reference/libraries/index.html
[43] http://www.ricardmarxer.com/fisica/
[44] http://code.compartmental.net/tools/minim/

3. METHODS

The following experiment has been created to assess gaming performance in a real-time tabletop video game context, described as *accuracy-precision* for Task 1 and *precision-rapidity* for Task 2, with the use of both tangible and multi-touch input control methods. Qualitative data regarding usability, efficiency, effectiveness, satisfaction of controls and user experience, have been gathered trough post-test and interviews. The experiment was based on a task oriented performance, and aimed at answering the following research questions:

- RQ1: Is there any significant difference between tangible and multi-touch in performing accurate and precise gestures in a real-time tabletop video game context?

- RQ2: Is there any significant difference between tangible and multi-touch in performing precise and rapid movements in a real-time tabletop video game context?

3.1 Experiment Setup and Protocol

The experiment involved 12 subjects, the design used a within-subjects repeated-measures. Each of the 12 subject matched both the two different tasks (see Figure 33) using each of the two input technologies: tangible and multi-touch. The experiment was split in two sessions; in each session the subjects had to repeat both Task 1 and Task 2, using one of the two control methods. A counterbalanced AB, BA method was used to determine what type of control the subject should have in the first session and what in the second, so as to avoid order effects. During the experiment, score, accuracy, precision, rapidity and reaction time were measured; at the end of each session, subjective report on comfort, ease of use, satisfaction and user experience were taken.

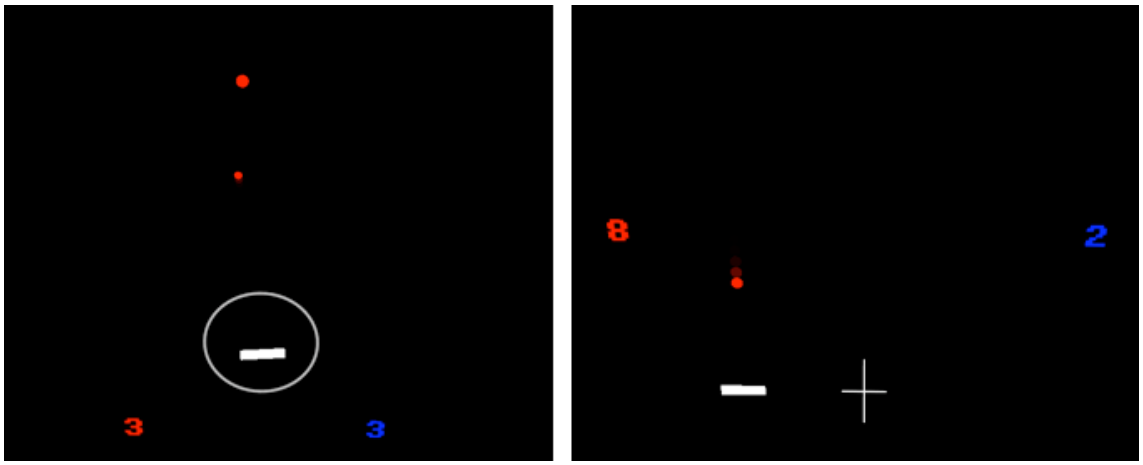

Figure 33: Two different tasks of the experiment, on the left side the target deflection Task 1, requiring accuracy and precision, on the right side the target interception task requiring precision and rapidity.

The subjects worked with one standard ReacTable puck and direct multi-touch input control. After a short training/warming up session where subject had the possibility to play with the same application of the experiment, they where asked to perform the real tasks with ≈2-3 minutes rest between one task and the other. Each task lasted ≈5 minutes; during the first session, a pre-test was filled in by the subjects at the beginning of the session and a post-test was filled at the end; the second session only implied the post-test at the end.

The average time of each session was ≈45 minutes (1 hour and 20 minutes for the whole experiment, i.e. session 1 + session 2). A training/warming-up phase was provided to all the subject before starting the completion of the tasks; this part of the session allowed them to get acquainted with the system's interface before starting the real experiment. The stages were structured as follows:

- Subjects were received and guided to a desk. Once there, they were asked to read and eventually sign a consent form of authorization for the academic use of the data generated during the experiment (duration: ≈5 min)

- If agreed with the consent form, subjects filled in the pre-test questionnaire (duration: ≈5 min)

- Training/Warming-Up phase: Subjects were guided to the ReacTable to start the experiment, where they received information about the tasks and the type of control to use. Further information were displayed on the ReacTable at the time the application started. Then, the subjects had a short period of time to practice and to get acquainted with the interface, the applications and the control before the beginning of the real tasks (duration: ≈4-5 min)

- Task 1: the subjects started the session with ball deflection and target hit; before the time elapsed, the subjects received a tone which was warning them that 10 seconds were left before the end of the task (duration: ≈5 min)

- Rest Phase: Subject were allowed to rest for a short period of time before proceeding to the next task (duration: ≈2-3 min)

- Task 2: the subjects continued the session with target interception; before the time elapsed, the subjects received a tone which was warning them that 10 seconds were left before the end of the task (duration: ≈5 min)

- End of tasks: subjects were asked to leave the ReacTable and guided to a desk to fill in the post-test questionnaire.

- Subjects filled in a post-test questionnaire. The subjects were provided with a series of adjectives to choose, with which they could describe their experience (see Appendix B, 1.2). Then, they had to indicate the three most preferred amongst the ones previously chosen, and discussed about these preferences in the post interview (duration: ≈5 min)

- Interview: subjects were guided to a specific sector of the room to be interviewed; after a brief talk about their experience and their opinion about the control used, subjects could finally leave the room for the end of the session (duration: ≈10 minutes)

3.2 Sample

A total of 12 volunteer subjects, with a mean age of 26.25 y/old, all males, 2 left handed, 10 right handed, with previous experience in using the ReacTable and multi-touch technology, have participated to the experiment. The subjects were collected through convenience sampling within the department of Audiovisual Technologies at the UPF (Universitat Pompeu Fabra). Each subject was exposed to both conditions (repeated measures), repeating the same tasks two times, one time per each input control method. Type of controls to be used first or second where counterbalanced according to the following criteria:

- Subjects with odd ID: session 1–Tangible, session 2–Multi-touch
- Subjects with even ID: session 1–Multi-touch, session 2–Tangible

3.3 Task Design

The tasks entailed to play two real-time tabletop video games, demanding for different specific skills in order to be successful in performance, i.e. *accuracy* and *precision* for Task 1, *precision* and *rapidity* for Task 2. After a short practice session, the subjects had ≈5 minutes to accomplish each task (i.e. ≈10 minutes for both); between the tasks, the subjects were allowed to rest for a time of ≈2-3 minutes. The tasks to be performed are described as follows:

- Task 1: *accuracy* and *precision* required; subjects had to intercept a virtual ball with a virtual paddle controlled through tangible or multi-touch input, then try to deflect the virtual ball towards a series of targets displayed in on the surface at random position in order to hit them. Their task was to hit as many targets as possible and always try to be as precise and as accurate as possible. Any time a target was hit, the score displayed at the bottom of the surface was increased, and a new target appeared at new random position. If the target was missed, an error was registered (i.e. missed shot + angular distance from the target) and a new virtual ball was generated for a new attempt; this process was continuing until the target was successfully hit. The size of the virtual ball was always 10px. Four different target size were used for the target to be hit with the ball, i.e. 10px, 15px, 25px, and 30px; three different displacement from the center of the surface on the X axis, i.e. 22px, 112px and 212px on both left and right side; four different displacement from the center of the paddle towards the top of the surface on the Y axis, i.e. 100px, 150px, 250px, 350px. Target position, size and displacement were randomly generated during the task for each subject (see Appendix C). A white circular line was delimiting the area in which the subjects

where free to move with the controls and eventually reduce the ID of 50px in both X and Y axis directions, the center of this area was displaced 218px on Y axis from the bottom of the surface. The subjects were not allowed to move the virtual paddle outside this area; if tired at wrist or at the forearm, they where eventually allowed to change their hands while playing. However, they were not allowed to interact with the controls using any kind of bimanual strategy.

- Task 2: *precision* and *rapidity* required; subjects had to intercept moving circular targets dropping down from the top of the surface, with a virtual paddle controlled through tangible or multi-touch input. Their task was to intercept as many circular targets as possible and always try to be as fast and as precise as possible. Any time a target was successfully intercepted, the score displayed on the sides of the surface was increased and the distance of the paddle from the ball at the moment of interception was registered (i.e. absolute distance from ball). Target size was always 15px. Three different target starting position from the top-center of the surface displaced on the X axis were used, i.e. 112px, 162px, 212px on both left and right side. Nine different speeds at which the circular targets moved toward the bottom of the surface, were divided for three different blocks of time during the 5 minutes of the task: 1) first 2 minutes, 600px/sec, 800px/sec and 900px/sec; 2) minutes 3-4, 1000px/sec, 1100px/sec, 1200px/sec; 3) last minute, 1300px/sec, 1400px/sec, 1500px/sec. Three different intervals of time at which the target appears were used, i.e. every 2sec, every 4sec and every 8sec. Target starting position, speed and interval of appearance were randomly generated during the task for each subject. A white cross was drawn at the center and displaced 218px on Y axis from the bottom part of the surface. Each time the subjects had performed interception movement (often sliding), whether they have intercepted or not the target, they were required to come back to the cross and keep a homing position until the next target appears. If tired at wrist or at the forearm, the subjects where eventually allowed to change their hands while playing. However, they were not allowed to interact with the controls using any kind of bimanual strategy.

3.4 Measures

- Pre-test questionnaire: demographics, tangible and multi-touch technology knowledge, ReacTable knowledge, video game playing frequency, video game playing enjoyment, preferred hand, junctions or eyes injuries or pathologies (see Appendix B, 1.1).

- Post-test questionnaire: usability test to asses comfort, ease of use and satisfaction of the control used, based on the Microsoft-Desirability Toolkit [5] (see Appendix B, 1.2).

- Post-test interview to assess additional qualitative information about user experience (see Appendix B, 1.3)

During the experiment, quantitative data regarding the gameplay were registered and plotted on a log file in real-time. These data were including several information about the gameplay and various parameters of both the two applications used in the two different tasks. Information plotted for Task 1 were: paddle rotation,

paddle position on Y and X axis, number of contact of the ball with the paddle, target size, target position on X and Y axis; in this case the paddle angular direction (Real Angle), the distance of the paddle from the target and the target size were used to calculate the target angular size (Target Angle) to subsequetly calculate the performance in Task 1 as follows:

- Task 1: Performance = TargetAngle – RealAngle. The closer to 0 is the value, the better the performance. Error value could be positive or negative according to if the target has been missed on its left (negative) or right (positive) side.

Information plotted for Task 2 were: paddle position on X and Y axis, paddle reaction time, paddle speed, target initial position (ballX), target speed (ball speed), paddle contact with the target (interception). While target speed was used to calculate the level of difficulty of the task in real-time, paddle position and and target position at the moment of interception were used to calculate the performance in Task 2 as follows:

- Task 2: Performance = - abs (Distance From Ball). The closer to 0 is the value, the better the performance. Error value could be only negative in which, whether the target was missed on left or right side, the error was always calculated as the absolute distance of the paddle from the ball.

Information on reaction time were used to perform an independent analysis on responsive performance of subjects with the use of the two controls, in order to obtain additional results and further investigate possible interesting outcomes.

4. RESULTS

We have compared performances of *accuracy-precision* and *precision-rapidity* in a tabletop RTI video game context, according to one group of subject exposed to two different conditions per two different tasks (see Figure 34): TUI, Accuracy-Precision (T-AP); TUI Precision-Rapidity (T-PR); Multi-Touch, Accuracy-Precision (MT-AP); Multi-Touch, Precision-Rapidity (MT-PR). The data were collected through a log file implemented into the application programmed for the experiment (see Appendix C).

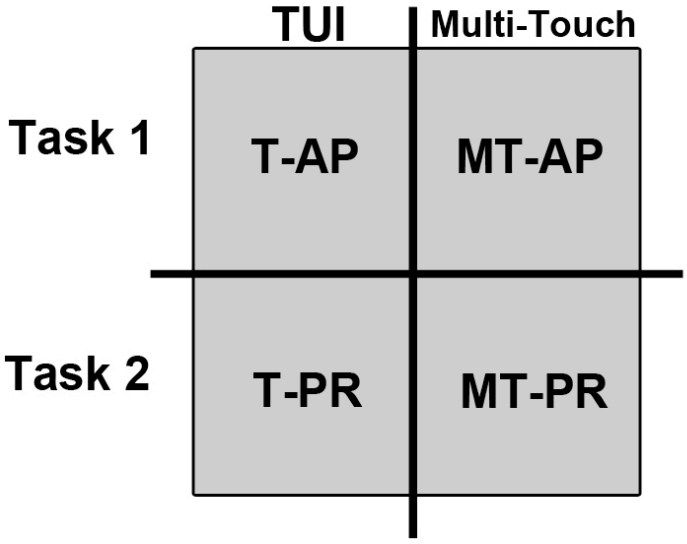

Figura 34: Experimental groups defined by conditions and tasks

Additional qualitative information on usability and user experience were gathered through post-test and interview, according to the methodology shown in [5] (see Appendix A). The data extracted by the log file were calculated and transformed in Python 2.7, then processed using SPSS for the statistical analysis of both performances in Task 1 and Task 2. Due to not total completion of the experiment, subjects 4 and 10 have been filtered out from the final analysis of Task 1 and subjects 6 and 12 have been filtered out from the final analysis of Task 2, actually decreasing the number of useful cases in both experiments from 12 to 10. Hence, a paired-samples t-test was performed to compare means of the dependent variables within samples in Task 1, showing significant difference between tangible and multi-touch conditions. A Wilcoxon test was performed to compare means of the dependent variables and reaction time within samples in Task 2, showing significant difference between tangible and multi-touch conditions. The effect of target size, target distance and target angle on the dependent variable in Task 1 was evaluated by applying a Pearson correlation, showing a strong significant correlation only between performance and target angle, a weak significant correlation between performance and target size and no significant correlation with target distance. The effect of target starting position (ball X), target speed and difficulty on the dependent variable in Task 2 was evaluated by applying a Spearman correlation, showing significant correlation between performance and all the three variables. Difficulty in Task 2 was calculated as: initial distance[ball X]*target speed. Finally, correlaiton analysis between performance and variables deriving from demographic test failed to show any significant effect on the dependent variables.

4.1 Task 1: Accuracy-Precision

Figure 35 shows the results of Task 1. A paired-samples t-test (two-tailed) was conducted to compare mean performances of each subject expressed as *accuracy-precision* in tangible and multi-touch conditions. There was a significant difference in performance between tangible (M=0.49, SD=1.17) and multi-touch control (M=-0.94, SD=1.71) conditions in favor of tangible control; (t(9)=2.68, p<0.05) (see Appendix A, 1.1, table 2, 3). A one-way ANOVA was conducted to compare the effect of controls on performance for tangible and multi-touch conditions. There was a significant effect of controls on performance for two conditions (F(1, 1625)=19.57, p<0.01) (see Appendix A, table 4).

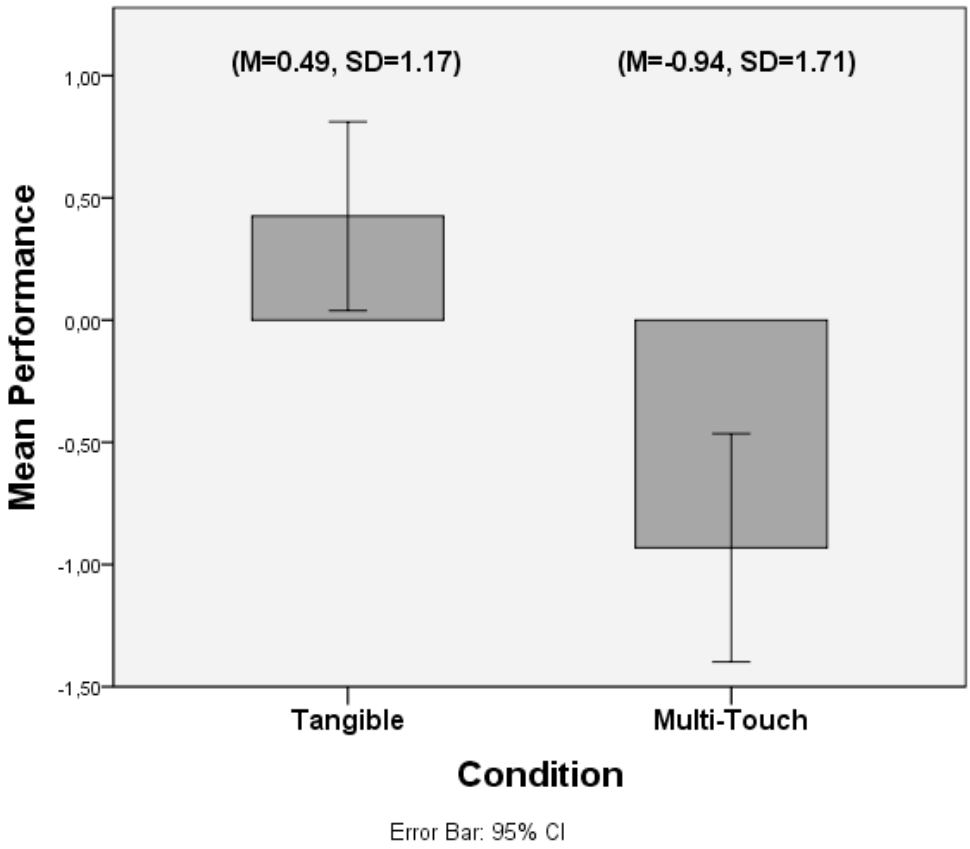

Figura 35: Mean performance for the same group under Tangible and Multi-touch condition. A significant difference was found between the two conditions (t(9)=2.68, p<0.05). An ANOVA between the two groups also revealed significant difference (F(1, 1625)=19.57, p=<0.01)

4.1.1 Correlation analysis between Performance and Target Size, Target Distance, Target Angle

A Pearson product-moment correlation coefficient was computed to assess the relationship between performance and target size, target distance and target angle (see Appendix A, 1.1.1, table 5). There was a weak positive correlation between overall conditions performance and target size (r=0.06, n=1627, p<0.01); a significant strong negative correlation was found between overall conditions performance and target angle (r=-0,61, n=1627, p<0.01), where the increasing of target distance and its decreasing in size, generally led to a decreasing in performance (see Figure 36). No significant correlation was found between performance and target distance (p=n.s.).

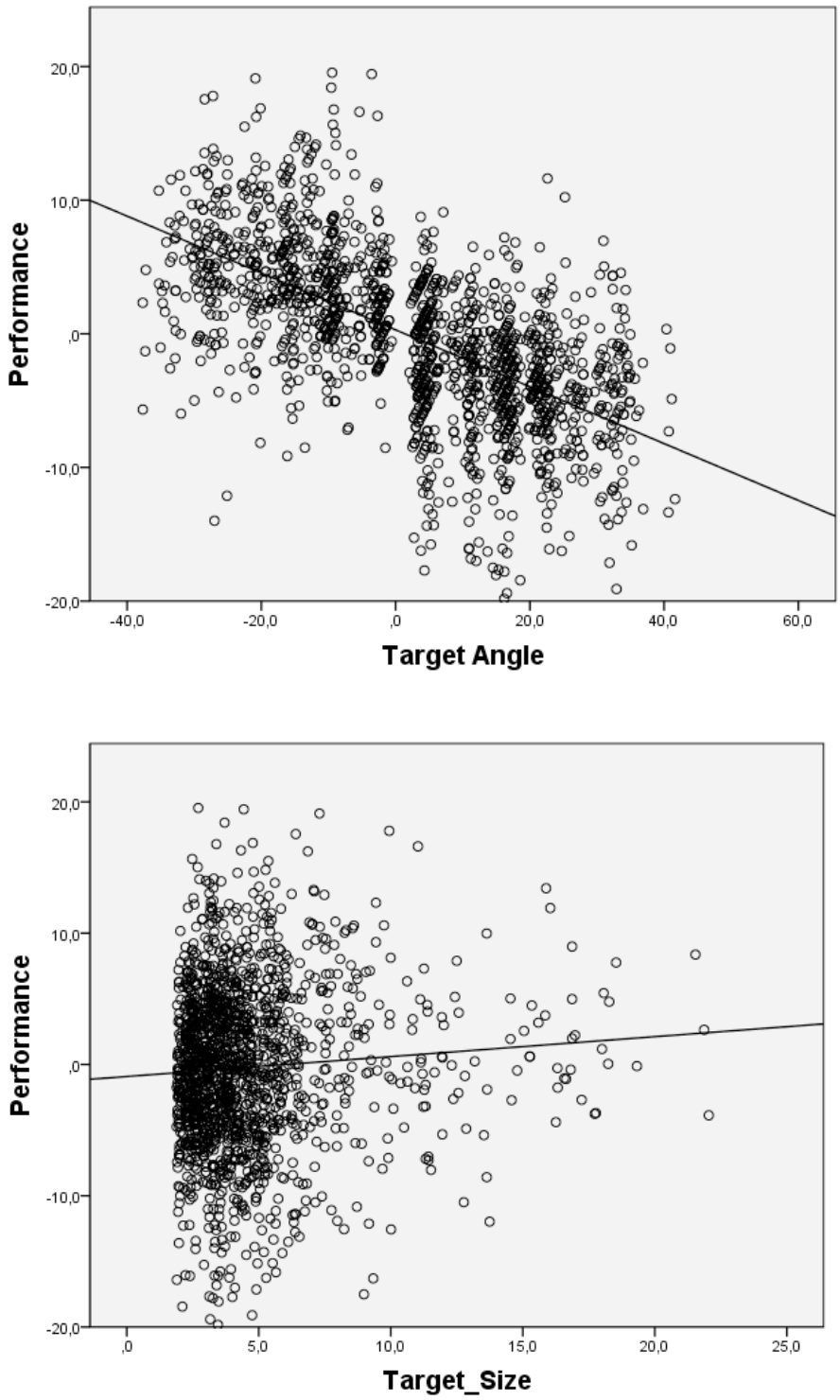

Figura 36: Pearson's Correlation between Performance over Target Size (bottom) and Target Angle (up)

Independent correlation analysis for tangible and multi-touch conditions revealed a stronger negative correlation between performance and target angle in tangible condition ($r=-0.71$, $n=853$, $p<0.01$) than in multi-touch condition ($r=0.52$, $n=774$, $p<0.01$); no significant correlation was found with target size and target distance

(p=n.s.) (see Appendix A, 1.1.1, table 6). A scatterplot summarizes the results (see Figure 37).

