Conceptual Integration and User Interface Metaphor for the Multi-Touch Control of Recorded Audio

Bruno Angeles

Music Technology Area
Schulich School of Music
McGill University
Montreal, Canada

December 2012

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Arts.

Abstract

The majority of touch-enabled musical production software tend to use metaphors from professional musical studio environments in their interface (e.g., a rackmount, turntables plus a crossfader, or a keyboard) or simply use single-finger input as a computer mouse. We identify a need for musical software that benefits from novel graphical user interface elements and innovative metaphors to provide control over pre-recorded music. We hypothesize that a software design approach using conceptual integration, or blending, will lead to new ludic interfaces for musical expression with the potential to facilitate DJ tasks.

Multi-touch technology offers the promise of going beyond traditional mouse-based user interfaces, and is especially pertinent in that it provides full embodiment: the user interacts directly with the visual feedback of the system. This change in paradigm has implications in software design, not yet fully understood in tools for musical expression.

This thesis first documents the existing methods of implementing multi-touch technology, before suggesting a taxonomy of multi-touch devices. A literature review of multi-touch systems for musical applications is also presented, after which metaphor and blending (also known as conceptual integration) are discussed. We apply blending to software design for multi-touch musical software and introduce our programming framework, *TactoSonix*.

Résumé

Dans leur interface utilisateur, la plupart des logiciels de production musicale pour écrans tactiles multipoints emploient des métaphores issues des studios de musique professionnels dans leur interface utilisateur (châssis à effets de guitare, tourne-disques, potentiomètres rectilignes, clavier), ou utilisent un doigt qui joue le rôle de souris. Cela nous a amenés à constater que ces logiciels gagneraient à inclure des éléments innovateurs au niveau de l'interface utilisateur et à employer des métaphores inexplorées dans un contexte de contrôle de morceaux de musique pré-enregistrée. Nous partons de l'hypothèse que la conception de logiciel assistée par l'intégration conceptuelle (*blending*) permettra le développement d'interfaces ludiques pour l'expression musicale et facilitera certaines tâches des DJs.

La technologie tactile multipoints nous permet d'imaginer des interfaces de logiciels non-traditionnelles, car elle combiner les interfaces de rétroaction visuelle et de contrôle du système. Ce changement de paradigme nous oblige à réévaluer la conception des logiciels de production musicale, et à étudier ses applications dans le domaine de l'expression musicale.

Ce mémoire répertorie les méthodes qui permettent actuellement de concevoir des dispositifs tactiles multipoints dont il propose une taxonomie. Il présente une analyse de la littérature des systèmes tactiles multipoints dans des contextes musicaux. Les concepts de métaphore et d'intégration conceptuelle (*blending*) sont étudiés puis appliqués à la conception de notre plate-forme de programmation musicale tactile multi-points, *TactoSonix*.

Acknowledgments

The author would like to dedicate this thesis to his parents Jorge and Anne-Marie in gratitude for their motivation and support, as well as for always insisting on the value of education and academic dedication.

The author would like to thank his supervisor Marcelo Wanderley for his continuous guidance, valuable advice, and patience, and his colleagues at McGill University's IDMIL, most notably Gabriel Vigliensoni, Joseph Malloch, Stephen Sinclair, and Marlon Schumacher for their advice. Martin Shank, colleague of the author, made several suggestions for the improvement of the user interface.

The author is extremely grateful to Olivier Plante, who graciously offered his time and industrial design expertise for the design of a DIY multi-touch table; this device was useful in the development and testing of software as part of this thesis.

Contents

List of Figures

List of Tables

List of Acronyms

CPU	Central Processing Unit
DIY	Do It Yourself
DJ	Disk Jockey
DOF(s)	Degree(s) of Freedom
DSP	Digital Signal Processing
FM	Frequency Modulation
FPS	Frames Per Second
GUI	Graphical User Interface
HCI	Human-Computer Interaction
IDMIL	Input Devices and Musical Interaction Laboratory
LED	Light-Emitting Diode
OSC	Open Sound Control
TUIO	Tangible User Interface Objects
WIMP	Window, Icon, Menu, Pointing Device

Chapter 1

Introduction and Context

The topic of this thesis is the implementation of a recreative music production software environment for multi-touch devices. We intend to create an environment that simplifies certain tasks of mixing music in order to ensure the musical coherence of the resulting mix. We focus on multi-touch technology because of its ability to combine the input and output devices, and hope it makes our ludic music production environment more accessible than traditional mouse and keyboard input.