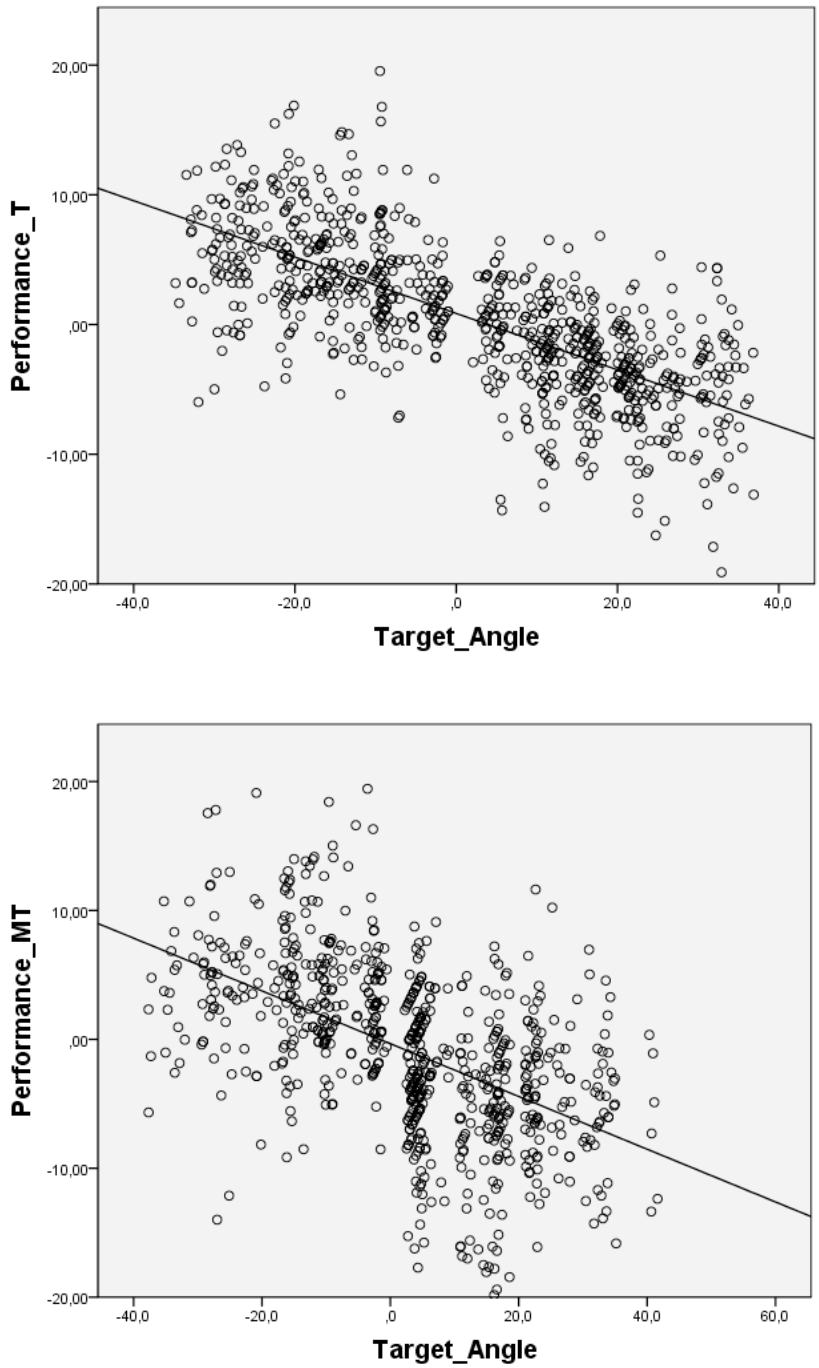

Figura 37: Pearson's Correlation between Performance and Target Angle in Tangible (up) and Multi-Touch (bottom) conditions

4.2 Task 2: Precision-Rapidity

A Wilcoxon Signed Ranks test was conducted to compare mean performance for each subject expressed as *precision-rapidity* in Task 2 between tangible and multi-touch conditions. The test showed that there was a significant difference in performance

between tangible (M=-57.8, SD=59.48) and multi-touch control (M=-63.56, SD=59.44) conditions in favor of tangible control; (Z=-1.98, *P*<0.05) (see Appendix A, 1.2, table 7, 8). The results of Task 2 are showed in Figure 38.

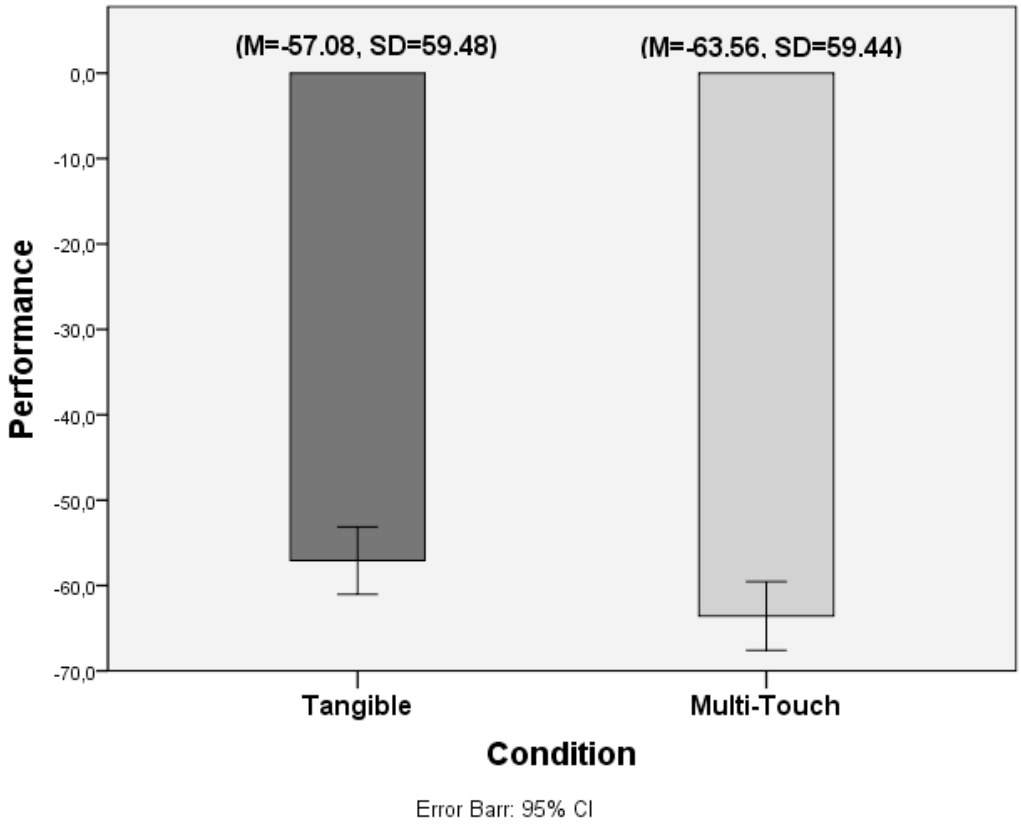

Figura 38: Mean performance for the same group under Tangible and Multi-touch condition. A significant difference was found between the two conditions (Z=-1.98, *P*<0.05)

4.2.1 Correlation analysis between Performance and Target Speed, Target Starting Position, Difficulty.

A Spearman Rho correlation coefficient was computed to assess the relationship between performance and target speed, target starting position (ball X) (see Appendix A, 1.2.1, table 9). There was a weak positive correlation between overall conditions performance and target starting position (r=0.12, n=1714, p<0.01), the result is summarized in Figure 39. A significant strong negative correlation was found between overall conditions performance and target speed (r=-0.68, n=1714, p<0.01), where a higher target speed corresponded in a lower performance (see Figure 40). Independent correlation analysis for tangible and multi-touch conditions showed a slightly stronger negative correlation between performance and target speed in tangible condition (r=-0.68, n=870, p<0.01) than in multi-touch condition (r=-0.67, n=844, p<0.01) (see Appendix A, 1.2.1, table 10).

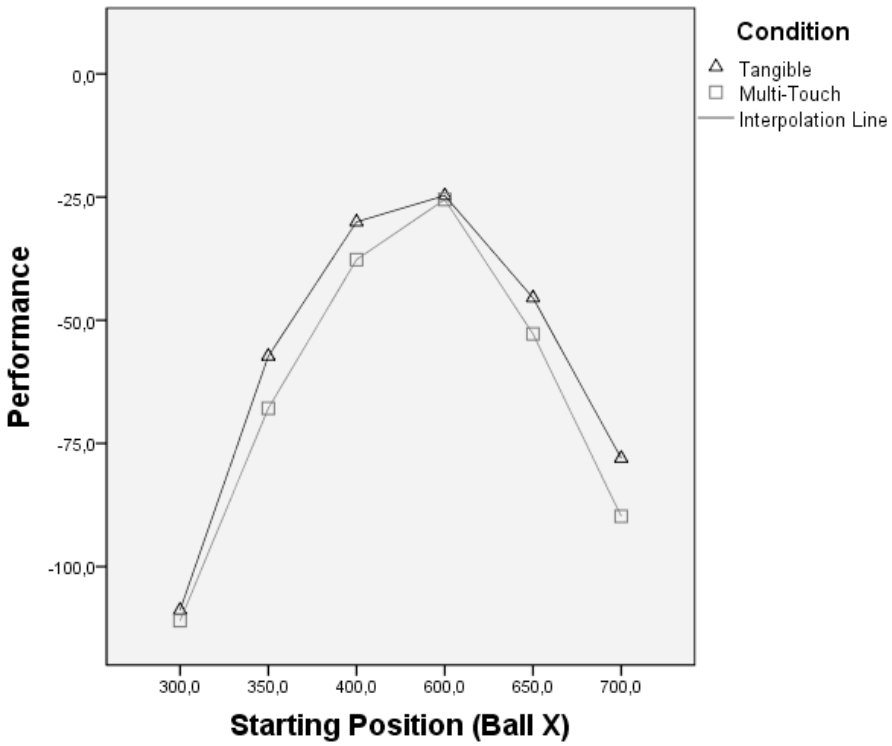

Figura 39: Spearman's Correlation bewteen performance and ball starting position. The plot shows the performance in the two conditions, where is possible to see how tangible generally outperformed multi-touch. Also, the plot shows the difference in performance between the extreme left area of the surface (300) and the rightmost one (700), where a better performance was produced in the latter. This is probably due to a majority of right handed subjects, where the effect of the performance in the contralateral area was better than in the ipsilateral one.

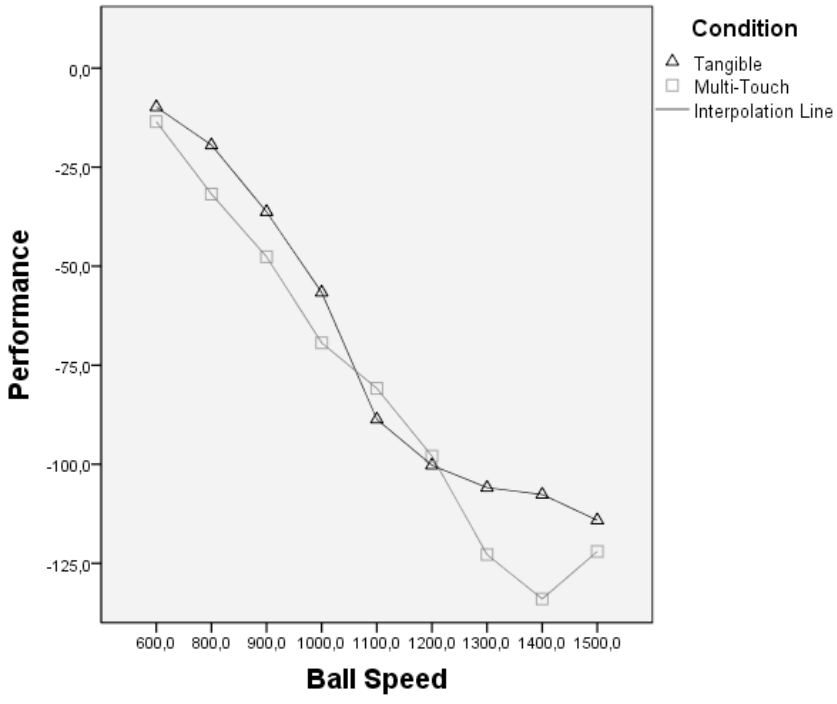

Figura 40: Spearman's Correlation between Performance and Target Speed

4.2.2 Reaction Time

An additional analysis was carried out on mean reaction time for both tangible and multi-touch conditions; the results are shown in Figure 41. A Wilcoxon Signed Ranks test was performed finding significant difference in mean reaction time between tangible (M=476.32, SD=24.77) and multi-touch control (M=483.73, SD=22.24) conditions in favor of tangible controls; (Z=-1.98, *P*<0.05) (see Appendix A, 1.2.2, table 11).

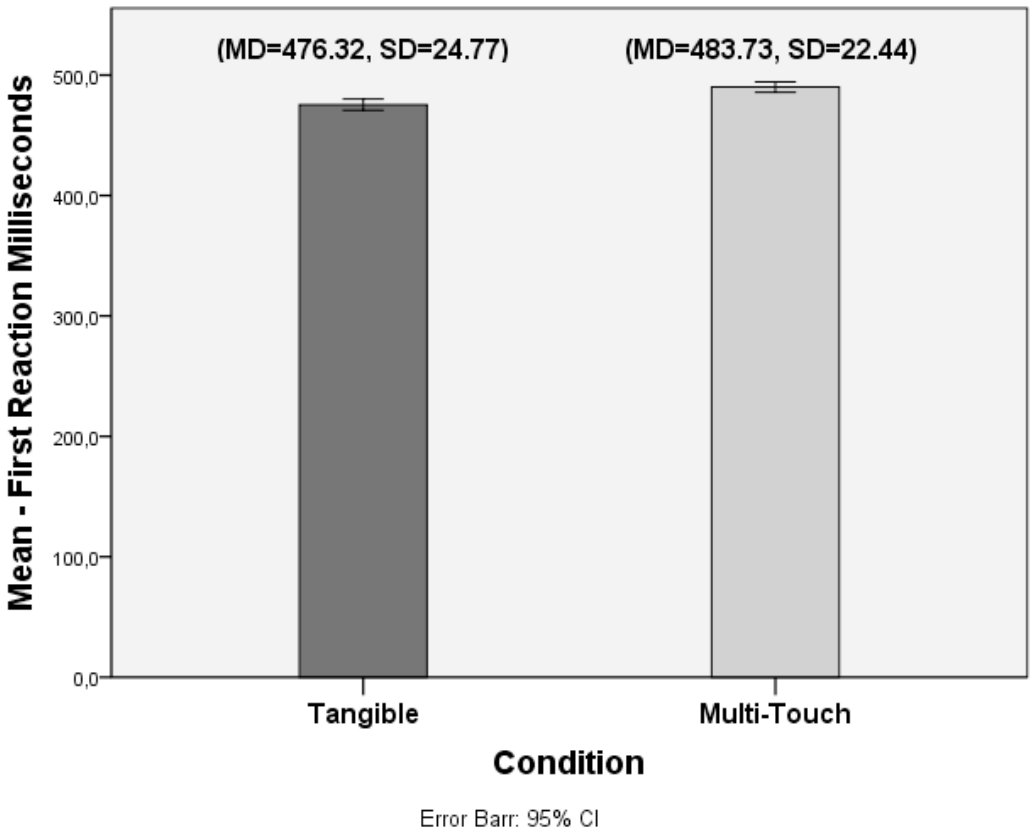

Figura 41: Mean reaction time for the same group under Tangible and Multi-touch condition. A significant difference was found between the two conditions (Z=-1.98, *P*<0.05)

A Spearman correlation analysis was performed to assess relationship between reaction time and target speed, target starting position (ball X) (see Appendix A, 1.2.3, table 12). A moderate negative correlation was found between overall conditions reaction time and target speed (r=-0.26, n=1714, p<0.01), where a higher target speed elicited a shorter reaction time; a weak negative correlation was found between reaction time and target starting position (ball X) (r=-0.08, n=1714, p<0.01); the scatterplot summarizing these results are shown in Figure 42.

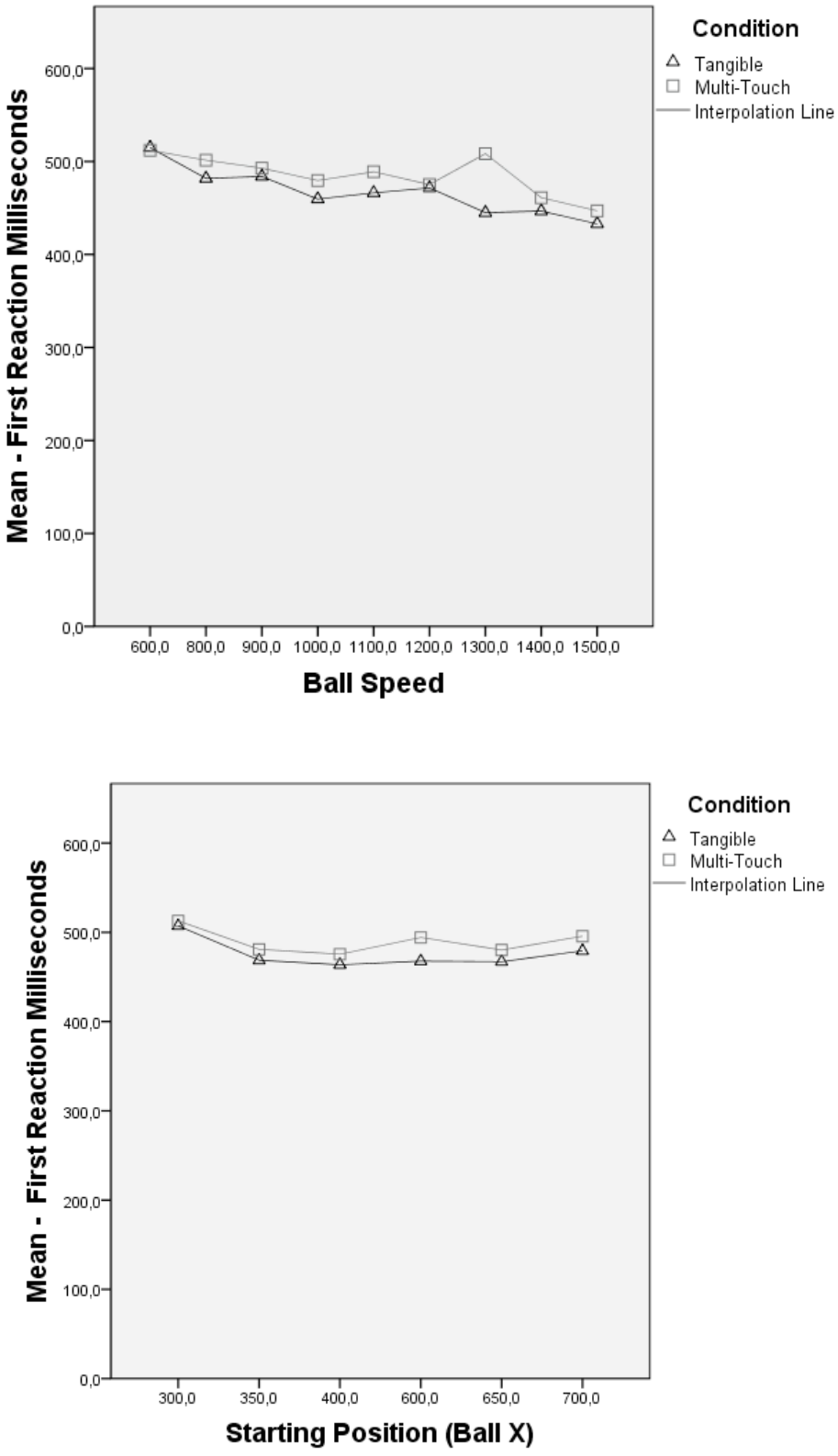

Figura 42: Spearman's Correlation analysis for Mean Reaction Time over Target Speed (up) and Target Starting Position (bottom)

4.3 Post-Test

Words selected by participants using the readapted version of the Microsoft Desirability Toolkit [5], were used to generate word clouds that provide a visual overview of participant reactions, to the use of both tangible and multi-touch controls during the experiment (see Appendix A). The bigger the word and the darker the degree of black displayed, the more an adjective has been preferred and the more frequently it has been selected. After having checked their preferred adjectives, the subjects were asked to indicate the three most representative for them amongst the chosen ones, then briefly interviewed on these three to asses more deep personal opinions and impressions about the controls. The results of these tests and the results of the interview are explained in the following points.

4.3.1 TUI evaluation

The subject's most selected adjectives to describe their experience with the use of tangible controls were *controllable* and *easy to use* (see Appendix A, 1.3.1, Figure 43). This results are consistent with the performance produced by subjects with tangibles in both tasks, where in Task 1 they felt more secure in controlling the angular rotation of the virtual paddle with the puck than with the their fingers, and in Task 2 they suffered less with the puck the friction produced by rapidly sliding the fingers on the surface when trying to intercept a moving target. Also, the better tracking technology provided by reacTIVision software for fiducial markers, had indeed influenced subjects evaluation of the controls in favor of tangibles. These implications will be discussed more in detail in point **5**.

4.3.2 Multi-Touch evaluation

Multi-touch control has been generally regarded as more uncontrollable and frustrating with respect to tangible control, where this negative opinion may be partially due to the tracking issues aforementioned. However, the adjectives that subjects have most selected to describe their experience with multi-touch were *accessible* and *intuitive* (see Appendix A, 1.3.2, Figure 44). This is a positive feedback in favor of this input technology, that witnesses how multi-touch is becoming always more familiar thanks to the increasing diffusion of portable touch-screen devices in the contemporary society. This result goes partially in contrast with the qualitative results produced by Lucchi et al in [86], where multi-touch was still regarded as a less intuitive input technology with respect to tangible controls.

4.3.3 Interview

Different trends have been detected by analyzing the interviews, and the tangible control have generally received more positive feedback than the multi-touch. For instance, more than one subject has defined the gameplay and the interaction under tangible condition as more entertaining and fun, feeling that they achieved a better performance with this type of control (see Appendix A, 1.3.3), which is actually consistent with the analysis of the quantitative data. In particular, subject 9 commented his experience with tangible controls, saying that he was "motivated to play, because the characteristics of the TUI were coherent with the movement of the hand and the virtual objects, therefore making the goal of the game clearer an the gameplay more entertaining" (see Appendix A, 1.3.3). However, when using the tangible control in Task 2, some users gave negative comments related with the latency between the performed movement and the visual feedback, as in the case of subject 11 where he said

that “the latency was frustrating me, in which I felt the control was not responsive with fast movements” (see Appendix A, 1.3.3). This problem was indeed related with the design of the application as well, which was sometimes demanding for movements too fast to be optimally detected by the tracking systems. However, this choice was taken on purpose to investigate the limits of the technology used, as well as trying to assess a tradeoff between fun and optimal performance within the gameplay, in which testing controls limitations and stress.

More negative feedback were provided from the subjects regarding the use of multi-touch controls. This control method have shown to elicit more complicated issues in control than tangible control method, as for example the problem related with one’s personal capacity to bend the fingers correctly to comfortably keep the contact with the surface. Subject 3 infact, commented his experience with this control saying that “it wasn’t feeling comfortable to try to find the optimal position with fingers all the time, while trying to keep concentrated on the gameplay at the same time” (see Appendix A, 1.3.3). This problem was certainly emphasized by the tracking system, who was not responding always optimally, as in the case of subject 1, who described the consequence of the latency problem as “making the control of the virtual object not reliable and not stable” (see Appendix A, 1.3.3). However, some positive comments have been given to this control in terms of intuitivity and approachability, where some users have defined the control as straightforward and very intuitive. For instance, subject 8 defined multi-touch as “an intuitive control, which doesn’t need particular explanation or instructions to be immediately understood and used” (see Appendix A, 1.3.3). Subject 9 commented this control method saying that “it is a nice control, in which it gives you the possibility to choose your own way to interact, an it is also very direct and easy to acquire” (see Appendix A, 1.3.3)

5. DISCUSSION & FUTURE WORK

5.1 Discussion

In this report we have presented a comparative analysis of TUI and multi-touch input control methods, within the context of a real-time tabletop videogame interaction. Through this design we wanted to investigate the effect of using both controls with this type of real-time applications and assess their performances, as well as retaining qualitative data regarding user experience, usability and satisfaction of the controls.

The study has revealed significant difference in performance between tangible and multi-touch controls under RTI video game for two different type of tasks, suggesting that tangible controls were more reliable when accurate and precise gestures were needed, and that they were also allowing for better responses when rapid and precise movements were required. These findings are patially confirming the results of previous works, which were testing these controls in non RTI context, and where tangible controls have generally outperformed multi-touch in manipulation and acquisition tasks [86, 114].

Overall, our results from the t-test and the analysis of variance in Task 1 suggested that tangible control was allowing subjects to perform more accurately and precisely in ball deflection and target hit task. This could be due to different reasons: 1) because of the robustness of a physical widget, such as a tangible puck, which is more consistent with angular rotation and fine-grained angular adjustements than multi-touch finger control, since it does not rely on how one is able to keep the fingers firmly in contact with the surface; 2) because of the more optimal tracking system for fiducial markers provided by reacTIVision, which probably gave to the subjects more security about their actions; 3) because of the fingers natural trembling and arm fatigue, which probably made difficult to keep a specific fixed position to perform accurate directional adjustments, in order to find the right angle to gradually reach the target. The quantitative results were reflecting the subject's comments in qualitative analysis, where most of them regarded tangible control under Task 1 as generally more reliable, stable and controllable, reporting haptic feedback of the tangible as a good feature, eliciting better sensation in control and giving more security about one's actions and performance (see Appendix A, 1.3.3).

The results of Task 2 also showed an advantage of tangible over multi-touch. This could be partially due to the aforementioned tracking implications, as well as to reasons more pragmatically related with the interaction between surface material and physical characteristics of the controls. This task was actually designed with the aim of balancing the characteristics of both controls, where multi-touch could compete with tangible in a more equal way, since not particularly accurate gestures were required. The results showed that the design partially succeded to achieve a balance between the two controls, since the mean difference of performance in Task 2 was only -0.6 in favour of tangible, and not almost the double as in Task 1 (see Appendix B). However, the outperforming of tangible in Task 2 can be addressed to the following reasons: 1) the rapid sliding of fingers on the acrylic surface could have produced a tiresome sensation in multi-touch condition after a while, which is confirmed by the comments reported by the subjects in the interview; 2) since tangibles are more eye-free controls, in which they provide haptic feedback that could partially disentangle subjects from the visual feedback of the application, it is possible that the visual stimuli could have been

perceived more fastly while using tangible than multi-touch, explaining then a better mean RT (Reaction Time) produced by tangible in this task; 3) the concurrence of latency in the tracking system at high speeds and the sometimes extreme difficulty of the application could have frustrated the subjects under the multi-touch conditions more than under the tangible one; however, some negative comments are reported in this case for both controls in the interview. One good point emerging from information gathered after this task through post-test and interview was that, when not many cumbersome and complicated gestures or manipulation are required while using multi-touch, this control is regarded as very intuitive and highly approachable (see Appendix A, 1.3.3).

Finally, the correlation analysis performed for both tasks have evidenced the effect of certain variables over the performance. While parameters as target size and target distance failed to show a strong correlation with performance in Task1, the target angle has shown to strongly affect the dependent variable, confirming the importance of using a control which could allow for more accurate fine-grained manipulation and accurate rotation, in order to be successful in performance. Also, independent correlation analysis for tangible and multi-touch revealed a stronger dependency of the former to target angle, suggesting that performance under tangible condition was more concentrated on gradual reaching of the right angle from a fixed point, whereas multi-touch might have been more dispersive and less consistent. In this case tangible controls have preduced less constant error in detecting the right angle to hit the target then multi-touch control, with a performance that was generally more accurate and precise. Strong relationships were also find between performance, target speed and difficulty in Task2. However, independent analysis for tangible and multi-touch did not reveal a big difference between the two conditions, suggesting that the difference in strategy when using these controls in Task 2 was minimal.

5.2 Faced problems and future works

Different challenges and problems have been faced during this study, that might eventually guide future works in the field of research. The first one was comparing tangible and multi-touch controls under RTI conditions and constraints, for the control of a tabletop video game application. In order to be able to obtain consistent results, the test had used strictly quantifiable measurements under time constraint, hence partially reducing the fun generated by the interaction with the interface. However, with this method we have been able to detect precise outcomes in the use of these controls in real-time context, where findings are generally suggesting that tangibles are less prone to mistakes and more precise in manipulations with respect to multi-touch also for video game control. However, since qualitative analysis showed a good preference of subjects towards multi-touch control in terms of intuitivity and accessibility, the use of this technology in RTI context needs to be more deeply investigated. Indeed, part of the results produced were subject to the limits in the design of the application, which is a recurrent problem in task oriented interaction, and also influenced by the technology used for fiducial and finger tracking. For instance, specific limitations were found in the reacTIVision finger tracking system under certain conditions, such as latency or loss of detection due to changes in fingers position. Especially when testing different subjects, the system had to be constantly recalibrated, so as to adapt to the finger's size of the single person. Besides this drawbacks, reacTIVision is indeed very convenient to be used to implement both tangible and multi-touch controls, in which it is an easy software to manage and not costfull in terms of computer memory [4]. One suggestion

to solve the aforementioned tracking problems, would be the comined use of a better software for finger tracking and reacTIVision for fiducial tracking, or the improvement of finger tracking system in the reacTIVision software by rewriting part of the source code. Furthermore, some problems were found regarding the action to be performed and surface material, which sometimes produced a non comfortable sensation in fastly sliding the fingers on the surface due to friction. This problem might be solved by using an *anti-stiction* coating surface in lieu of the classical ReacTable one, to facilitate glides and similar gestures through a reduced friction. Regarding the analysis of the data, we propose to integrate and readapt the Fitts's law [35] to our model, in order to create an ID that can give more consistency to the final results. The use of a readapted version of the Microsoft Desirability Toolkit [5] strongly helped in detecting those impalpable aspects of the user experience, that provided fundamental insights and suggestions for the interpretation of the quantitative results. We would recommend the use of this questionnaire to gather qualitative information even in other studies.

6. CONCLUSION

In this project we have presented results on comparative analysis of tangible and multi-touch input control methods in two different performances, i.e *accuracy-precision* and *precision-rapidity*, under RTI conditions and constraints. Our work was motivated by the desire to test the response of these two control methods within a real-time tabletop video game context which, surprisingly, has not been receiving particular attention from the scientific community so far. Most of the focus on comparative analysis between these two control methods, is still influenced by the WIMP paradigm and related to issues like optimal target aquiring and selection, etc.

The analysis of the data has shown that TUIs are providing more reliability and consistency in performing accurate and precise manipulations than multi-touch also in this interaction context. However, this study represents just a first step, that might lead the research in the field of input control methods to allow for more attention towards RTI paradigm, especially for what concerns the use of tangible and multi-touch technologies as mean to control applications other than modern readaptation of classical WIMP GUIs (as in the case of iPad for instance). This could represent a great chance to investigate more deeply the potential of this control methods, as well as exploiting different types of "non conventional" gestures. Surely, one of the limits presented by our experiment was the constraining of multi-touch interaction only to unimanual strategy. However, it is in our will for future work to test these controls in RTI context, allowing subjects to freely choose between a unimanual and a bimanual interaction strategy.

REFERENCES

1. Accot, J., and S. Zhai. 1997. ‘‘Beyond Fitts’ Law: Models for Trajectory-Based HCI Tasks.’’ In: *Proceedings of the 1997 ACM Conference on Human Factors in Computing Systems*. New York: ACM Press, pp. 295–302.
2. Accot, J., and S. Zhai. 1999. ‘‘Performance Evaluation of Input Devices in Trajectory-Based Tasks: An Application of the Steering Law.’’ In: *Proceedings of the 1999 ACM Conference on Human Factors in Computing Systems*. New York: ACM Press, pp. 466–472.
3. Accot, J., and Zhai, S. (1997). Beyond Fitts' law: models for trajectory-based HCI tasks. *In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '97)*. ACM, New York, NY, USA, pp. 295-302.
4. Bencina, R., Kaltenbrunner, M., and Jorda, S. (2005). Improved Topological Fiducial Tracking in the reacTIVision System. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* - Workshops - Volume 03 (CVPR '05), Vol. 3. IEEE Computer Society, Washington, DC, USA, pp. 99-105.
5. Benedek, J., and Miner, T. (2002). Measuring Desirability: New methods for evaluating desirability in a usability lab setting. In: *Proceedings of UPA Usability Professional Association*. Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052.
6. Benko, H., Wilson, A.D., and Baudisch, P. (2006). Precise selection techniques for multi-touch screens. In: *Proceedings of the SIGCHI conference on Human Factors in computing systems* (CHI '06), Rebecca Grinter, Thomas Rodden, Paul Aoki, Ed Cutrell, Robin Jeffries, and Gary Olson (Eds.). ACM, New York, NY, USA, pp. 1263-1272.
7. Bérard, F. (2003). The Magic Table: Computer-Vision Based Augmentation of a Whiteboard for Creative Meetings. In: *Proceedings of IEEE International Conference in Computer Vision.*
8. Block, F., Gutwin, C., Haller, M., Gellersen, H., and Billinghurst, M. (2008). Pen and paper techniques for physical customisation of tabletop interfaces. *2008 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*, pp. 17-24.
9. Brouwer, A. M., Brenner, E., and Smeets, J. B. (2000). Hitting moving objects. The dependency of hand velocity on the speed of the target. *Experimental Brain Research*, 133(2), pp. 242–248.
10. Brown, E., Buxton, W. and Murtagh, K. (1990) Windows on tablets as a means of achieving virtual input devices. In D. Diaper et al. (Eds), *Human-Computer Interaction - INTERACT '90.* Amsterdam: Elsevier Science Publishers B.V. (North-Holland), pp. 675-681.
11. Brown, J.S., Collins, A., and Duguid, P. (1988). Situated Cognition and the

culture of learning. *IRL Report*, No. IRL 88–0008.

12. Buxton, B. (2010). A Touching Story: A Personal Perspective on the History of Touch Interfaces Past and Future. *Information Display*, 41(May), pp. 444-448.
13. Buxton, W. (1983). Lexical and pragmatic considerations of input structures. *SIGGRAPH Comput*. Graph. 17, 1 (January 1983), pp. 31-37.
14. Buxton, W. (1986) There's More to Interaction than Meets the Eye: Some Issues in Manual Input. In Norman, D. A. and Draper, S. W. (Eds.), (1986), *User Centered System Design: New Perspectives on Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, pp. 319-337.
15. Buxton, W. (1986). There's More to Interaction than Meets the Eye: Some Issues in Manual Input. In Norman, D. A. and Draper, S. W. (Eds.), (1986), User Centered System Design: *New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates.
16. Buxton, W. (1987). ‘‘The Haptic Channel.’’ In R. M. Baecker and W. A. S. Buxton, eds. *Readings in Human-Computer Interaction: A Multidisciplinary Approach.* San Mateo, California: Morgan Kaufmann, pp. 357–365.
17. Buxton, W. (1994). Combined keyboard / touch tablet input device. XEROX *Disclosure Journal*, 19(2), March/April 1994, pp.109-111.
18. Buxton, W. (2008). Multi-touch systems that i have known and loved http://www.billbuxton.com/multitouchOverview.html
19. Buxton, W. (in progress). Gesture Based Interaction. *Unfinished book manuscript*.
20. Buxton, W. (in progress). Human Input to Computer Systems: Theories, Techniques and Technology. *Unfinished book manuscript*.
21. Buxton, W. (in progress). Some Representative 2D Tasks. *Unfinished book manuscript.*
22. Buxton, W. and Myers, B. (1986). A study in two-handed input. In: *Proceedings of CHI '86*, pp. 321–326.
23. Buxton, W., Hill, R. and Rowley, P. (1985). Issues and techniques in touch-sensitive tablet input, *Computer Graphics*, 19(3), In: *Proceedings of SIGGRAPH'85*, pp. 215–223.
24. Buxton,W. (2008). The Long Nose of Innovation. http://www.businessweak.com/innovate/content/jan2008/id2008012_297369.htm
25. Caeyenberghs, K., Wilson, P. H., Van Roon, D., Swinnen, S. P., and Smits-Engelsman, B. C. M. (2009). Increasing convergence between imagined and executed movement across development: evidence for the emergence of movement representations. *Developmental Science*, 12(3), pp. 474-483.
26. Card, S.K., English, W.K., and Burr, B. J. (1978). Evaluation of mouse, rate-controlled isometric joystick, step keys and text keys for text selection on a CRT. *Ergonomics*, 21(8), pp. 601-613.

27. Costanza, E. and Robinson, J.A. (2003). A region adjacency tree approach to the detection and design of fiducials, In: *Vision, Video and Graphics (VVG)*, pp. 63–70.

28. De Lussanet, M. H. E., Smeets, J. B. J., and Brenner, E. (2001). The effect of expectations on hitting moving targets: influence of the preceding target's speed. *Experimental Brain Research*, 137(2), pp. 246–248.

29. Dietz, P., Leigh, D. (2001). DiamondTouch: A Multi-user Touch Technology. *In: Proceedings of the 14th Annual Symposium on User Interface Software and Technology*. UIST 2001, ACM, New York, NY, USA, pp. 219-226.

30. Dourish, P. (2000). Embodied Interaction : Exploring the Foundations of a New Approach to HCI. Work, *HCI in the (HCI in the New Millennium)*, pp. 1-16.

31. Dourish, P. (2004). Where the action is: the foundations of embodied interaction. Cambridge, MIT Press.

32. Drewes, H. (2010). Only one Fitts' law formula please!. In: *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems* (CHI EA '10). ACM, New York, NY, USA, pp. 2813-2822.

33. Fishkin, K.P. (2004). A taxonomy for and analysis of tangible interfaces. *Personal Ubiquitous Comput.* 8, 5 (September 2004), pp. 347-358.

34. Fishkin, K.P. (2004). A taxonomy for and analysis of tangible interfaces, *Personal and Ubiquitous Computing*, Vol. 8, No. 5, pp.347-358.

35. Fitts, P.M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, volume 47, number 6, June 1954, pp. 381–391. (Reprinted in *Journal of Experimental Psychology*: General, 121(3), pp. 262–269, 1992).

36. Fitzmaurice, G.W., Ishii, H. and Buxton, W. (1995) Bricks: Laying the Foundations for Graspable User Interfaces, In: *Proceedings of CHI'95*, pp. 442-449.

37. Fjeld, M. , Bichsel, M. and Rauterberg, M. (1998) BUILD-IT: An Intuitive Design Tool Based on Direct Object Manipulation. In I. Wachsmut and M. Frölich (eds.): *Gesture and Sign Language in Human-Computer Interaction*, Lecture Notes in Artificial Intelligence, Vol. 1371, Berlin: Springer-Verlag, pp. 297-308.

38. Foley, J.D., Wallace, V.L., and Chan, P. (1984). The human factors of computer graphics interaction techniques. *IEEE Comput. Graph. Appl.* 4, 11 (November 1984), pp. 13-48.

39. Frazer, J. (1995). An Evolutionary Architecture. *Themes VII*. London: Architectural Association.

40. Ganser, C., Kennel, T., Birkeland, N., and Kunz, A. (2005). Computer-supported Environment for Creativity Processes in Globally Distributed Teams. In: *Proceedings of the International Conference on Engineering Design ICED 2005*, pp. 109-110.

41. Graham, T.C.N.,. Watts, L.A. Calvary, C., Coutaz, J., Dubois, E., and Nigay, L. (2000). A dimension space for the design of interactive systems within their physical environments. *In Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques* (DIS '00), Daniel Boyarski and Wendy A. Kellogg (Eds.). ACM, New York, NY, USA, pp. 406-416.

42. Grossman, T. and Balakrishnan, R. (2005). A probabilistic approach to modeling two-dimensional pointing. *ACM Trans. Comput.-Hum. Interact.* 12, 3 (September 2005), pp. 435-459.

43. Guiard, Y. (1987). Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain Model. In: *Journal of Motor Behavior*, 19(4), pp. 486-517.

44. Guiard, Y. (2001). Disentangling relative from absolute amplitude in Fitts' law experiments. In: *CHI '01 extended abstracts on Human factors in computing systems* (CHI EA '01). ACM, New York, NY, USA, pp. 315-316.

45. Guiard, Y. (2009). The problem of consistency in the design of Fitts' law experiments: consider either target distance and width or movement form and scale. In: *Proceedings of the 27th international conference on Human factors in computing systems* (CHI '09). ACM, New York, NY, USA, pp. 1809-1818.

46. Guiard, Y., M. Beaudouin-Lafon, and D. Mottet. 1999. ‘‘Navigation as a Multiscale Pointing Extending Fitts’ Model to Very High Precision Tasks.’’ In: *Proceedings of the 1999 ACM Conference on Human Factors in Computing Systems*. New York: ACM Press, pp. 450–457.

47. Han, J.Y. (2005). Low-cost Multi-touch Sensing through Frustrated Total Internal Reflection. In: *Proceedings of UIST 2005*; ACM, New York, NY, USA, pp. 115-118.

48. Hartson, H. R., and Gray, P. (1992). Temporal Aspects of Tasks in the User Action Notation. *Human-Computer Interaction*, 7(1), Taylor and francio, pp. 1-45.

49. Herot, C., Weinzapfel, G. (1978). One-Point Touch Input of Vector Information from Computer Displays, *Computer Graphics*, 12(3), pp. 210–216.

50. Hilliges, O., Butz, A., Izadi, S., and Wilson, A. D. (2010). Interaction on the Tabletop: Bringing the Physical to the Digital. In: *Tabletops-Horizontal Interactive Displays*, Springer, pp. 189–221.