To this end, we first document the methods of creating multi-touch device and present a taxonomy of multi-touch devices. Our attention then turns to a discussion on the implications of multi-touch input to software design and a literature review of multi-touch musical software environments. We proceed to the field of cognition with a discussion of metaphor and its application to software design. *TactoSonix*, our software environment, is presented as an illustration of multi-touch software design based on metaphor.

The current chapter introduces several concepts related to multi-touch technology, while laying out the motivation behind the reported work.

1.1 Multi-touch Technology

1.1.1 Definition

Multi-touch technology concerns both hardware and software that use finger input or the manipulation of physical objects, as opposed to the mouse and keyboard of traditional computers. Multi-touch interaction can be implemented with different methods and configurations, described in Chapter 2.

Such devices offer the promise of moving beyond traditional WIMP (Windows, Icons, Menu, Pointing Device) interfaces [6]. Multi-touch devices have been used in different contexts, but our research focuses on musical applications.

1.1.2 Distinctions

Important distinctions must be made at the level of visual feedback between touch tablets and touch screens, to follow the nomenclature presented by Buxton et al. [7]. Touch screens provide visual input to the user, while touch tablets do not. Another possible distinction can be made in the display method of different touch screens (e.g., LCD, projector, pico projector, plasma).

The sensing method for finger position results in another distinction. We have documented seven different methods for the implementation of multi-touch hardware: camera-based solutions, capacitive sensing, resistive sensing, magnetic sensing, acoustic sensing, optical fibre, and dispersive signal technology (see Chapter 2). Although these methods may not have much meaning to the user, the software designer must take them into account because of their individual features and restrictions.

A third distinction stems from the support, or lack thereof, for tangible objects (also known as graspables [8]). These are objects that can be placed on the interaction surface to be recognized by the system. The most popular multi-touch system nowadays with support for graspables is the *reacTable* [9].

1.1.3 Historical Background

Single-touch devices have been around since the 1970s. The *PLATO IV* system at the University of Illinois was implemented between 1972 and 1976 [10]. In 1978, a single-touch system [11] using strain gauges and providing visual feedback was reported, while Sasaki et al. [12] performed early research in using single-touch pressure-sensitive tablets for musical applications in 1981. In terms of multi-touch devices, Lee, Buxton, and Smith disclosed a force-sensitive system with capacitive sensing in 1985; this system determined pressure based on the deformation of the touch surface and the “intrinsic spreading of the compressible pressure tip” [13].

Moog’s multiply touch-sensitive clavier has a design that was “based on a conventional wood action keyboard" [14], first described in 1982. In 2005, a few months before Robert Moog's passing, he presented with Eaton a prototype device registering five degrees of freedom per key [15]. Not all controls could be controlled simultaneously, since the values tracked were "a) the left-to-right position of the player's finger, b) the front-to-back position of the player's finger, c) the area of the surface of the player's finger that is touching the key surface, d) the depth to which the key is depressed, and e) the force with which the key is depressed after it reaches the lower limit of its travel" [15]. This promising system does not seem to have progressed beyond the status of prototype. Another keyboard-like controller with multi-touch control is Haken's *Continuum* [16][2].

In the mid-1990s, Pinkston et al. reported a setup using four plastic sheets equipped with 128 Force Sensing Resistors (FSRs), for placement under a dance floor [17]. Although not meant for manual interaction, the system was an investigation into the possibilities of force-sensitive multi-touch technology. In 2009, Wessel performed one of his composition for the SLABS controller; SLABS consists of 24 or 32 (there are two models) single-touch pressure-sensitive pads that send XY coordinates and pressure information over OSC. The control data for the three degrees of freedom of the controller is sent at an audio rate, which "provides for a high degree of control intimacy" [18]. SLABS 32 allows multi-touch interaction by presenting the pressure pads as an array of eight by four sensors, assembled tightly.