51. Hinckley, K. (2006). Input Technologies and Techniques, In: *Handbook of Human-Computer Interaction*, ed. by A. Sears and J. Jacko.

52. Hinckley, K., Pahud, M., and Buxton, B. (2010). Direct Display Interaction via Simultaneous Pen + Multi-touch Input. *Society for Information Display (SID) Symposium Digest of Technical Papers*, May 2010, Volume 41(1), Session 38, pp. 537-540.

53. Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H. and Buxton, B. (2010). Manual Deskterity : An Exploration of Simultaneous Pen + Touch Direct Input. In: *Proceedings of*

the 28th International Conference Extended Abstracts on Human Factors in Computing Systems, CHI'10 (alt.chi), pp. 2793 – 2802.

54. Hofer, R., Kunz, A. (2009). TNT: Touch 'n' TUI on LC-Displays. In: *Proceedings of the 8th International Conference on Entertainment Computing ICEC '09*, Paris, France, pp. 222-227.

55. Hofer, R., Kunz, A., and Kaplan, P. (2008). MightyTrace: Multiuser Tracking Technology on LC-Displays. In: *Proceedings of CHI 2008,* ACM, New York, NY, USA, pp. 215-218.

56. Hofer, R., Naeff, D., and Kunz, A. (2009). FLATIR: FTIR Multi-touch Detection on a Discrete Distributed Sensor Array. In: *Proceedings of the Third International Conference on Tangible and Embedded Interaction TEI '09*, ACM, New York, NY, USA, pp. 317-322.

57. Holman, D. (2007). Gazetop: interaction techniques for gaze-aware tabletops. Group (p. 1660). ACM.

58. Holmquist L, Redstro, J, Ljungstrand, P. (1999). Token-based access to digital information. In: *Proceedings of the 1st inter- national symposium on handheld and ubiquitous computing (HUC'99)*, Karlsruhe, Germany, September 1999, pp 234–245.

59. Huang, Y., Gross, M. D., Do, E. Y. L., and Eisenberg, M. (2009). Easigami: A reconfigurable folded-sheet TUI. In: *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, ACM, pp. 107–112.

60. Hunt, A., and Kirk, R. (2000). Trends in Gestural Control of Music, chapter *Mapping Strategies for Musical Performance*. Ircam - Centre Pompidou.

61. Ishii, H., and Ullmer, B. (1997) Tangible Bits: Towards Seamless Interfaces between People , Bits and Atoms. (S. Pemberton, Ed.) *Interfaces*, 97pp(March), ACM, pp. 234-241.

62. Izadi, S., Hodges, S., Taylor, S., Rosenfeld, D., Villar, N., Butler, A., and Westhues, J. (2008). 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).* ACM, New York, NY, USA, pp. 269-278.

63. Jacob, J.K.R., Sibert, L.E. McFarlane, D.C., and Preston Mullen Jr, M. (1994). Integrality and separability of input devices. *ACM Trans. Comput.-Hum. Interact.* 1, 1 (March 1994), pp. 3-26.

64. Jagacinski, R. J., Repperger, D. W., Moran, M. S., Ward, S. L., and Glass, B. (1980). Fitts' Law and the Microstructure of Rapid Discrete Movements. *Journal of Experimental Psychology Human Perception and Performance*, 6(2), pp. 309-320.

65. Jagacinski, R. J., Repperger, D. W., Ward, S. L., and Moran, M. S. (1980). A test of fitts' law with moving targets. *Hum Factors*, 22(2):225–233, April 1980.

66. Jagacinski, R.J., and Monk, D.L. (1985). Fitts' Law in two dimension with hand and head movements. *Journal of motor behavior*, 17, 77-95.

67. Jagacinski, R.J., Monk, D.L. (1985). Fitts' Law in two dimensions with hand

and head movements. *Journal of motor behavior*, Vol. 17, No. 1, pp. 77-95.

68. Jagacinski, R.J., Repperger, D.W., Ward, S.L. and Moran, M.S. (1980). A test of Fitts' Law with moving targets, *Human Factors*, 22, pp. 225-233.

69. Jetter, H.C., Gerken, J., Zöllner, M., Reiterer, H., and Milic-Frayling, N. (2011), Materializing the Query with Facet-Streams–A Hybrid Surface for Collaborative Search on Tabletops, *The ACM CHI Conference on Human Factors in Computing Systems CHI*, May, Vancouver, Canada, pp. 7–12.

70. Jordà, S. (2007). Interactivity and Live Computer Music, In: *The Cambridge Companion to Electronic Music,* Edited by Nick Collins and Julio d'Escrivan. Cambridge University Press, UK.

71. Jordà, S. (2008). On stage: the reactable and other musical tangibles go real, *Int. J. Arts and Technology,* Vol. 1, Nos.

72. Jorda, S., Kaltenbrunner, M., Geiger, G., and Alonso, M. (2006) The reacTable: A Collaborative Musical Instrument. 15th IEEE *International Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises WETICE06*, pp. 406-411.

73. Kabbash, P., Buxton, W. and Sellen, A. (1994). Two-Handed Input in a Compound Task, In: *Proc. of CHI94*, pp. 417-423.

74. Kaltenbrunner, M. (2009). Reactivision and Tuio: a tangible tabletop toolkit. In: *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pp. 9–16.

75. Kaltenbrunner, M., and Bencina, R. (2007). Reactivision: a computer-vision framework for table-based tangible interaction. In: *Proceedings of the 1st international conference on Tangible and Embedded Enteraction*, pp. 69–74.

76. Karam, M. (2005). A taxonomy of Gestures in Human Computer Interaction. *ACM Transactions on ComputerHuman Interactions*, (ECSTR-IAM05-009), pp. 1-45.

77. Kiousis, S. (2002). 'Interactivity: a concept explication'. New Media and Society, Vol4(3), pp. 355–383.

78. Kirk, D., Sellen, A., Taylor, S., Villar, N., and Izadi. S. (2009). Putting the physical into the digital: Issues in designing hybrid interactive surfaces. In: *Proceedings of the 2009 British Computer Society Conference on Human-Computer Interaction*, pp. 35–44.

79. Kopper, R., Bowman, D.A., Silva, M.G., and McMahan, R.P. (2010). A human motor behavior model for distal pointing tasks. *Int. J. Hum.-Comput. Stud.* 68, 10 (October 2010), pp. 603-615.

80. Krueger, M.W. (1983). Artificial Reality, Addison-Wesley, Reading, MA.

81. Krueger, M.W., Gionfriddo, T., and Hinrichsen, K. (1985). VIDEOPLACE - An Artificial Reality, In: *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'85)*, pp. 35–40.

82. Kunz, A., Fjeld, M. (2010). From Table–System to Tabletop: Integrating Technology into Interactive Surfaces, In: *Tabletops - Horizontal Interactive Displays*, ISBN 978-1-84996-112-7, Volume Human-Computer Interaction

Series, Issue Tabletops - Horizontal Interactive Displays, pp. 53-72.

83. Lave, J., and Wenger, E. (1991). Situated Learning; legitimate peripheral participation. Cambridge University Press.

84. Lee, S., Buxton, W., and Smith, K.C. A multi-touch three dimensional touch-sensitive tablet. In: *Proc.* CHI 1985, ACM Press (1985), pp. 21–25.

85. Leganchuk, A., Zhai, S.& Buxton, W. (1998). Manual and Cognitive Benefits of Two-Handed Input: An Experimental Study. *Transactions on Human-Computer Interaction*, 5(4), pp. 326–359.

86. Lucchi, A., Jermann, P., Zufferey, G., and Dillenbourg, P. (2010). An empirical evaluation of touch and tangible interfaces for tabletop displays. In*: Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction* (TEI '10). ACM, New York, NY, USA, pp. 177-184.

87. MacKenzie, I. S., and Buxton, W. (1994). The prediction of pointing and dragging times in graphical user interfaces. *Interacting with Computers*, 6, pp. 213-227.

88. MacKenzie, I. S., and W. A. S. Buxton. 1992. ‘‘Extending Fitts’ Law to Two Dimensional Tasks.’’ In: *Proceedings of the 1992 ACM Conference on Human Factors in Computing Systems*. New York: ACM Press, pp. 219–226.

89. MacKenzie, I. S., Sellen, A., and Buxton, W. (1991). A comparison of input devices in elemental pointing and dragging tasks. In: *Proceedings of the CHI `91 Conference on Human Factors in Computing Systems*, New York: ACM, pp. 161-166.

90. MacKenzie, I.S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Hum.-Comput. Interact.* 7, 1 (March 1992), pp. 91-139.

91. Mackinlay, J., Card, S.K., and Robertson, G.G. (1990). A semantic analysis of the design space of input devices. *Hum.-Comput. Interact.* 5, 2 (June 1990), pp. 145-190.

92. Maekawa, T., Itoh, Y., Kawai, N., Kitamura, Y., and Kishino, F. (2009). MADO interface: a window like a tangible user interface to look into the virtual world. In: *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, ACM, pp. 175–180.

93. Mallett, R. (2006). Time Traveler: A Scientist's Personal Mission to Make Time Travel a Reality. Thunder's Mouth Press.

94. Marshall, P., Fleck, R., Harris, A., Rick, J., Hornecker, E., Rogers, Y., Yuill, N., and Dalton, N.S. (2009). Fighting for control: children's embodied interactions when using physical and digital representations. In: *Proceedings of the 27th international conference on Human factors in computing systems (CHI '09)*. ACM, New York, NY, USA, pp. 2149–2152.

95. McAvinney, P. (1986). The Sensor Frame - A Gesture-Based Device for the Manipulation of Graphic Objects, Carnegie-Mellon University.

96. Merrill, D., Kalanithi, J., and Maes, P. (2007). Siftables: towards sensor

network user interfaces. In: *Proceedings of the 1st international conference on Tangible and embedded interaction*, ACM, pp. 75–78.

97. Meyer, D. E., Abrams, R. A., Kornblum, S., Wright, C. E., and Smith, J. E. (1988). Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological Review*, 95(3), American Psychological Association, pp. 340-370.

98. Moscovich, T., and Hughes, J.F. (2008). Indirect mappings of multi-touch input using one and two hands. In: *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI '08)*. ACM, New York, NY, USA, pp. 1275–1284.

99. Nielsen, B., Ren, S., and Agha, G. (1998). Specification of Real-Time Interaction Constraints. In: *Proceedings of the The 1st IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC '98)*. IEEE Computer Society, Washington, DC, USA.

100. Parés, N. and Parés, R. (2006). "Towards a Model for a Virtual Reality Experience: The Virtual Subjectiveness". *Presence* 15(5), pp. 524–538.

101. Patten J, Ishii H, Hines J, Pangaro G (2001) SenseTable: A Wireless Object Tracking Platform for Tangible User Interfaces. In: *Proceedings of CHI 2001*; ACM, New York, NY, USA, pp. 253-260.

102. Patten, J., Recht, B., and Ishii, H. (2002). Audiopad: a tag-based interface for musical performance. In: *Proceedings of the 2002 conference on New interfaces for musical expression (NIME '02)*, Eoin Brazil (Ed.). National University of Singapore, Singapore, Singapore, pp. 1-6.

103. Penny, S. (2010/2011) Interactivity – Who cares? Forthcoming in: *Fiberculture*, Open Humanities Press, Douglas Kellner, ed.

104. Piazza, T., Fjeld, M. (2007). Ortholumen: Using Light for Direct Tabletop Input. In: *Proceedings of IEEE TableTop 2007*, IEEE Computer Society, Los Alamitos, CA, USA, pp. 193-196.

105. Port, N.L., Lee, D., Dassonville, P., and Georgopoulos, A. P. (1997) Manual interception of moving targets: I. Performance and movement initiation, *Experimental Brain Research*, Springer Berlin / Heidelberg, Biomedical and Life Sciences, Volume 116, Issue 3, pp. 406-420.

106. Radwin, R. G., Vanderheiden, G. C., and Lin, M. L. (1990). A method for evaluating head-controlled computer input devices using Fitts law. *Human Factors*, 32(4), pp. 423-438.

107. Radwin, R.G., Vanderheiden, G.C., and Lin, M. (1990). A method for evaluating head-controlled computer input devices using Fitts law. *Hum. Factors* 32, 4 (August 1990), 423-438.

108. Rafaeli, S. (1988) 'Interactivity: from New Media to Communication', in Hawkins, R. P., Wieman, J. M. and Pingree, S. (eds.), *Advancing Communication Science: Merging ass and Interpersonal Processes.* Newbury Park, CA: Sage, pp. 110–34.

109. Rekimoto, J. (2002). SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces. In: *Proceedings of CHI 2002*; ACM,

New York, NY, USA, pp. 113-120.

110. Smith, G.C. (1995). The marble answering machine. In: The Hand That Rocks the Cradle, pages 60–65.

111. Steuer, J.S. (1992). "Defining Virtual Reality: Dimensions Determining Telepresence", *Journal of Communication* 42(4), pp. 73–93.

112. Suzuki, H., and Kato, H. (1995). Interaction-level support for collaborative learning: AlgoBlock — An open programming language, In: *Proceedings of CSCL*.

113. Terrenghi, L., Kirk, D., Sellen, A., and Izadi, S. (2007). Affordances for manipulation of physical versus digital media on interactive surfaces. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 1157–1166.

114. Tuddenham, P., Kirk, D.S., Izadi, S. (2010) Graspables revisited: multi-touch vs. tangible input for tabletop displays in acquisition and manipulation tasks. In: *CHI 2010*, pp. 2223-2232.

115. Ullmer, B. (2002). Tangible Interfaces for Manipulating Aggregates of Digital Information, *Ph.D. Dissertation*, Massachusetts Institute of Technology.

116. Ullmer, B. and Ishii, H. (2000). Emerging frameworks for tangible user interfaces. *IBM Syst. J.* 39, 3-4 (July 2000), pp. 915-931.

117. Ullmer, B., and Ishii, H. (1997) The metaDESK: models and prototypes for tangible user interfaces. In: *Proceedings of the 10th annual ACM symposium on User interface software and technology,* ACM, Vol. 97, pp. 223-232.

118. Underkoffler, J. and Ishii, H. (1999). Urp: A luminous-tangible workbench for urban planning and design, In: *Proceedings of CHI '99*, NY: ACM, pp. 386–393.

119. Wanderley, M. M. (2001), September 2001. Gestural control of music. Paper presented at the *Int. Workshop on Human Supervision and Control in Engineering and Music*, Kassel, Germany, September.

120. Wang, F., and Ren, X. (2009). Empirical evaluation for finger input properties in multi-touch interaction. In: *Proceedings of the 27th international conference on Human factors in computing systems (CHI '09)*. ACM, New York, NY, USA, pp. 1063–1072.

121. Weiss, M., Voelker, S., Sutter, C., and Borchers, J. (2010). BendDesk: Dragging Across the Curve. ACM, Work, pp. 1-10.

122. Wellner, P. (1993) Interacting with Paper on the Digital Desk. *Commun.* ACM, pp. 86-96.

123. Wilson, A. (2005). PlayAnywhere: A Compact Interactive Tabletop Projection-Vision System. In: *Proceedings of the 18th Annual ACM Synposium on User Interface Software and Technology UIST 2005*, ACM, New York, NY, USA, pp. 83-92.

124. Zabramski, S., Gkouskos, D., Lind, M. (2011). A comparative evaluation of mouse, stylus and finger input in shape tracing. I Georgios Christou,

Panayiotis Zaphiris, Ee Lai-Chong Law (red.) In: *Proceedings of the 1st European Workshop on HCI Design and Evaluation*, Toulouse, France: IRIT Press, pp. 57-61.

125. Zabramski, S., Gkouskos, D., Lind, M. A. (2011). Comparative evaluation of mouse, stylus and finger input in shape tracing. In: *Proceedings of the 1st European Workshop on HCI Design and Evaluation: The inuence of domain on Human Computer Interaction design and evaluation*. Toulouse, France: IRIT Press, pp. 57-61.

126. Zaleski, M. and Moray, M. (1986). Hitts' law? A test of the relationship between information load and movement precision. *In: Proceedings of the 21st Annual NASA University Conference on Manual Control.*

127. Zhai, S., and Milgram, P. (1998). Quantifying coordination in multiple DOF movement and its application to evaluating 6 DOF input devices. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '98), Clare-Marie Karat, Arnold Lund, Jolle Coutaz, and John Karat (Eds.). *ACM Press/Addison-Wesley Publishing Co*., New York, NY, USA, pp. 320-327.

128. Zhai, S., Kong, J., and Ren, X. (2004). Speed-accuracy tradeoff in Fitts' law tasks: on the equivalency of actual and nominal pointing precision. *Int. J. Hum.-Comput.* Stud. 61, 6 (December 2004), pp. 823-856.

APPENDIX A

1.1 Tables of Results Task 1

Table 2: Overall performance Task 1. Mean and SD

Report

Performance

Condition	Mean	N	Std. Deviation
Tangible	,425	853	5,7518
Multitouch	-,932	774	6,6174
Total	-,220	1627	6,2139

Table 3: Mean performance of subjects in tangible and multi-touch conditions. Paired Samples t-test

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Tangible	,220	10	1,2327	,3898
	Multitouch	-,940	10	1,7135	,5418

Correlation for paired samples

		N	Correlation	Sig.
Pair 1	Tangible and Multitouch	10	,613	,060

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	Tangible - Multitouch	1,1600	1,3664	,4321	,1825	2,1375	2,685	9	,025

Table 4: Analysis of variance between performance under tangible and multi-touch condition. One-Way ANOVA

ANOVA

Performance

	Sum of Squares	df	Mean Square	F	Sig.
Fra gruppi	747,305	1	747,305	19,575	,000
Entro gruppi	62037,490	1625	38,177		
Totale	62784,796	1626			

1.1.1 Correlation analysis Task 1

Table 5: Correlation analysis of global performance over target size, target distance and target angle. Pearson's Correlation

Correlation

		Performance	target_size	target_dist	target_angle
Performance	Pearson Correlation	1	,068**	-,036	-,615**
	Sig. (2-tailed)		,006	,151	,000
	N	1627	1627	1627	1627
target_size	Pearson Correlation	,068**	1	-,806**	-,093**
	Sig. (2-tailed)	,006		,000	,000
	N	1627	1627	1627	1627
target_dist	Pearson Correlation	-,036	-,806**	1	,090**
	Sig. (2-tailed)	,151	,000		,000
	N	1627	1627	1627	1627
target_angle	Pearson Correlation	-,615**	-,093**	,090**	1
	Sig. (2-tailed)	,000	,000	,000	
	N	1627	1627	1627	1627

**.Correlation is significant at level 0,01 (2-tailed).

Table 6: Independent correlation analysis of tangible and multi-touch performance over target size, target distance and target angle. Pearson's Correlation

Correlation

		Performance_T	Performance_MT	target_size	target_dist	target_angle
Performance_T	Correlazione di Pearson	1		,063	-,048	-,711**
	Sig. (2-code)		.	,067	,158	,000
	N	853	0	853	853	853
Performance_MT	Correlazione di Pearson		1	,051	-,003	-,525**
	Sig. (2-code)	.		,158	,925	,000
	N	0	774	774	774	774
target_size	Correlazione di Pearson	,063	,051	1	-,806**	-,093**
	Sig. (2-code)	,067	,158		,000	,000
	N	853	774	1627	1627	1627
target_dist	Correlazione di Pearson	-,048	-,003	-,806**	1	,090**
	Sig. (2-code)	,158	,925	,000		,000
	N	853	774	1627	1627	1627
target_angle	Correlazione di Pearson	-,711**	-,525**	-,093**	,090**	1
	Sig. (2-code)	,000	,000	,000	,000	
	N	853	774	1627	1627	1627

**.Correlation is significant at level 0,01 (2-tailed).

1.2 Tables of results Task 2

Table 7: Overal performance Task2. Mean and SD

Report

performance

Condition	Mean	N	Std. Deviation
Tangible	-57,082	870	59,4835
Multitouch	-63,566	844	59,4420
Total	-60,275	1714	59,5341

Table 8: Mean performance of subjects in tangible and multi-touch conditions. Paired samples Wilcoxon test

Ranks

		N	Mean Ranks	Suma of Ranks
Performance_MT - Performance_T	Negative Ranks	8[a]	5,88	47,00
	Positive Ranks	2[b]	4,00	8,00
	Ties	0[c]		
	Total	10		

a. Performance_MT < Performance_T
b. Performance_MT > Performance_T
c. Performance_MT = Performance_T

Test[b]

	Performance_MT - Performance_T
Z	-1,988[a]
Sig. Asymp. 2-tailed	,047

1.2.1 Correlation analysis

Table 9: Correlation analysis of global performance over target speed, target starting position and target speed*initial position (difficulty). Spearman's Correlation

Correlation

			performance	ballspeed	ballx	difficulty
Spearman's Rho	performance	Correlation Coefficient	1,000	-,681**	,126**	-,753**
		Sig. (2-tailed)	.	,000	,000	,000
		N	1714	1714	1714	1714
	ballspeed	Correlation Coefficient	-,681**	1,000	-,007	,659**
		Sig. (2-tailed)	,000	.	,763	,000
		N	1714	1714	1714	1714
	ballx	Correlation Coefficient	,126**	-,007	1,000	-,189**
		Sig. (2-tailed)	,000	,763	.	,000
		N	1714	1714	1714	1714
	difficulty	Correlation Coefficient	-,753**	,659**	-,189**	1,000
		Sig. (2-tailed)	,000	,000	,000	.
		N	1714	1714	1714	1714

**.Correlation significant at level 0,01 (2-tailed).

Table 10: Independent corelation analysis for tangible and multi-touch performance over target speed, target starting position and target speed*initial position (difficulty). Spearman's Correlation

Correlation

			Performance_T	Performance_MT	ballspeed	ballx	difficulty
Spearman's Rho	Performance_T	Correlation Coefficient	1,000	.	-,689**	,127**	-,737**
		Sig. (2-tailed)	.	.	,000	,000	,000
		N	870	0	870	870	870
	Performance_MT	Correlation Coefficient	.	1,000	-,679**	,127**	-,774**
		Sig. (2-tailed)	.	.	,000	,000	,000
		N	0	844	844	844	844
	ballspeed	Correlation Coefficient	-,689**	-,679**	1,000	-,007	,659**
		Sig. (2-tailed)	,000	,000	.	,763	,000
		N	870	844	1714	1714	1714
	ballx	Correlation Coefficient	,127**	,127**	-,007	1,000	-,189**
		Sig. (2-tailed)	,000	,000	,763	.	,000
		N	870	844	1714	1714	1714
	difficulty	Correlation Coefficient	-,737**	-,774**	,659**	-,189**	1,000
		Sig. (2-tailed)	,000	,000	,000	,000	.
		N	870	844	1714	1714	1714

**.Correlation significant at level 0,01 (2-tailed).

1.2.2 Reaction Time analysis

Table 11: Overal performance reaction time, Mean and SD

Report

Reaction_Time

Condition	Mean	N	Std. Deviation
Tangible	476,320	10	24,7735
Multitouch	491,150	10	22,2479
Totale	483,735	20	24,1464

Table 12: Correlation analysis of mean reaction time over target speed, target initial position and target speed*initial position (difficulty). Spearman's Correlation

Correlation

			first_reaction_millis	ballspeed	ballx	difficulty
Spearman's Rho	first_reaction_millis	Correlation Coefficient	1,000	-,269**	-,081**	-,060*
		Sig. (2-tailed)	.	,000	,001	,013
		N	1714	1714	1714	1714
	ballspeed	Correlation Coefficient	-,269**	1,000	-,007	,659**
		Sig. (2-tailed)	,000	.	,763	,000
		N	1714	1714	1714	1714
	ballx	Correlation Coefficient	-,081**	-,007	1,000	-,189**
		Sig. (2-tailed)	,001	,763	.	,000
		N	1714	1714	1714	1714
	difficulty	Correlation Coefficient	-,060*	,659**	-,189**	1,000
		Sig. (2-tailed)	,013	,000	,000	.
		N	1714	1714	1714	1714

**.Correlation significant at level 0,01 (2-tailed).
*.Correlation significant at level 0,05 (2-tailed).

1.3 Post-Test analysis

1.3.1 Tangible evaluation (word cloud)

1.3.2 Multi-Touch evaluation (word cloud)

1.3.3 Comments from the interview

The following part of the appendix reports the comments extracted through the post interview about user experience and controls usability. The subjects have been guided to the interview, by making them choose the three adjectives that they regarded as more significant to describe their performances, amongst the ones previously chosen in the post test. All the discussions emerged during the interview were developed on top of this choices. Since many trends in describing the controls were very similar, the most significant comments will be here reported.

Positive comments on tangible controls:

"I didn't need any particular instruction to understand how to control the paddle in the gameplay"

"I think the shape of the object was appropriate for this type of game"

“I thin the control was really adequate for the first task, especially for finding the right angle, it was very reliable in rotation”

“the control made the game very entertaining and fun”

“even it was simple to use it was challenging anyway, especially in the first task I felt I could calibrate my movements very well”

“controlling the paddle with the tangible makes the game really entertaining”

“by using this control, I made sense of the task immediately and without problems, I think it’s appropriate for this kind of video games”

“I could perceive my learning curve while playing with it, I think I improved during the gameplay, it was definitely better than the multi-touch”

“the haptic feedback made the control definitely more reliable and easy to use”

“it’s nice because you don’t have to apply any pressure on the surface when you use it”

“it really allows you to use your gaming skills”

“when I was using it I didn’t have to be concerned about the image of the paddle, and it was better because in this case I could concentrate more on the targets”

Negative comments on tangible controls:

“in the second task, because of the delay I felt that was slow”

“I felt a bit tired with my hand in rotating the wrist”

“the latency was making me think that I was loosing control”

“sometimes the object was frictioning on the surface and it felt a bit uncomfortable”

“I think multi-touch was better for the second task”

Positive comments on multi-touch control:

“the control is intuitive and it was funny to play with it”

“I think using two fingers from the same hand to control the paddle was a good combination”

“the control was very straightforward”

“you can choose your own way to interact with it”

“the relation with the virtual object is direct and you understand immediately how to control it”

“I didn’t need someone to explain how to control the virtual object, and it was very nice to control it with a part of the body, because it makes the movements very intuitive”

“I liked it more than the tangible, I felt more free I using it”

Negative comments on multi-touch controls:

“it was tiring rotating with my wrist all the time in the first task, maybe controlling it with two hands would be better”

“I felt that I was not able to keep the proper angle in the first task”

“the fingers were often loosing the contact with the surface and it made me stressed”

“the control is inefficient because is not able to respond to my movements in the proper way, especially in the second task, it wasn’t as fast as I”

“the tracking is too problematic, I couldn’t play and have fun”

“is not stable and not appropriate for the surface, it’s annoying the continuous friction”

“I felt I wasn’t free to move as I wanted, because it often loosed the contact”

“is not good for wrist rotation, not comfortable and not reliable”

“it’s more tyring and less precise than the tangible”

“at long time it’s tiring, and it doesn’t feel comfortable in extreme rotation”

“I felt definitely more sure and precise with the tangible”

APPENDIX B

1.1 Pre-Test Questionnaire

Please read the questions carefully before responding.
If you have any doubts ask the experimenter for help.
Note that all the information that you will provide here will be kept absolutely confidential and in respect of your personal privacy.
Thank you for participating.

* Required

Subject *
Do not fill this field, the experimenter will do it, thank you.

Age *

Sex *

- ○ Male
- ○ Female

Nationality *

Have you ever used a Tabletop interface (e.g. ReacTable) ? *

- ○ Yes
- ○ No

Have you ever used a multitouch interface (e.g. ipad, android tablet) ? *

- ○ Yes
- ○ No

How much do you enjoy playing videogames? *

	1	2	3	4	5	
Not at All	○	○	○	○	○	Very Much

How often do you play videogames? *

- ○ Daily
- ○ 2-3 times a week
- ○ Once a month
- ○ 2-3 times a year
- ○ Never

What is your preferred hand? *

- ○ Right
- ○ Left

Do you suffer from any deficiency or any pathological problem to your eyes? *

- ○ Yes
- ○ No

If yes, please describe it here briefly.

Do you suffer from any syndrom or any pathological problem at your arms or junctions (e.g. carpal tunnel syndrome) *

- ○ Yes
- ○ No

If yes, please describe it here briefly.

1.2 Post-Test Questionnaire

Read over the following list of adjectives and choose the ones that you think more apply to the interface you have just used.
Please read them carefully before choosing amongst them.
If you have any doubts ask the experimenter.
Note that all the information you will provide here will be kept absolutely confidential and in respect of your personal privacy.
Thank you for participating.

* Required

Subject *
Do not fill this field, the experimenter will do it, thank you.

Considering the control you have just used, tick those adjectives that best describe your experience with it. *
You can choose as many adjectives as you wish. Please choose more than six.

- ☐ Accessible
- ☐ Approachable
- ☐ Adequate
- ☐ Boring
- ☐ Complex
- ☐ Controllable
- ☐ Difficult
- ☐ Easy to Use
- ☐ Efficient
- ☐ Effortless
- ☐ Entertaining
- ☐ Familiar
- ☐ Fast
- ☐ Flexible

- [ ] Frustrating
- [ ] Fun
- [ ] Hard to Use
- [ ] Inadequate
- [ ] Inefficient
- [ ] Intuitive
- [ ] Responsive
- [ ] Rigid
- [ ] Secure
- [ ] Slow
- [ ] Stable
- [ ] Stressful
- [ ] Uncontrollable
- [ ] Unapproachable
- [ ] Unusable
- [ ] Usable
- [ ] Useful

Now please indicate the three adjectives amongst the one you have chosen, that most closely match your personal reactions to the control you had just used. *

Please indicate just three adjectives, write them in capitol letters.

APPENDIX C

1.1 Processing code for application used in Task 1, implementation of Multi-Touch control

```
import fullscreen.*;
import processing.opengl.*;
import javax.media.opengl.GL;
import fisica.*;
import ddf.minim.*;
import TUIO.*;

PrintWriter output;

Minim minim;
AudioSample bop;
AudioSample ball;
AudioSample score;
AudioSample start;
AudioPlayer count;

FWorld world;
FBox paddle;
FCircle target_1, target_2;
FCircle ball1;
FCircle ball2;

TuioProcessing tuioClient;

boolean is_t = false;
boolean is_a = false;
boolean is_b = false;
boolean ball1Exists = false;
boolean ball2Exists = false;

FullScreen fs;

PGraphicsOpenGL pgl;
GL gl;

float px,py,px2,py2;
float angle,angle2;
float radius = 100;
float frequency = 3;
float frequency2 = 3;
float x, x2;

final int RED = 1;
final int BLU = 0;

final int START = 1;
final int R = 2;
final int B = 3;
final int END = 0;
```

```
int playingTime;
int ballTime;
int totalTime;
int startingTime;
int contact_p_b = 0;
int state;
int red_state;
int blu_state;
int[] randDist = {200,300,400,450};
int[] randDispRed = {300,400,490};
int[] randDispBlu = {550,650,740};
int[] randSize = {10,15,25,30};
int targetX;
int targetY;
int rdist;
int rdispR;
int rdispB;
int rs;
int targetWidth;
float ballPosX;
float ballPosY;

float alfa = 50;

PFont f;
PFont adore;
TuioCursor a;
TuioCursor b;
TuioObject my_obj;


static final boolean FULL_SCREEN = true;
//static final boolean DUAL_VIEW    = true;

//static final boolean FULL_SCREEN = false;
static final boolean DUAL_VIEW   = false;

static final boolean DEBUG = false;

static final float M_PI  = (float)Math.PI;
static final float M_PI2 = (float)Math.PI*2.0;

//
// Calibration values
//
int g_calibration_mode = 0;

static final int OFFSET_XY = 1;
static final int ROTATE_XY = 2;
static final int ROTATE_Z  = 3;
static final int SCALE_XY  = 4;

float g_ax = 0.0, g_ay = 0.0, g_az = 0.0;
float g_sx = 1.0, g_sy = 1.0;
float g_ox = 0.0, g_oy = 0.0;
```

```
// Load / Store Calibration Parameters
// Thx to http://www.vbforums.com/showthread.php?t=308145
void CalibrationSave(String filename)
{
float[] params = { g_ax, g_ay, g_az,    g_sx, g_sy,    g_ox, g_oy };


      try {
         FileOutputStream fos = new FileOutputStream(filename);
         ObjectOutputStream out = new ObjectOutputStream(fos);
         out.writeObject(params);
         out.flush();
         out.close();
         println("Calibration file saved.");
      }
      catch (IOException e)
      {
          println(e);
      }
}


void CalibrationLoad(String filename)
{
       try {
         FileInputStream fis = new FileInputStream(filename);
         ObjectInputStream in = new ObjectInputStream(fis);
         float[] p = (float[])in.readObject();

         g_ax = p[0];
         g_ay = p[1];
         g_az = p[2];
         g_sx = p[3];
         g_sy = p[4];
         g_ox = p[5];
         g_oy = p[6];

         in.close();

         println("Calibration file loaded.");
       }
       catch (Exception e)
       {
           println(e);
       }
}




void DrawCalibrationGrid()
{
 float tx = width/2.5;
 float ty = height/2.5;
 float tz = -1;

// stroke(0,256,0);
// fill(0,256,0);
```

```
stroke(255);
fill(255);

 for(int i=0; i<7; i++)
 {
  float ix = width*i/6.0;
  float iy = height*i/6.0;
  line(ix, 0, ix, height);
  line(0, iy, width, iy);
 }

 stroke(255,0,0);
 fill(255,0,0);

  if(g_calibration_mode == OFFSET_XY)
    text("> OFFSET XY <", tx, ty, tz);

  if(g_calibration_mode == ROTATE_XY)
    text("> ROTATE XY <", tx, ty, tz);

  if(g_calibration_mode == ROTATE_Z )
    text("> ROTATE Z <",  tx, ty, tz);

  if(g_calibration_mode == SCALE_XY )
    text("> SCALE XY <",  tx, ty, tz);
}

void keyPressed()
{

if(key == 'r') // Reset
{
 g_ax = 0.0; g_ay = 0.0; g_az = 0.0;
 g_sx = 1.0; g_sy = 1.0;
 g_ox = 0.0; g_oy = 0.0;
}

if(key == 's') // Save to file
{
 CalibrationSave("calibrationS2_M.dat");
}

if(key == 'l') // Load from file
{
 CalibrationLoad("calibrationS2_M.dat");
}


if(key == ENTER)
{
  g_calibration_mode = (g_calibration_mode + 1) % 5;
}

if (key != CODED)
{
  return;
}
```

```
if(g_calibration_mode == OFFSET_XY)
{
  if (keyCode == LEFT)   g_ox-=2.5;
  if (keyCode == RIGHT) g_ox+=2.5;
  if (keyCode == UP)     g_oy-=2.5;
  if (keyCode == DOWN)   g_oy+=2.5;
}
if(g_calibration_mode == ROTATE_XY)
{
  if (keyCode == LEFT)   g_ax-=0.025;
  if (keyCode == RIGHT) g_ax+=0.025;
  if (keyCode == UP)     g_ay-=0.025;
  if (keyCode == DOWN)   g_ay+=0.025;
}
if(g_calibration_mode == ROTATE_Z)
{
  if (keyCode == LEFT)   g_az-=0.0125;
  if (keyCode == RIGHT) g_az+=0.0125;
}
if(g_calibration_mode == SCALE_XY)
{
  if (keyCode == LEFT)   g_sx-=0.025;
  if (keyCode == RIGHT) g_sx+=0.025;
  if (keyCode == UP)     g_sy-=0.025;
  if (keyCode == DOWN)   g_sy+=0.025;
}

}


// Original idea by Daniel Gallardo
void ApplyCalibrationMatrix()
{
int g_w = 1024;
int g_h = 768;

float fov = PI/3.0;
float cameraZ = (height/2.0) / tan(fov/2.0);

//perspective(fov, float(g_w)/float(g_h),
//            0.0, 100.0);

    translate(g_ox,
              g_oy,
                 -4.0f);

    translate(g_w/2, g_w/2, 0);

     rotateX(g_ax);
     rotateY(g_ay);

     rotateZ(g_az);

     scale(g_sx, g_sy, 1);


    translate(-g_w/2, -g_h/2, 0);
```

```
}


// these are some helper variables which are used
// to create scalable graphical feedback
float cursor_size = 20;
float object_size = 40;
float table_size = 480;
float scale_factor = 1.2;
PFont font;


// Don't show frame, as in http://processing.org/hacks/hacks:undecoratedframe
void init()
{
  if(FULL_SCREEN)
  {
    frame.removeNotify();
    frame.setUndecorated(true);
    frame.addNotify();
  }

  super.init();
}


void setup()
{

  output = createWriter("S1_M/ID_01.txt");
  startingTime = millis();
  state = START;

  totalTime = 10000;

  red_state = 0;
  blu_state = 0;
  adore = loadFont("Adore64-30.vlw");
  f = loadFont("Verdana-48.vlw");
  textSize(28);

  minim = new Minim(this);

  bop = minim.loadSample("Bop.aiff", 512);
  ball = minim.loadSample("Ball.aiff", 512);
  score = minim.loadSample("Score.aiff", 512);
  start = minim.loadSample("Start.aiff", 512);
  count = minim.loadFile("Countr.wav", 2048);

  tuioClient  = new TuioProcessing(this);

  Fisica.init(this);

  world = new FWorld();
  world.setGravity(0,0);

      rs = int(random(4));
```

```
        rdist = int(random(4));
        rdispR = int(random(3));

        target_1 = new FCircle(randSize[rs]);
        target_1.setPosition(randDispRed[rdispR], randDist[rdist]);
        target_1.setFill(255,0,0);
        target_1.setNoStroke();
        target_1.setStaticBody(true);
        world.add(target_1);

        rdispB = int(random(3));

        target_2 = new FCircle(randSize[rs]);
        target_2.setPosition(randDispBlu[rdispB], randDist[rdist]);
        target_2.setFill(0,0,255);
        target_2.setNoStroke();
        target_2.setStaticBody(true);

  if(FULL_SCREEN)
     size(1024, 768, OPENGL); // <-- resolution of the other screen here
      else
     size(1024,768, OPENGL); // <-- whatever fits the desktop

frameRate(60);

  if(DUAL_VIEW)
  {
    delay(5000); // wait for the window to be set up...
    frame.setLocation(screen.width, 0); // Move it on the other screen
  }
  else
  {
     frame.setLocation(0, 0);
  }

 // disable 2xFSAA   (enabled by default in newer Processings)
  hint(DISABLE_OPENGL_2X_SMOOTH);
  // hint(ENABLE_OPENGL_4X_SMOOTH)

  hint(DISABLE_OPENGL_ERROR_REPORT);
  hint(DISABLE_DEPTH_TEST);

 // Enable GL_*_SMOOTH hints
  smooth();

  noStroke();
  fill(0);

  hint(ENABLE_NATIVE_FONTS);
  font = createFont("Arial", 32);
  scale_factor = height/table_size;

  strokeWeight(4); // nicer

  // an instance of the TuioClient
  // since we add "this" class as an argument, the TuioClient expects
  // an implementation of the TUIO callback methods (see below)
```

```
  // 3333 is default in reactivision (and due to a bug, it seems that it can NOT be changed ?)
  //tuioClient  = new TuioProcessing(this, 3333);

  my_obj = new TuioObject(0,0,0,0,0);


output.println("ID_01\tMin\tSec\tMillis\tRed\tBlu\tTuiXpos\tTuiYpos\tAngle\tContact\tTargetX\tTar
getY\tTargetWidth\tBallX\tBallY");

  CalibrationLoad("calibrationS2_M.dat");
}


//============================================================================
============
// within the draw method we retrieve an array of TuioObject and TuioCursor
// from the TuioClient and then loop over both lists to draw the graphical feedback.
void draw()
{
    fill(0,alfa);
    noStroke();
    rect(0,0,width,height);

ApplyCalibrationMatrix();

  int milliseconds = (millis() - startingTime);
  int seconds = (millis() - startingTime) / 1000;
  int minutes = seconds / 60;

  if (seconds == totalTime){
    state = END;
    output.close();
  }
  if (seconds == totalTime - 9){
    count.play();
  }

  switch(state){
    case START:
    startSplash();
    drawAnimatedBall();
    break;
    case R:
    drawBall1();
    drawScore();
    drawLine();
    targetWidth = randSize[rs];
    targetX = randDispRed[rdispR];
    targetY = randDist[rdist];
    if(ball1Exists || ball2Exists){
    output.println(     "\t"+minutes+"\t" +seconds+"\t"+(millis() -
ballTime)+"\t"+red_state+"\t"+blu_state+"\t"+my_obj.getScreenX(width)+"\t"+my_obj.getScreenY
(height)+"\t"
    +int((my_obj.getAngleDegrees()*-1)
+90)+"\t"+contact_p_b+"\t"+targetX+"\t"+targetY+"\t"+targetWidth+"\t\t"+int(ballPosX*pow(10,
2))/pow(10,2)+"\t"+int(ballPosY*pow(10,2))/pow(10,2));
    output.flush();
    }
```

```
    break;
    case B:
    drawBall2();
    drawScore();
    drawLine();
    targetWidth = randSize[rs];
    targetX = randDispBlu[rdispB];
    targetY = randDist[rdist];
    if(ball1Exists || ball2Exists){
    output.println(     "\t"+minutes+"\t" +seconds+"\t"+(millis() -
ballTime)+"\t"+red_state+"\t"+blu_state+"\t"+my_obj.getScreenX(width)+"\t"+my_obj.getScreenY
(height)+"\t"
    +int((my_obj.getAngleDegrees()*-1)
+90)+"\t"+contact_p_b+"\t"+targetX+"\t"+targetY+"\t"+targetWidth+"\t\t"+int(ballPosX*pow(10,
2))/pow(10,2)+"\t"+int(ballPosY*pow(10,2))/pow(10,2));
    output.flush();
    }
    break;
    case END:
    endSplash();
 }
if(g_calibration_mode > 0)
{
  DrawCalibrationGrid();
 }

}
void drawBall1() {
 if (ball1Exists){
  ballPosX = ball1.getX();
  ballPosY = ball1.getY();
  if ((ballPosY < 30 || ballPosY > 760) || (ballPosX < 100 || ballPosX > 900)){
    world.remove(ball1);
    ball1Exists = false;
  }
 }
  if (state == R){
    if ((frameCount % 240) == 0) {
    ball1 = new FCircle(10);
    ball1.setGroupIndex(-1);
    ball1.setNoStroke();
    ball1.setFill(255,0,0);
    ball1.setPosition(width/2,50);
    ball1.setVelocity(0,300);
    ball1.setRestitution(1);
    ball1.setDamping(0);
    world.add(ball1);
    ballTime = millis() - playingTime;
    ball1Exists = true;
    ball.trigger();
   }
    world.step();
    world.draw(this);
  }
 }

 void drawBall2() {
 if (ball2Exists){
```

```
  ballPosX = ball2.getX();
  ballPosY = ball2.getY();
  if ((ballPosY < 30 || ballPosY > 760) || (ballPosX < 100 || ballPosX > 900)){
    world.remove(ball2);
    ball2Exists = false;
  }
 }
 if (state == B){
    if ((frameCount % 240) == 0) {
    ball2 = new FCircle(10);
    ball2.setGroupIndex(-1);
    ball2.setNoStroke();
    ball2.setFill(0,0,255);
    ball2.setPosition(width/2,50);
    ball2.setVelocity(0,300);
    ball2.setRestitution(1);
    ball2.setDamping(0);
    world.add(ball2);
    ballTime = millis() - playingTime;
    ball2Exists = true;
    ball.trigger();
   }
    world.step();
    world.draw(this);
  }
 }

 void drawLine(){
  noFill();
  stroke(255,150);
  ellipse(width/2,550,150,150);
}

void updateScore(int scorer){
    if(scorer == RED){
      red_state++;
      score.trigger();
    }
    if(scorer == BLU){
      blu_state++;
      score.trigger();
    }
}

void drawScore() {
  textFont(adore,32);
  fill(255,0,0);
  text(red_state, 335, 650);
  fill(0,0,255);
  text(blu_state, 672, 650);
}

void drawAnimatedBall(){
  fill(255,0,0);
  ellipse(px,py,10,10);

  px = width/2 + cos(radians(angle))*(radius/1.5);
```

```
  py = 550 + sin(radians(angle))*(radius/1.5);

  angle2 = 0;

  for (int i = 0; i < width; i++){
     px2 = width/8 + cos(radians(angle2))*(radius/2);
     py2 = width/8 + sin(radians(angle2))*(radius/2);
     angle2 -= frequency2;
  }

  angle -= frequency;
  x += 1;
}

void addTuioCursor(TuioCursor tcur) {
if ( !is_a && !is_b)
{ a = tcur;
  is_a = true;
  return;
}
if ( !is_a && is_b)
{ a = tcur;
  is_a = true;
  is_t = true;
  //calculate x,y,angle of my_obj
  float x = (a.getX() + b.getX())/2.0;
  float y = (a.getY() + b.getY())/2.0;
  float angle = atan2(a.getX() - b.getX(),+ (b.getY() - a.getY())) - PI/2.0;
  my_obj.update(new TuioTime(frameCount),x,y,angle);
  //addTuioObject(my_obj);
  paddle = new FBox(60, 15);
  paddle.setNoStroke();
  paddle.setFill(255);
  paddle.setStaticBody(true);
  paddle.setRestitution(1);
  paddle.setPosition(my_obj.getScreenX(width),my_obj.getScreenY(height));
  world.add(paddle);
  return;
}
if ( is_a && is_b)
{return; }
if ( is_a && !is_b)
{ b = tcur;
  is_b = true;
  is_t = true;
  //calculate x,y,angle of my_obj
  float x = (a.getX() + b.getX())/2.0;
  float y = (a.getY() + b.getY())/2.0;
  float angle = atan2(a.getX() - b.getX(),+ (b.getY() - a.getY())) - PI/2.0;;
  my_obj.update(new TuioTime(frameCount),x,y,angle);
  //addTuioObject(my_obj);
  paddle = new FBox(60, 15);
  paddle.setNoStroke();
  paddle.setFill(255);
  paddle.setStaticBody(true);
  paddle.setRestitution(1);
  paddle.setPosition(my_obj.getScreenX(width),my_obj.getScreenY(height));
  world.add(paddle);
```

```
  return;
  }
 }

void removeTuioCursor(TuioCursor tcur){
if (is_a && !is_b){
  is_a = false;
  //removeTuioObject(my_obj);
  world.remove(paddle);
  return;
}
if (!is_a && is_b){
  is_b = false;
  //removeTuioObject(my_obj);
  world.remove(paddle);
  return;
}
if (!is_a && !is_b){
  return;
  }
if ( is_a && is_b){
  is_b = false;
  //removeTuioObject(my_obj);
  world.remove(paddle);
  return;
  }
 }


/*void addTuioObject(TuioObject my_obj){
  paddle = new FBox(60, 15);
  paddle.setNoStroke();
  paddle.setFill(255);
  paddle.setStaticBody(true);
  paddle.setRestitution(1);
  paddle.setPosition(my_obj.getScreenX(width),my_obj.getScreenY(height));
 }*/

void updateTuioCursor (TuioCursor tcur){
  if (is_a && is_b){
    float x = (a.getX() + b.getX())/2.0;
    float y = (a.getY() + b.getY())/2.0;
    float angle = atan2(a.getX() - b.getX(),b.getY() - a.getY()) - PI/2.0*-1;
    my_obj.update(new TuioTime(frameCount),x,y,angle);
    paddle.setRotation(my_obj.getAngle());
    paddle.setPosition(my_obj.getScreenX(width),my_obj.getScreenY(height));
    //updateTuioObject(my_obj);
  }
}

/*void updateTuioObject (TuioObject my_obj){
  paddle.setRotation(my_obj.getAngle());
  paddle.setPosition(my_obj.getScreenX(width),my_obj.getScreenY(height));
}

void removeTuioObject(TuioObject my_obj) {
 if (is_a && !is_b){
  is_a = false;
```

```
  removeTuioObject(my_obj);
  world.remove(paddle);
  return;
 }
 if (!is_a && is_b){
  is_b = false;
  removeTuioObject(my_obj);
  world.remove(paddle);
  return;
 }
 if (!is_a && !is_b){
  return;
 }
  is_b = false;
  removeTuioObject(my_obj);
  world.remove(paddle);
  return;
  } */



void contactStarted(FContact contact) {
  FBody body1 = contact.getBody1();
  FBody body2 = contact.getBody2();

  if (contact.getBody1() == paddle){
   contact_p_b += 1;
   bop.trigger();
  }else{
   contact_p_b = 0;
  }

  FBody b = null;
  if (contact.getBody1() == target_1 || contact.getBody1() == target_2 ) {
    b = contact.getBody2();
  }
  if (b == null) {
    return;
  }

  world.remove(b);

  }

void contactResult(FContactResult result) {
  if (result.getBody1() == target_1 && result.getBody2() == ball1){
    updateScore(RED);
    ball1Exists = false;
    state = B;
  }
 if (result.getBody1() == target_2 && result.getBody2() == ball2){
    updateScore(BLU);
    ball2Exists = false;
    state = R;
  }

  if (result.getBody1() != paddle && result.getBody1() == target_1){
      rdist = int(random(4));
```

```
      rdispR = int(random(3));
      rs = int(random(4));
      world.remove(target_1);
      target_2 = new FCircle(randSize[rs]);
      target_2.setPosition(randDispBlu[rdispB], randDist[rdist]);
      target_2.setFill(0,0,255);
      target_2.setNoStroke();
      target_2.setStaticBody(true);
      world.add(target_2);
      }
  if (result.getBody1() != paddle && result.getBody1() == target_2){
      rdist = int(random(4));
      rdispB = int(random(3));
      rs = int(random(4));
      world.remove(target_2);
      target_1 = new FCircle(randSize[rs]);
      target_1.setPosition(randDispRed[rdispR], randDist[rdist]);
      target_1.setFill(255,0,0);
      target_1.setNoStroke();
      target_1.setStaticBody(true);
      world.add(target_1);
      }

}

void startSplash(){
  textFont(f,25);
  fill(255);
  textAlign(CENTER);
  text("You're about to start the experiment.",width/2 , 190);
  textFont(f,15);
  fill(255);
  textAlign(CENTER);
  text("You will see RED and BLU circles of different size appearing on the screen in
series,",width/2 , 230);
  text("RED and BLU balls will drop from the center of the screen alternately.",width/2 , 260);
  text("Your task will be to intercept the balls with the white paddle, and throw them toward the
circles.",width/2 , 290);
  textFont(f,20);
  fill(255,0,0);
  textAlign(CENTER);
  text("Try to be as accurate and as precise as possible.",width/2 , 320);
  text("Try to hit as many circles as possible.",width/2 , 350);
  fill(255);
  text("You will have 5 minutes on the whole to accomplish the task.",width/2 , 380);
  textFont(f,20);
  fill(255);
  textAlign(CENTER);
  text("Please try to be always concentrated.",width/2 , 410);
  textFont(f,20);
  fill(255);
  textAlign(CENTER);
  text("To start the experiment place the fingers on the white paddle.",width/2 , 440);
  noStroke();
  fill(255);
  rect(width/2-30,550-5,60,10);
  if(is_t == true){
    red_state = 0;
```

```
    blu_state = 0;
    state = R;
    alfa = 200;
    totalTime = 350;
    start.trigger();
    startingTime = millis();
    playingTime = millis() - startingTime;
 }
}
void endSplash() {
    fill(255);
    textFont(adore,22);
    textAlign(CENTER);
    text("Time elapsed", width/2, height/2);
    text("Thank you for participating ;)", width/2, height/2 + 30);
 }

void stop()
{
  bop.close();
  ball.close();
  score.close();
  start.close();
  count.close();
  minim.stop();

  super.stop();
}
```

1.2 Processing code for application used in Task 1, implementation of Tangible control

```
import fullscreen.*;
import processing.opengl.*;
import javax.media.opengl.GL;
import fisica.*;
import ddf.minim.*;
import TUIO.*;

PrintWriter output;

Minim minim;
AudioSample bop;
AudioSample ball;
AudioSample score;
AudioSample start;
AudioPlayer count;

FWorld world;
FBox paddle;
FCircle target_1, target_2;
FCircle ball1;
FCircle ball2;

TuioProcessing tuioClient;
TuioObject myobj;
```

```
boolean is_t = false;
boolean ball1Exists = false;
boolean ball2Exists = false;

FullScreen fs;

PGraphicsOpenGL pgl;
GL gl;

float px,py,px2,py2;
float angle,angle2;
float radius = 100;
float frequency = 3;
float frequency2 = 3;
float x, x2;
float alfa = 50;

final int RED = 1;
final int BLU = 0;

final int START = 1;
final int R = 2;
final int B = 3;
final int END = 0;

int playingTime;
int ballTime;
int totalTime;
int startingTime;
int contact_p_b = 0;
int state;
int red_state;
int blu_state;
int[] randDist = {200,300,400,450};
int[] randDispRed = {300,400,490};
int[] randDispBlu = {550,650,740};
int[] randSize = {10,15,25,30};
int targetX;
int targetY;
int rdist;
int rdispR;
int rdispB;
int rs;
int targetWidth;
float ballPosX;
float ballPosY;

PFont f;
PFont a;


static final boolean FULL_SCREEN = true;
//static final boolean DUAL_VIEW    = true;

//static final boolean FULL_SCREEN = false;
static final boolean DUAL_VIEW  = false;

static final boolean DEBUG = false;
```

```
static final float M_PI  = (float)Math.PI;
static final float M_PI2 = (float)Math.PI*2.0;

//
// Calibration values
//
int g_calibration_mode = 0;

static final int OFFSET_XY = 1;
static final int ROTATE_XY = 2;
static final int ROTATE_Z  = 3;
static final int SCALE_XY  = 4;

float g_ax = 0.0, g_ay = 0.0, g_az = 0.0;
float g_sx = 1.0, g_sy = 1.0;
float g_ox = 0.0, g_oy = 0.0;


// Load / Store Calibration Parameters
// Thx to http://www.vbforums.com/showthread.php?t=308145
void CalibrationSave(String filename)
{
float[] params = { g_ax, g_ay, g_az,    g_sx, g_sy,    g_ox, g_oy };


      try {
         FileOutputStream fos = new FileOutputStream(filename);
         ObjectOutputStream out = new ObjectOutputStream(fos);
         out.writeObject(params);
         out.flush();
         out.close();
         println("Calibration file saved.");
      }
      catch (IOException e)
      {
          println(e);
      }
}


void CalibrationLoad(String filename)
{
       try {
         FileInputStream fis = new FileInputStream(filename);
         ObjectInputStream in = new ObjectInputStream(fis);
         float[] p = (float[])in.readObject();

         g_ax = p[0];
         g_ay = p[1];
         g_az = p[2];
         g_sx = p[3];
         g_sy = p[4];
         g_ox = p[5];
         g_oy = p[6];

         in.close();
```

```
          println("Calibration file loaded.");
        }
        catch (Exception e)
        {
            println(e);
        }
}



void DrawCalibrationGrid()
{
 float tx = width/2.5;
 float ty = height/2.5;
 float tz = -1;

// stroke(0,256,0);
// fill(0,256,0);
stroke(255);
fill(255);

 for(int i=0; i<7; i++)
 {
  float ix = width*i/6.0;
  float iy = height*i/6.0;
  line(ix, 0, ix, height);
  line(0, iy, width, iy);
 }

 stroke(255,0,0);
 fill(255,0,0);

  if(g_calibration_mode == OFFSET_XY)
    text("> OFFSET XY <", tx, ty, tz);

  if(g_calibration_mode == ROTATE_XY)
    text("> ROTATE XY <", tx, ty, tz);

  if(g_calibration_mode == ROTATE_Z )
    text("> ROTATE Z <",  tx, ty, tz);

  if(g_calibration_mode == SCALE_XY )
    text("> SCALE XY <",  tx, ty, tz);
}

void keyPressed()
{

if(key == 'r') // Reset
{
 g_ax = 0.0; g_ay = 0.0; g_az = 0.0;
 g_sx = 1.0; g_sy = 1.0;
 g_ox = 0.0; g_oy = 0.0;
}

if(key == 's') // Save to file
{
```

```
 CalibrationSave("calibration.dat");
}

if(key == 'l') // Load from file
{
 CalibrationLoad("calibration.dat");
}


if(key == ENTER)
{
  g_calibration_mode = (g_calibration_mode + 1) % 5;
}

if (key != CODED)
{
  return;
}

if(g_calibration_mode == OFFSET_XY)
{
  if (keyCode == LEFT)  g_ox-=2.5;
  if (keyCode == RIGHT) g_ox+=2.5;
  if (keyCode == UP)    g_oy-=2.5;
  if (keyCode == DOWN)  g_oy+=2.5;
}
if(g_calibration_mode == ROTATE_XY)
{
  if (keyCode == LEFT)  g_ax-=0.025;
  if (keyCode == RIGHT) g_ax+=0.025;
  if (keyCode == UP)    g_ay-=0.025;
  if (keyCode == DOWN)  g_ay+=0.025;
}
if(g_calibration_mode == ROTATE_Z)
{
  if (keyCode == LEFT)  g_az-=0.0125;
  if (keyCode == RIGHT) g_az+=0.0125;
}
if(g_calibration_mode == SCALE_XY)
{
  if (keyCode == LEFT)  g_sx-=0.025;
  if (keyCode == RIGHT) g_sx+=0.025;
  if (keyCode == UP)    g_sy-=0.025;
  if (keyCode == DOWN)  g_sy+=0.025;
}

}


// Original idea by Daniel Gallardo
void ApplyCalibrationMatrix()
{
int g_w = 1024;
int g_h = 768;

float fov = PI/3.0;
float cameraZ = (height/2.0) / tan(fov/2.0);
```

```
//perspective(fov, float(g_w)/float(g_h),
//              0.0, 100.0);

    translate(g_ox,
              g_oy,
                 -4.0f);

    translate(g_w/2, g_w/2, 0);

     rotateX(g_ax);
     rotateY(g_ay);

     rotateZ(g_az);

     scale(g_sx, g_sy, 1);


    translate(-g_w/2, -g_h/2, 0);

}


// these are some helper variables which are used
// to create scalable graphical feedback
float cursor_size = 20;
float object_size = 40;
float table_size = 480;
float scale_factor = 1.2;
PFont font;


// Don't show frame, as in http://processing.org/hacks/hacks:undecoratedframe
void init()
{
  if(FULL_SCREEN)
  {
    frame.removeNotify();
    frame.setUndecorated(true);
    frame.addNotify();
  }

  super.init();
}


void setup()
{

  output = createWriter("S1_T/ID_12.txt");
  startingTime = millis();
  state = START;

  totalTime = 10000;

  red_state = 0;
  blu_state = 0;
  f = loadFont("Adore64-30.vlw");
  a = loadFont("Verdana-48.vlw");
```

```
  textSize(28);

  minim = new Minim(this);

  bop = minim.loadSample("Bop.aiff", 512);
  ball = minim.loadSample("Ball.aiff", 512);
  score = minim.loadSample("Score.aiff", 512);
  start = minim.loadSample("Start.aiff", 512);
  count = minim.loadFile("Countr.wav", 2048);


  tuioClient  = new TuioProcessing(this);

  Fisica.init(this);

  world = new FWorld();
  world.setGravity(0,0);

      rs = int(random(4));

      rdist = int(random(4));
      rdispR = int(random(3));

      target_1 = new FCircle(randSize[rs]);
      target_1.setPosition(randDispRed[rdispR], randDist[rdist]);
      target_1.setFill(255,0,0);
      target_1.setNoStroke();
      target_1.setStaticBody(true);
      world.add(target_1);

      rdispB = int(random(3));

      target_2 = new FCircle(randSize[rs]);
      target_2.setPosition(randDispBlu[rdispB], randDist[rdist]);
      target_2.setFill(0,0,255);
      target_2.setNoStroke();
      target_2.setStaticBody(true);

  if(FULL_SCREEN)
    size(1024, 768, OPENGL); // <-- resolution of the other screen here
     else
    size(1024,768, OPENGL); // <-- whatever fits the desktop

frameRate(60);

  if(DUAL_VIEW)
  {
    delay(5000); // wait for the window to be set up...
    frame.setLocation(screen.width, 0); // Move it on the other screen
  }
  else
  {
     frame.setLocation(0, 0);
  }

 // disable 2xFSAA   (enabled by default in newer Processings)
  hint(DISABLE_OPENGL_2X_SMOOTH);
  // hint(ENABLE_OPENGL_4X_SMOOTH)
```

```
  hint(DISABLE_OPENGL_ERROR_REPORT);
  hint(DISABLE_DEPTH_TEST);

 // Enable GL_*_SMOOTH hints
  smooth();

  noStroke();
  fill(0);

  hint(ENABLE_NATIVE_FONTS);
  font = createFont("Arial", 32);
  scale_factor = height/table_size;

  strokeWeight(4); // nicer

  // an instance of the TuioClient
  // since we add "this" class as an argument, the TuioClient expects
  // an implementation of the TUIO callback methods (see below)

  // 3333 is default in reactivision (and due to a bug, it seems that it can NOT be changed ?)
  //tuioClient  = new TuioProcessing(this, 3333);

  myobj = new TuioObject(0,0,0,0,0);


output.println("ID_01\tMin\tSec\tMillis\tRed\tBlu\tTuiXpos\tTuiYpos\tAngle\tContact\tTargetX\tTar
getY\tTargetWidth\tBallX\tBallY");

  CalibrationLoad("calibration.dat");

}


//============================================================================
============
// within the draw method we retrieve an array of TuioObject and TuioCursor
// from the TuioClient and then loop over both lists to draw the graphical feedback.
void draw()
{
    fill(0,alfa);
    noStroke();
    rect(0,0,width,height);

ApplyCalibrationMatrix();

  int milliseconds = (millis() - startingTime);
  int seconds = (millis() - startingTime) / 1000;
  int minutes = seconds / 60;

  if (seconds == totalTime){
    state = END;
    output.close();
  }
  if (seconds == totalTime - 9){
    count.play();
  }
```

```
  switch(state){
    case START:
    startSplash();
    drawAnimatedBall();
    break;
    case R:
    drawBall1();
    drawScore();
    drawLine();
    targetWidth = randSize[rs];
    targetX = randDispRed[rdispR];
    targetY = randDist[rdist];
    if(ball1Exists || ball2Exists){
    output.println(     "\t"+minutes+"\t" +seconds+"\t"+(millis() -
ballTime)+"\t"+red_state+"\t"+blu_state+"\t"+myobj.getScreenX(width)+"\t"+myobj.getScreenY(he
ight)+"\t"

+int(myobj.getAngleDegrees())+"\t"+contact_p_b+"\t"+targetX+"\t"+targetY+"\t"+targetWidth+"\
t\t"+int(ballPosX*pow(10,2))/pow(10,2)+"\t"+int(ballPosY*pow(10,2))/pow(10,2));
    output.flush();
    }
    break;
    case B:
    drawBall2();
    drawScore();
    drawLine();
    targetWidth = randSize[rs];
    targetX = randDispBlu[rdispB];
    targetY = randDist[rdist];
    if(ball1Exists || ball2Exists){
    output.println(     "\t"+minutes+"\t" +seconds+"\t"+(millis() -
ballTime)+"\t"+red_state+"\t"+blu_state+"\t"+myobj.getScreenX(width)+"\t"+myobj.getScreenY(he
ight)+"\t"

+int(myobj.getAngleDegrees())+"\t"+contact_p_b+"\t"+targetX+"\t"+targetY+"\t"+targetWidth+"\
t\t"+int(ballPosX*pow(10,2))/pow(10,2)+"\t"+int(ballPosY*pow(10,2))/pow(10,2));
    output.flush();
    }
    break;
    case END:
    endSplash();
 }
if(g_calibration_mode > 0)
{
  DrawCalibrationGrid();
 }

}

void drawBall1() {
 if (ball1Exists){
  ballPosX = ball1.getX();
  ballPosY = ball1.getY();
  if ((ballPosY < 30 || ballPosY > 760) || (ballPosX < 100 || ballPosX > 900)){
    world.remove(ball1);
    ball1Exists = false;
  }
 }
```

```
  if (state == R){
    if ((frameCount % 240) == 0) {
    ball1 = new FCircle(10);
    ball1.setGroupIndex(-1);
    ball1.setNoStroke();
    ball1.setFill(255,0,0);
    ball1.setPosition(width/2,50);
    ball1.setVelocity(0,300);
    ball1.setRestitution(1);
    ball1.setDamping(0);
    world.add(ball1);
    ballTime = millis() - playingTime;
    ball1Exists = true;
    ball.trigger();
   }
    world.step();
    world.draw(this);
  }
 }

void drawBall2() {
 if (ball2Exists){
  ballPosX = ball2.getX();
  ballPosY = ball2.getY();
  if ((ballPosY < 30 || ballPosY > 760) || (ballPosX < 100 || ballPosX > 900)){
    world.remove(ball2);
    ball2Exists = false;
  }
 }
  if (state == B){
   if ((frameCount % 240) == 0) {
    ball2 = new FCircle(10);
    ball2.setGroupIndex(-1);
    ball2.setNoStroke();
    ball2.setFill(0,0,255);
    ball2.setPosition(width/2,50);
    ball2.setVelocity(0,300);
    ball2.setRestitution(1);
    ball2.setDamping(0);
    world.add(ball2);
    ballTime = millis() - playingTime;
    ball2Exists = true;
    ball.trigger();
   }
  }
    world.step();
    world.draw(this);
 }

void updateScore(int scorer){
    if(scorer == RED){
      red_state++;
      score.trigger();
    }
    if(scorer == BLU){
      blu_state++;
      score.trigger();
    }
```

```
}

void drawScore() {
  textFont(f,32);
  fill(255,0,0);
  text(red_state, 335, 700);
  fill(0,0,255);
  text(blu_state, 672, 700);
}

void drawAnimatedBall(){
  fill(255,0,0);
  ellipse(px,py,10,10);

  px = width/2 + cos(radians(angle))*(radius/1.5);

  py = 550 + sin(radians(angle))*(radius/1.5);

  angle2 = 0;

  for (int i = 0; i < width; i++){
     px2 = width/8 + cos(radians(angle2))*(radius/2);
     py2 = width/8 + sin(radians(angle2))*(radius/2);
     angle2 -= frequency2;
  }

  angle -= frequency;
  x += 1;
}

void drawLine(){
  noFill();
  stroke(255,150);
  ellipse(width/2,550,150,150);
}

void contactStarted(FContact contact) {
  FBody body1 = contact.getBody1();
  FBody body2 = contact.getBody2();

  if (contact.getBody1() == paddle){
   contact_p_b += 1;
   bop.trigger();
  }else{
   contact_p_b = 0;
  }

  FBody b = null;
  if (contact.getBody1() == target_1 || contact.getBody1() == target_2 ) {
    b = contact.getBody2();
  }
  if (b == null) {
    return;
  }

  world.remove(b);

  }
```

```
void contactResult(FContactResult result) {
  if (result.getBody1() == target_1 && result.getBody2() == ball1){
    updateScore(RED);
    ball1Exists = false;
    state = B;
  }
 if (result.getBody1() == target_2 && result.getBody2() == ball2){
    updateScore(BLU);
    ball2Exists = false;
    state = R;
  }

  if (result.getBody1() != paddle && result.getBody1() == target_1){
       rdist = int(random(4));
       rdispR = int(random(3));
       rs = int(random(4));
       world.remove(target_1);;
       target_2 = new FCircle(randSize[rs]);
       target_2.setPosition(randDispBlu[rdispB], randDist[rdist]);
       target_2.setFill(0,0,255);
       target_2.setNoStroke();
       target_2.setStaticBody(true);
       world.add(target_2);
       }
  if (result.getBody1() != paddle && result.getBody1() == target_2){
       rdist = int(random(4));
       rdispB = int(random(3));
       rs = int(random(4));
       world.remove(target_2);
       target_1 = new FCircle(randSize[rs]);
       target_1.setPosition(randDispRed[rdispR], randDist[rdist]);
       target_1.setFill(255,0,0);
       target_1.setNoStroke();
       target_1.setStaticBody(true);
       world.add(target_1);
       }

}

void addTuioObject(TuioObject tobj) {
  is_t = true;
  paddle = new FBox(50,15);
  paddle.setFill(255);
  paddle.setNoStroke();
  paddle.setRestitution(1);
  paddle.setStaticBody(true);
  world.add(paddle);
}

void updateTuioObject (TuioObject tobj){
  myobj.update(new TuioTime(frameCount),tobj.getX(),tobj.getY(),tobj.getAngle());
  paddle.setRotation(tobj.getAngle());
  paddle.setPosition(tobj.getScreenX(width),tobj.getScreenY(height));
  paddle.setWidth(60);
}

void removeTuioObject (TuioObject tobj) {
```

```
  world.remove(paddle);
}
void startSplash(){
  textFont(a,25);
  fill(255);
  textAlign(CENTER);
  text("You're about to start the experiment.",width/2 , 190);
  textFont(a,15);
  fill(255);
  textAlign(CENTER);
  text("You will see RED and BLU circles of different size appearing on the screen in
series,",width/2 , 230);
  text("RED and BLU balls will drop from the center of the screen alternately.",width/2 , 260);
  text("Your task will be to intercept the balls with the white paddle, and throw them toward the
circles.",width/2 , 290);
  textFont(a,20);
  fill(255,0,0);
  textAlign(CENTER);
  text("Try to be as accurate and as precise as possible.",width/2 , 320);
  text("Try to hit as many circles as possible.",width/2 , 350);
  fill(255);
  text("You will have 5 minutes on the whole to accomplish the task.",width/2 , 380);
  textFont(a,20);
  fill(255);
  textAlign(CENTER);
  text("Please try to be always concentrated.",width/2 , 410);
  textFont(a,20);
  fill(255);
  textAlign(CENTER);
  text("To start the experiment place the tangible on the white paddle.",width/2 , 440);
  noStroke();
  fill(255);
  rect(width/2-30,550-5,60,10);
  if(is_t == true){
    red_state = 0;
    blu_state = 0;
    state = R;
    alfa = 200;
    totalTime = 350;
    start.trigger();
    startingTime = millis();
    playingTime = millis() - startingTime;
 }
}
void endSplash() {
    fill(255);
    textFont(f,22);
    textAlign(CENTER);
    text("Time elapsed", width/2, height/2);
    text("Thank you for participating ;)", width/2, height/2 + 30);
 }

void stop()
{
  bop.close();
  ball.close();
  score.close();
  start.close();
```

```
  count.close();
  minim.stop();

  super.stop();
}
```

1.3 Processing code for application used in Task 2, implementation of Multi-Touch control

```
import fullscreen.*;
import processing.opengl.*;
import javax.media.opengl.GL;
import ddf.minim.*;
import fisica.*;

import TUIO.*;

PrintWriter output;

Minim minim;
AudioSample bop;
AudioSample ball;
AudioSample score;
AudioSample start;
AudioSample wrong;
AudioPlayer count;

FWorld world;
FBox paddle;
FCircle ball1, ball2;

TuioProcessing tuioClient;

boolean is_t = false;
boolean is_a = false;
boolean is_b = false;

FullScreen fs;

PGraphicsOpenGL pgl;
GL gl;

final int RED = 1;
final int BLU = 0;

final int START = 0;
final int PLAYING = 1;
final int END = 2;

int playingTime;
int totalTime;
int startingTime;
int ballTime;
int state;
int red_state;
int blu_state;
```

```
int[] randomVel1 = {600,800,900};
int[] randomVel2 = {1000,1100,1200};
int[] randomVel3 = {1300,1400,1500};
int[] randomPos = {300,350,400,600,650,700};
int[] randomTime = {120,240,480};
int vel;
int randball;
int randpos;
int randtime;
int velCurrent = 0;
int posCurrent = 0;
int ballX;
int ballSpeed;
float ballPosY;
boolean ball1Exists = false;
boolean ball2Exists = false;

float alfa = 50;
float px,py,px2,py2;
float angle,angle2;
float radius = 100;
float frequency = 3;
float frequency2 = 3;
float x;

PFont f;
PFont adore;

TuioCursor a;
TuioCursor b;
TuioObject my_obj;


static final boolean FULL_SCREEN = true;
//static final boolean DUAL_VIEW    = true;

//static final boolean FULL_SCREEN = false;
static final boolean DUAL_VIEW  = false;

static final boolean DEBUG = false;

static final float M_PI  = (float)Math.PI;
static final float M_PI2 = (float)Math.PI*2.0;

//
// Calibration values
//
int g_calibration_mode = 0;

static final int OFFSET_XY = 1;
static final int ROTATE_XY = 2;
static final int ROTATE_Z  = 3;
static final int SCALE_XY  = 4;

float g_ax = 0.0, g_ay = 0.0, g_az = 0.0;
float g_sx = 1.0, g_sy = 1.0;
float g_ox = 0.0, g_oy = 0.0;
```

```
// Load / Store Calibration Parameters
// Thx to http://www.vbforums.com/showthread.php?t=308145
void CalibrationSave(String filename)
{
float[] params = { g_ax, g_ay, g_az,    g_sx, g_sy,    g_ox, g_oy };


      try {
         FileOutputStream fos = new FileOutputStream(filename);
         ObjectOutputStream out = new ObjectOutputStream(fos);
         out.writeObject(params);
         out.flush();
         out.close();
         println("Calibration file saved.");
      }
      catch (IOException e)
      {
          println(e);
      }
}


void CalibrationLoad(String filename)
{
       try {
         FileInputStream fis = new FileInputStream(filename);
         ObjectInputStream in = new ObjectInputStream(fis);
         float[] p = (float[])in.readObject();

         g_ax = p[0];
         g_ay = p[1];
         g_az = p[2];
         g_sx = p[3];
         g_sy = p[4];
         g_ox = p[5];
         g_oy = p[6];

         in.close();

         println("Calibration file loaded.");
       }
       catch (Exception e)
       {
           println(e);
       }
}




void DrawCalibrationGrid()
{
 float tx = width/2.5;
 float ty = height/2.5;
 float tz = -1;

// stroke(0,256,0);
```

```
// fill(0,256,0);
stroke(255);
fill(255);

 for(int i=0; i<7; i++)
 {
  float ix = width*i/6.0;
  float iy = height*i/6.0;
  line(ix, 0, ix, height);
  line(0, iy, width, iy);
 }

 stroke(255,0,0);
 fill(255,0,0);

  if(g_calibration_mode == OFFSET_XY)
    text("> OFFSET XY <", tx, ty, tz);

  if(g_calibration_mode == ROTATE_XY)
    text("> ROTATE XY <", tx, ty, tz);

  if(g_calibration_mode == ROTATE_Z )
    text("> ROTATE Z <",  tx, ty, tz);

  if(g_calibration_mode == SCALE_XY )
    text("> SCALE XY <",  tx, ty, tz);
}



void keyPressed()
{

if(key == 'r') // Reset
{
 g_ax = 0.0; g_ay = 0.0; g_az = 0.0;
 g_sx = 1.0; g_sy = 1.0;
 g_ox = 0.0; g_oy = 0.0;
}

if(key == 's') // Save to file
{
 CalibrationSave("calibrationS2_M.dat");
}

if(key == 'l') // Load from file
{
 CalibrationLoad("calibrationS2_M.dat");
}


if(key == ENTER)
{
  g_calibration_mode = (g_calibration_mode + 1) % 5;
}

if (key != CODED)
```

```
{
  return;
}

if(g_calibration_mode == OFFSET_XY)
{
  if (keyCode == LEFT)   g_ox-=2.5;
  if (keyCode == RIGHT) g_ox+=2.5;
  if (keyCode == UP)     g_oy-=2.5;
  if (keyCode == DOWN)   g_oy+=2.5;
}
if(g_calibration_mode == ROTATE_XY)
{
  if (keyCode == LEFT)   g_ax-=0.025;
  if (keyCode == RIGHT) g_ax+=0.025;
  if (keyCode == UP)     g_ay-=0.025;
  if (keyCode == DOWN)   g_ay+=0.025;
}
if(g_calibration_mode == ROTATE_Z)
{
  if (keyCode == LEFT)   g_az-=0.0125;
  if (keyCode == RIGHT) g_az+=0.0125;
}
if(g_calibration_mode == SCALE_XY)
{
  if (keyCode == LEFT)   g_sx-=0.025;
  if (keyCode == RIGHT) g_sx+=0.025;
  if (keyCode == UP)     g_sy-=0.025;
  if (keyCode == DOWN)   g_sy+=0.025;
}

}


// Original idea by Daniel Gallardo
void ApplyCalibrationMatrix()
{
int g_w = 1024;
int g_h = 768;

float fov = PI/3.0;
float cameraZ = (height/2.0) / tan(fov/2.0);

//perspective(fov, float(g_w)/float(g_h),
//            0.0, 100.0);

    translate(g_ox,
              g_oy,
                 -4.0f);

    translate(g_w/2, g_w/2, 0);

     rotateX(g_ax);
     rotateY(g_ay);

     rotateZ(g_az);

     scale(g_sx, g_sy, 1);
```

```
    translate(-g_w/2, -g_h/2, 0);

}


// these are some helper variables which are used
// to create scalable graphical feedback
float cursor_size = 20;
float object_size = 40;
float table_size = 480;
float scale_factor = 1.2;
PFont font;


// Don't show frame, as in http://processing.org/hacks/hacks:undecoratedframe
void init()
{
  if(FULL_SCREEN)
  {
    frame.removeNotify();
    frame.setUndecorated(true);
    frame.addNotify();
  }

  super.init();
}


void setup()
{
  output = createWriter("S2_M/ID_01.txt");
  startingTime = millis();
  state = START;

  totalTime = 10000;

  red_state = 0;
  blu_state = 0;
  adore = loadFont("Adore64-30.vlw");
  f = loadFont("Verdana-48.vlw");
  textSize(28);

  minim = new Minim(this);

  bop = minim.loadSample("Bop.aiff", 512);
  ball = minim.loadSample("Ball.aiff", 512);
  score = minim.loadSample("Score.aiff", 512);
  start = minim.loadSample("Start.aiff", 512);
  wrong = minim.loadSample("Wrong.aiff", 512);
  count = minim.loadFile("Countr.wav", 2048);

  tuioClient  = new TuioProcessing(this);

  Fisica.init(this);

  world = new FWorld();
```

```
  world.setGravity(0,0);


  if(FULL_SCREEN)
    size(1024, 768, OPENGL); // <-- resolution of the other screen here
     else
    size(1024,768, OPENGL); // <-- whatever fits the desktop

frameRate(60);

  if(DUAL_VIEW)
  {
    delay(5000); // wait for the window to be set up...
    frame.setLocation(screen.width, 0); // Move it on the other screen
  }
  else
  {
     frame.setLocation(0, 0);
  }

 // disable 2xFSAA   (enabled by default in newer Processings)
  hint(DISABLE_OPENGL_2X_SMOOTH);
  // hint(ENABLE_OPENGL_4X_SMOOTH)

  hint(DISABLE_OPENGL_ERROR_REPORT);
  hint(DISABLE_DEPTH_TEST);

 // Enable GL_*_SMOOTH hints
  smooth();

  noStroke();
  fill(0);

  hint(ENABLE_NATIVE_FONTS);
  font = createFont("Arial", 32);
  scale_factor = height/table_size;

  strokeWeight(4); // nicer

  // an instance of the TuioClient
  // since we add "this" class as an argument, the TuioClient expects
  // an implementation of the TUIO callback methods (see below)

  // 3333 is default in reactivision (and due to a bug, it seems that it can NOT be changed ?)
  //tuioClient  = new TuioProcessing(this, 3333);

  my_obj = new TuioObject(0,0,0,0,0);


output.println("ID_01\tMin\tSec\tMillis\tRed\tBlu\tSpeed\t\tAccel\t\tXpos\tYpos\tBallX\tBallY\tBall
Speed");

  CalibrationLoad("calibrationS2_M.dat");
}


//=========================================================================
============
```

```
// within the draw method we retrieve an array of TuioObject and TuioCursor
// from the TuioClient and then loop over both lists to draw the graphical feedback.
void draw()
{


ApplyCalibrationMatrix();


  fill(0,alfa);
  noStroke();
  rect(0,0,width,height);

  int milliseconds = (millis() - startingTime);
  int seconds = (millis() - startingTime) / 1000;
  int minutes = seconds / 60;

  if (seconds > totalTime){
    state = END;
    output.close();
  }
  if(seconds == totalTime - 8){
    count.play();
  }
   if (seconds > 180){
    ball1.setVelocity(0,randomVel2[vel]);
    ball2.setVelocity(0,randomVel2[vel]);
    ballSpeed = randomVel2[velCurrent];
  }
  if (seconds > 290){
    ball1.setVelocity(0,randomVel3[vel]);
    ball2.setVelocity(0,randomVel3[vel]);
    ballSpeed = randomVel3[velCurrent];
  }

  switch(state) {
    case START:
      startSplash();
      drawAnimatedBall();
      break;
    case PLAYING:
      drawBall();
      //drawBall2();
      drawScore();
      drawLine();
     if(ball1Exists || ball2Exists){
      output.println(    "\t"+minutes+"\t" +seconds+"\t"+(millis() -
ballTime)+"\t"+red_state+"\t"+blu_state+"\t"

+int(a.getMotionSpeed()*pow(10,2))/pow(10,2)+"\t\t"+int(a.getMotionAccel()*pow(10,2))/pow(10,
2)+"\t\t"
       +my_obj.getScreenX(width)+"\t"+my_obj.getScreenY(height)+"\t"+ballX+
"\t"+int(ballPosY)+"\t"+ballSpeed);
      output.flush();
     }
      break;
    case END:
      endSplash();
```

```
        break;

 }

if(g_calibration_mode > 0)
{
  DrawCalibrationGrid();
 }
}

void drawBall() {
if (ball1Exists){
  ballPosY = ball1.getY();
  if (ballPosY > 780){
   world.remove(ball1);
   ball1Exists = false;
  }
 }
 if (ball2Exists){
  ballPosY = ball2.getY();
  if (ballPosY > 780){
   world.remove(ball2);
   ball2Exists = false;
 }
}
 if (((frameCount) % randomTime[randtime]) == 0) {
     vel = int(random(3));
     randball = int(random(10));
     randpos = int(random(6));
     randtime = int(random(3));
    while(true){
      if (vel == velCurrent){
        vel = int(random(3));
      }else{
        velCurrent = vel;
        ballSpeed = randomVel1[velCurrent];
        break;
      }
    }
    while(true){
      if (randpos == posCurrent){
        randpos = int(random(6));
      }else{
        posCurrent = randpos;
        ballX = randomPos[posCurrent];
        break;
      }
    }
 if (randball < 6){
     ball1 = new FCircle(15);
     ball1.setGroupIndex(-1);
     ball1.setNoStroke();
     ball1.setFill(255,0,0);
     ball1.setPosition(randomPos[posCurrent],50);
     ball1.setVelocity(0,randomVel1[velCurrent]);
     ball1.setRestitution(1);
     ball1.setDamping(0);
     world.add(ball1);
```

```
      ballTime = millis() - playingTime;
      ball1Exists = true;
      ball.trigger();
     }else{
      ball2 = new FCircle(15);
      ball2.setGroupIndex(-1);
      ball2.setNoStroke();
      ball2.setFill(0,0,255);
      ball2.setPosition(randomPos[randpos],50);
      ball2.setVelocity(0,randomVel1[velCurrent]);
      ball2.setRestitution(1);
      ball2.setDamping(0);
      world.add(ball2);
      ballTime = millis() - playingTime;
      ball2Exists = true;
      ball.trigger();
      }
   }
   world.step();
   world.draw(this);
  }


 void drawLine(){
   noFill();
   stroke(255);
   line(width/2-30,600,width/2+30,600);
   line(width/2,570,width/2,630);
}

void updateScore(int scorer){
     if(scorer == RED){
       red_state++;
     }
     if(scorer == BLU){
       blu_state++;
     }
}

void drawScore() {
   textFont(adore,32);
   fill(255,0,0);
   text(red_state, 207, height/2);
   fill(0,0,255);
   text(blu_state, 800, height/2);
   noFill();
   stroke(255,0,0);
   strokeWeight(2);
}

void drawAnimatedBall(){
   fill(255,0,0);
   ellipse(px,py,10,10);

   px = width/2 + cos(radians(angle))*(radius/1.5);

   py = 600 + sin(radians(angle))*(radius/1.5);
```

```
  angle2 = 0;

  for (int i = 0; i < width; i++){
     px2 = width/8 + cos(radians(angle2))*(radius/2);
     py2 = width/8 + sin(radians(angle2))*(radius/2);
     angle2 -= frequency2;
  }

  angle -= frequency;
  x += 1;
}

void addTuioCursor(TuioCursor tcur) {
if ( !is_a && !is_b)
{ a = tcur;
  is_a = true;
  return;
}
if ( !is_a && is_b)
{ a = tcur;
  is_a = true;
  is_t = true;
  //calculate x,y,angle of my_obj
  float x = (a.getX() + b.getX())/2.0;
  float y = (a.getY() + b.getY())/2.0;
  float angle = atan2(a.getX() - b.getX(),b.getY() - a.getY()) - PI/2.0*-1;
  my_obj.update(new TuioTime(frameCount),x,y,angle);
  //addTuioObject(my_obj);
  paddle = new FBox(60, 15);
  paddle.setNoStroke();
  paddle.setFill(255);
  paddle.setStaticBody(true);
  paddle.setRestitution(1);
  world.add(paddle);
  return;
}
if ( is_a && is_b)
{return; }
if ( is_a && !is_b)
{ b = tcur;
  is_b = true;
  is_t = true;
  //calculate x,y,angle of my_obj
  float x = (a.getX() + b.getX())/2.0;
  float y = (a.getY() + b.getY())/2.0;
  float angle = atan2(a.getX() - b.getX(),b.getY() - a.getY()) - PI/2.0*-1;
  my_obj.update(new TuioTime(frameCount),x,y,angle);
  //addTuioObject(my_obj);
  paddle = new FBox(60, 15);
  paddle.setNoStroke();
  paddle.setFill(255);
  paddle.setStaticBody(true);
  paddle.setRestitution(1);
  world.add(paddle);
  return;
 }
}
```

```
void removeTuioCursor(TuioCursor tcur){
if (is_a && !is_b){
  is_a = false;
  //removeTuioObject(my_obj);
  world.remove(paddle);
  return;
}
if (!is_a && is_b){
  is_b = false;
  //removeTuioObject(my_obj);
  world.remove(paddle);
  return;
}
if (!is_a && !is_b){
  return;
  }
if ( is_a && is_b){
  is_b = false;
  //removeTuioObject(my_obj);
  world.remove(paddle);
  return;
  }
 }


/*void addTuioObject(TuioObject my_obj){
  paddle = new FBox(60, 15);
  paddle.setNoStroke();
  paddle.setFill(255);
  paddle.setStaticBody(true);
  paddle.setRestitution(1);
  paddle.setPosition(my_obj.getScreenX(width),my_obj.getScreenY(height));
 }*/

void updateTuioCursor (TuioCursor tcur){
  if (is_a && is_b){
    float x = (a.getX() + b.getX())/2.0;
    float y = (a.getY() + b.getY())/2.0;
    float angle = atan2(a.getX() - b.getX(),b.getY() - a.getY()) - PI/2.0*-1;
    my_obj.update(new TuioTime(frameCount),x,y,angle);
    paddle.setRotation(my_obj.getAngle());
    paddle.setPosition(my_obj.getScreenX(width),my_obj.getScreenY(height));
    //updateTuioObject(my_obj);
  }
}


void contactStarted(FContact contact) {
  FBody body1 = contact.getBody1();
  FBody body2 = contact.getBody2();

  FBody b = null;
  if (contact.getBody1() == paddle) {
    b = contact.getBody2();
  }
  if (b == null) {
    return;
  }
```

```
  world.remove(b);
 }


void contactResult(FContactResult result) {
  if ((result.getBody1() == paddle && result.getBody2() == ball1) ||
      (result.getBody2() == paddle && result.getBody1() == ball1)){
    updateScore(RED);
    ball1Exists = false;
    score.trigger();
  } else if ((result.getBody1() == paddle && result.getBody2() == ball2) ||
             (result.getBody2() == paddle && result.getBody1() == ball2)){
    updateScore(BLU);
    ball2Exists = false;
    score.trigger();
   }
 }

void startSplash(){
  int seconds = (millis() - startingTime) / 1000;
  textFont(f,25);
  fill(255);
  textAlign(CENTER);
  text("You're about to start the experiment.",width/2 , 190);
  textFont(f,16);
  fill(255);
  textAlign(CENTER);
  text("RED and BLU balls will drop from the top of the screen at random positions.",width/2 ,
230);
  text("Your task will be to intercept the balls with the white paddle ,",width/2 , 260);
  fill(255,0,0);
  text("Try to be as rapid and as quick as possible.",width/2 , 290);
  text("Try to intercept as many ball as possible.",width/2 , 320);
  fill(255);
  text("You will have 5 minutes on the whole to accomplish the task.",width/2 , 350);
  text("Please try to be always concentrated.",width/2 , 380);
  textFont(f,20);
  fill(255);
  textAlign(CENTER);
  text("To start the experiment place the fingers on the white paddle.",width/2 , 410);
  noStroke();
  fill(255);
  rect(width/2-30,600-5,60,10);
  if(is_t == true){
    red_state = 0;
    blu_state = 0;
    state = PLAYING;
    alfa = 200;
    totalTime = 350;
    startingTime = millis();
    playingTime = millis() - startingTime;
    start.trigger();
 }
}

void endSplash() {
    fill(255);
    textFont(adore,22);
```

```
    textAlign(CENTER);
    text("Time elapsed", width/2, height/2);
    text("Thank you for participating ;)", width/2, height/2 + 30);
}

void stop()
{
  bop.close();
  ball.close();
  score.close();
  start.close();
  count.close();
  minim.stop();

  super.stop();
}
```

1.4 Processing code for application used in Task 2, implementation of Tangible control

```
import fullscreen.*;
import processing.opengl.*;
import javax.media.opengl.GL;
import ddf.minim.*;
import fisica.*;

import TUIO.*;

Minim minim;
AudioSample bop;
AudioSample ball;
AudioSample score;
AudioSample start;
AudioSample wrong;
AudioPlayer count;

FWorld world;
FBox paddle;
FCircle ball1, ball2;
FBody body1, body2;

TuioProcessing tuioClient;
TuioObject myobj;

FullScreen fs;

PGraphicsOpenGL pgl;
GL gl;

final int RED = 1;
final int BLU = 0;
final int START = 0;
final int PLAYING = 1;
final int END = 2;

int red_state;
```

```
int blu_state;
int g_calibration_mode = 0;
int totalTime;
int startingTime;
int playingTime;
int state;
int ballTime;

int[] randomVel1 = {600,800,900};
int[] randomVel2 = {1000,1100,1200};
int[] randomVel3 = {1300,1400,1500};
int[] randomPos = {300,350,400,600,650,700};
int[] randomTime = {120,240,480};
int vel;
int randball;
int randpos;
int randtime;
int velCurrent = 0;
int posCurrent = 0;
int ballX;
int ballSpeed;
float ballPosY;
boolean ball1Exists = false;
boolean ball2Exists = false;

float g_ax = 0.0, g_ay = 0.0, g_az = 0.0;
float g_sx = 1.0, g_sy = 1.0;
float g_ox = 0.0, g_oy = 0.0;
float alfa = 50;
float px,py,px2,py2;
float angle,angle2;
float radius = 100;
float frequency = 3;
float frequency2 = 3;
float x;

boolean is_t = false;

PFont f;
PFont adore;
PrintWriter output;


static final boolean FULL_SCREEN = true;
//static final boolean DUAL_VIEW   = true;

//static final boolean FULL_SCREEN = false;
static final boolean DUAL_VIEW  = false;
static final boolean DEBUG = false;

static final float M_PI  = (float)Math.PI;
static final float M_PI2 = (float)Math.PI*2.0;
//
// Calibration values
//
static final int OFFSET_XY = 1;
static final int ROTATE_XY = 2;
static final int ROTATE_Z  = 3;
```

```
static final int SCALE_XY   = 4;



// Load / Store Calibration Parameters
// Thx to http://www.vbforums.com/showthread.php?t=308145
void CalibrationSave(String filename)
{
float[] params = { g_ax, g_ay, g_az,    g_sx, g_sy,    g_ox, g_oy };


      try {
         FileOutputStream fos = new FileOutputStream(filename);
         ObjectOutputStream out = new ObjectOutputStream(fos);
         out.writeObject(params);
         out.flush();
         out.close();
         println("Calibration file saved.");
      }
      catch (IOException e)
      {
          println(e);
      }
}


void CalibrationLoad(String filename)
{
       try {
         FileInputStream fis = new FileInputStream(filename);
         ObjectInputStream in = new ObjectInputStream(fis);
         float[] p = (float[])in.readObject();

         g_ax = p[0];
         g_ay = p[1];
         g_az = p[2];
         g_sx = p[3];
         g_sy = p[4];
         g_ox = p[5];
         g_oy = p[6];

         in.close();

         println("Calibration file loaded.");
       }
       catch (Exception e)
       {
           println(e);
       }
}

void DrawCalibrationGrid()
{
 float tx = width/2.5;
 float ty = height/2.5;
 float tz = -1;

// stroke(0,256,0);
```

```
// fill(0,256,0);
stroke(255);
fill(255);

 for(int i=0; i<7; i++)
 {
  float ix = width*i/6.0;
  float iy = height*i/6.0;
  line(ix, 0, ix, height);
  line(0, iy, width, iy);
 }

 stroke(255,0,0);
 fill(255,0,0);

  if(g_calibration_mode == OFFSET_XY)
    text("> OFFSET XY <", tx, ty, tz);

  if(g_calibration_mode == ROTATE_XY)
    text("> ROTATE XY <", tx, ty, tz);

  if(g_calibration_mode == ROTATE_Z )
    text("> ROTATE Z <",  tx, ty, tz);

  if(g_calibration_mode == SCALE_XY )
    text("> SCALE XY <",  tx, ty, tz);
}

void keyPressed()
{

if(key == 'r') // Reset
{
 g_ax = 0.0; g_ay = 0.0; g_az = 0.0;
 g_sx = 1.0; g_sy = 1.0;
 g_ox = 0.0; g_oy = 0.0;
}

if(key == 's') // Save to file
{
 CalibrationSave("calibration.dat");
}

if(key == 'l') // Load from file
{
 CalibrationLoad("calibration.dat");
}


if(key == ENTER)
{
  g_calibration_mode = (g_calibration_mode + 1) % 5;
}

if (key != CODED)
{
  return;
}
```

```
if(g_calibration_mode == OFFSET_XY)
{
  if (keyCode == LEFT)   g_ox-=2.5;
  if (keyCode == RIGHT) g_ox+=2.5;
  if (keyCode == UP)     g_oy-=2.5;
  if (keyCode == DOWN)   g_oy+=2.5;
}
if(g_calibration_mode == ROTATE_XY)
{
  if (keyCode == LEFT)   g_ax-=0.025;
  if (keyCode == RIGHT) g_ax+=0.025;
  if (keyCode == UP)     g_ay-=0.025;
  if (keyCode == DOWN)   g_ay+=0.025;
}
if(g_calibration_mode == ROTATE_Z)
{
  if (keyCode == LEFT)   g_az-=0.0125;
  if (keyCode == RIGHT) g_az+=0.0125;
}
if(g_calibration_mode == SCALE_XY)
{
  if (keyCode == LEFT)   g_sx-=0.025;
  if (keyCode == RIGHT) g_sx+=0.025;
  if (keyCode == UP)     g_sy-=0.025;
  if (keyCode == DOWN)   g_sy+=0.025;
}

}


// Original idea by Daniel Gallardo
void ApplyCalibrationMatrix()
{
int g_w = 1024;
int g_h = 768;

float fov = PI/3.0;
float cameraZ = (height/2.0) / tan(fov/2.0);

//perspective(fov, float(g_w)/float(g_h),
//            0.0, 100.0);

    translate(g_ox,
              g_oy,
                 -4.0f);

    translate(g_w/2, g_w/2, 0);

     rotateX(g_ax);
     rotateY(g_ay);

     rotateZ(g_az);

     scale(g_sx, g_sy, 1);


    translate(-g_w/2, -g_h/2, 0);
```

```
}


// these are some helper variables which are used
// to create scalable graphical feedback
float cursor_size = 20;
float object_size = 40;
float table_size = 480;
float scale_factor = 1.2;

PFont font;


// Don't show frame, as in http://processing.org/hacks/hacks:undecoratedframe
void init()
{
  if(FULL_SCREEN)
  {
    frame.removeNotify();
    frame.setUndecorated(true);
    frame.addNotify();
  }

  super.init();
}


void setup()
{
  output = createWriter("S2_T/ID_12.txt");
  startingTime = millis();
  state = START;

  totalTime = 10000;

  red_state = 0;
  blu_state = 0;
  adore = loadFont("Adore64-30.vlw");
  f = loadFont("Verdana-48.vlw");
  textSize(28);

  minim = new Minim(this);

  bop = minim.loadSample("Bop.aiff", 512);
  ball = minim.loadSample("Ball.aiff", 512);
  score = minim.loadSample("Score.aiff", 512);
  start = minim.loadSample("Start.aiff", 512);
  wrong = minim.loadSample("Wrong.aiff", 512);
  count = minim.loadFile("Countr.wav", 2048);

  tuioClient  = new TuioProcessing(this);

  Fisica.init(this);

  world = new FWorld();
  world.setGravity(0,0);
```

```
  if(FULL_SCREEN)
    size(1024,768, OPENGL); // <-- resolution of the other screen here
     else
    size(1024,768, OPENGL); // <-- whatever fits the desktop

frameRate(60);

  if(DUAL_VIEW)
  {
    delay(5000); // wait for the window to be set up...
    frame.setLocation(screen.width, 0); // Move it on the other screen
  }
  else
  {
     frame.setLocation(0, 0);
  }

 // disable 2xFSAA   (enabled by default in newer Processings)
  hint(DISABLE_OPENGL_2X_SMOOTH);
  // hint(ENABLE_OPENGL_4X_SMOOTH)

  hint(DISABLE_OPENGL_ERROR_REPORT);
  hint(DISABLE_DEPTH_TEST);

 // Enable GL_*_SMOOTH hints
  smooth();

  noStroke();
  fill(0);

  hint(ENABLE_NATIVE_FONTS);
  font = createFont("Arial", 32);
  scale_factor = height/table_size;

  strokeWeight(4); // nicer

  // an instance of the TuioClient
  // since we add "this" class as an argument, the TuioClient expects
  // an implementation of the TUIO callback methods (see below)

  // 3333 is default in reactivision (and due to a bug, it seems that it can NOT be changed ?)
  //tuioClient  = new TuioProcessing(this, 3333);

  CalibrationLoad("calibration.dat");

  myobj = new TuioObject(0,0,0,0,0);


output.println("ID_01\tMin\tSec\tMillis\tRed\tBlu\tSpeed\t\tAccel\t\tXpos\tYpos\tBallX\tBallY\tBall
Speed");
}


//=================================================================
============
// within the draw method we retrieve an array of TuioObject and TuioCursor
// from the TuioClient and then loop over both lists to draw the graphical feedback.
void draw()
```

```
{


ApplyCalibrationMatrix();

  fill(0,alfa);
  noStroke();
  rect(0,0,width,height);

  int milliseconds = (millis() - startingTime);
  int seconds = (millis() - startingTime) / 1000;
  int minutes = seconds / 60;

  if (seconds > totalTime){
    state = END;
    output.close();
  }
  if (seconds == totalTime - 8){
    count.play();
  }
  if (seconds > 180){
    ball1.setVelocity(0,randomVel2[vel]);
    ball2.setVelocity(0,randomVel2[vel]);
    ballSpeed = randomVel2[velCurrent];
  }
  if (seconds > 290){
    ball1.setVelocity(0,randomVel3[vel]);
    ball2.setVelocity(0,randomVel3[vel]);
    ballSpeed = randomVel3[velCurrent];
  }


  switch(state) {
    case START:
      startSplash();
      drawAnimatedBall();
      break;
    case PLAYING:
      drawBall1();
      //drawBall2();
      drawScore();
      drawLine();
      if(ball1Exists || ball2Exists){
      output.println(    "\t"+minutes+"\t" +seconds+"\t"+(millis() -
ballTime)+"\t"+red_state+"\t"+blu_state+"\t"

+int(myobj.getMotionSpeed()*pow(10,2))/pow(10,2)+"\t\t"+int(myobj.getMotionAccel()*pow(10,2)
)/pow(10,2)+"\t\t"
      +myobj.getScreenX(width)+"\t"+myobj.getScreenY(height)+"\t"+ballX+
"\t"+int(ballPosY)+"\t"+ballSpeed);
      output.flush();
      }
      break;
    case END:
      endSplash();
      break;
 }
```

```
if(g_calibration_mode > 0)
{
  DrawCalibrationGrid();
 }
}

void drawBall1() {
 if (ball1Exists){
  ballPosY = ball1.getY();
  if (ballPosY > 780){
   world.remove(ball1);
   ball1Exists = false;
  }
 }
 if (ball2Exists){
  ballPosY = ball2.getY();
  if (ballPosY > 780){
   world.remove(ball2);
   ball2Exists = false;
 }
}
 if (((frameCount) % randomTime[randtime]) == 0) {
    vel = int(random(3));
    randball = int(random(10));
    randpos = int(random(6));
    randtime = int(random(3));
   while(true){
     if (vel == velCurrent){
       vel = int(random(3));
     }else{
       velCurrent = vel;
       ballSpeed = randomVel1[velCurrent];
       break;
     }
   }
   while(true){
     if (randpos == posCurrent){
       randpos = int(random(6));
     }else{
       posCurrent = randpos;
       ballX = randomPos[posCurrent];
       break;
     }
   }
    if (randball < 6){
    ball1 = new FCircle(15);
    ball1.setGroupIndex(-1);
    ball1.setNoStroke();
    ball1.setFill(255,0,0);
    ball1.setPosition(randomPos[posCurrent],50);
    ball1.setVelocity(0,randomVel1[velCurrent]);
    ball1.setRestitution(1);
    ball1.setDamping(0);
    world.add(ball1);
    ballTime = millis() - playingTime;
    ball1Exists = true;
    ball.trigger();
   }else{
```

```
    ball2 = new FCircle(15);
    ball2.setGroupIndex(-1);
    ball2.setNoStroke();
    ball2.setFill(0,0,255);
    ball2.setPosition(randomPos[randpos],50);
    ball2.setVelocity(0,randomVel1[velCurrent]);
    ball2.setRestitution(1);
    ball2.setDamping(0);
    world.add(ball2);
    ballTime = millis() - playingTime;
    ball2Exists = true;
    ball.trigger();
    }
  }
  world.step();
  world.draw(this);
 }

 void drawLine(){
  noFill();
  stroke(255);
  line(width/2-30,600,width/2+30,600);
  line(width/2,550,width/2,650);
}

void updateScore(int scorer){
    if(scorer == RED){
      red_state++;
    }
    if(scorer == BLU){
      blu_state++;
  }
}

void drawScore() {
  textFont(adore,32);
  fill(255,0,0);
  text(red_state, 207, height/2);
  fill(0,0,255);
  text(blu_state, 800, height/2);
  noFill();
  stroke(255,0,0);
  strokeWeight(2);
}

void drawAnimatedBall(){
  fill(255,0,0);
  ellipse(px,py,10,10);

  px = width/2 + cos(radians(angle))*(radius/1.5);

  py = 600 + sin(radians(angle))*(radius/1.5);

  angle2 = 0;

  for (int i = 0; i < width; i++){
     px2 = width/8 + cos(radians(angle2))*(radius/2);
     py2 = width/8 + sin(radians(angle2))*(radius/2);
```

```
      angle2 -= frequency2;
  }

  angle -= frequency;
  x += 1;
}

void contactStarted(FContact contact) {
  FBody body1 = contact.getBody1();
  FBody body2 = contact.getBody2();

  FBody b = null;
  if (contact.getBody1() == paddle) {
    b = contact.getBody2();
  }
  if (b == null) {
    return;
  }
  world.remove(b);

 }


void contactResult(FContactResult result) {
  if ((result.getBody1() == paddle && result.getBody2() == ball1) ||
       (result.getBody2() == paddle && result.getBody1() == ball1)){
    updateScore(RED);
    ball1Exists = false;
    score.trigger();
  } else if ((result.getBody1() == paddle && result.getBody2() == ball2) ||
              (result.getBody2() == paddle && result.getBody1() == ball2)){
    updateScore(BLU);
    ball2Exists = false;
    score.trigger();
   }
 }

void addTuioObject(TuioObject tobj) {
  is_t = true;
  paddle = new FBox(50,15);
  paddle.setFill(255);
  paddle.setNoStroke();
  paddle.setRestitution(1);
  paddle.setStaticBody(true);
  world.add(paddle);
}

void updateTuioObject (TuioObject tobj){
  myobj.update(new
TuioTime(frameCount),tobj.getX(),tobj.getY(),tobj.getXSpeed(),tobj.getYSpeed(),tobj.getMotionAccel
());
  paddle.setPosition(tobj.getScreenX(width),tobj.getScreenY(height));
  paddle.setWidth(60);
}

void removeTuioObject (TuioObject tobj) {
  world.remove(paddle);
}
```

```
void startSplash(){
  int seconds = (millis() - startingTime) / 1000;
  textFont(f,25);
  fill(255);
  textAlign(CENTER);
  text("You're about to start the experiment.",width/2 , 190);
  textFont(f,16);
  fill(255);
  textAlign(CENTER);
  text("RED and BLU balls will drop from the top of the screen at random positions.",width/2 ,
230);
  text("Your task will be to intercept the balls with the white paddle ,",width/2 , 260);
  fill(255,0,0);
  text("Try to be as rapid and as quick as possible.",width/2 , 290);
  text("Try to intercept as many ball as possible.",width/2 , 320);
  fill(255);
  text("You will have 5 minutes on the whole to accomplish the task.",width/2 , 350);
  text("Please try to be always concentrated.",width/2 , 380);
  textFont(f,20);
  fill(255);
  textAlign(CENTER);
  text("To start the experiment place the tangible on the white paddle.",width/2 , 410);
  noStroke();
  fill(255);
  rect(width/2-30,600-5,60,10);
  if(is_t == true){
    red_state = 0;
    blu_state = 0;
    state = PLAYING;
    alfa = 200;
    totalTime = 350;
    startingTime = millis();
    playingTime = millis() - startingTime;
    start.trigger();
 }
}

void endSplash() {
    fill(255);
    textFont(adore,22);
    textAlign(CENTER);
    text("Time elapsed", width/2, height/2);
    text("Thank you for participating ;)", width/2, height/2 + 30);
}

void stop()
{
  bop.close();
  ball.close();
  score.close();
  start.close();
  count.close();
  minim.stop();

  super.stop();
}
```