Jeff Han is often credited with popularizing low-cost multi-touch devices, largely due to the engaging software demos he presented in a breakthrough conference at TED2005. However, Johnstone's *Rolky* [1] was already using the same physical concept, namely Frustrated Total Internal Reflection (FTIR), some 20 years earlier. The *Rolky* is shown in Figure 2.5. Han visited past applications of FTIR in [19] and described his own setup.

1.1.4 Why Multi-Touch?

We argue that the reason for the popularity of multi-touch interaction (observable in the variety of tablets on the commercial market) lies in what Fishkin calls "full embodiment": "the output device *is* the input device" [20], as opposed to traditional interaction with a computer monitor, mouse, and keyboard.

Multi-touch interaction is essentially a system with two degrees of freedom (the X and Y coordinates of each contact point), which is extended to three degrees of freedom in pressure-sensitive implementations.

In comparing conventional hardware and a “soft machine” [21] implemented as a touch screen with visual feedback, Nakatani and Rohrlich argue that soft machines overcome the limitations of hard machines, namely inflexibility and the management of complexity. Software allows the design of evolving interfaces (“progressive disclosure of controls” [21]), which overcomes these limitations. We argue that this “progressive disclosure of controls”, discussed in 1983, was overlooked in favour of the WIMP paradim visible in Microsoft Windows, Apple OS, and most Linux GUIs since. A reason for this choice is that the emergence of personal computers in the 1980s and 1990s occured through office settings, and office tasks do not require software that evolves through different states, whereas tasks such as gaming and music production do (progress in a word processor is continuous, unlike the discrete states visible in video games). Our view is that the “progressive disclosure of controls” should be favoured in multi-touch software, given that it strengthens the immersivity of the user experience by getting rid of the intermediary that WIMP menus represent.

1.1.5 A Confusing Third Degree of Freedom

Many papers report systems that are said to be “pressure-sensitive” [22] [23] [18], while others use the term “force-sensitive” [24] [25]. All such systems intend to measure the amount of movement intended by the user in the axis perpendicular to the surface. Force cannot be measured directly: it is through its effect (e.g. the displacement of an elastic body) that force can be estimated.

1.1.6 Multi-touch vs Tangibles

Multi-touch systems like the *reacTable* [9] and *DiamondTouch* [26] allow the recognition of objects on the touch surface. Not all systems are capable of such recognition, our research focusing more on the multi-touch capabilities of systems than on their support for tangibles. Fiducials[1] are graphical patterns that can be identified via computer vision.

1.1.7 Issues in Multi-touch Interaction

Sousa and Matsumoto [27] identified occlusion (when a hand hides parts of the visual feedback), missing states (inability to detect hovering motion until contact), lack of preci-

[1]http://reactivision.sourceforge.net, accessed 2010/11/17

sion [sic], and pointer stability (jitter) as "natural restrictions when interacting with touch screens". A solution to the low resolution of such interaction can be achieved through dual-touch interaction — see Section 3.1.6. By "precision", both papers mean resolution, the minimum size measured in the interaction, whereas "precision" means repeatability or reproducibility.

Previously, Buxton [7] had also identified friction and the lack of haptic feedback as issues in touch tablets. Although multi-touch devices can sense and track fingers, they cannot natively identify which finger of the hand is touching the device. Multi-touch systems enhanced with above-table hand tracking (often performed via computer vision) could be able of such identification.

As can be seen in Chapter 2, multi-touch hardware can provide visual feedback in two ways: front projection and rear projection (it is also possible not to have any visual feedback). It can be argued that occlusion is an issue in both front-projected and rear-projected setups: fingers can block the user's view in both cases, but it is only in front-projected setups that the user's hands can block the visual feedback and not its perception by the performer.

1.1.8 Haptic Feedback

At the moment, localized haptic feedback on multi-touch devices is not yet widespread, although many portable touch-enabled devices include a haptic actuator. Device-wide haptic feedback is already possible on commercially-available mobile phones. *MudPad* [28] is an example of a system implementing localized active haptic feedback. Verrillo's work from the 1960s to the 1990s [29] explored the detectability thresholds of human vibrotactile sensation and the use of such feedback in controlling a musical instrument. Interestingly, Verrillo demonstrated variations in sensitivity to haptic feedback in different age groups. He also observed that most studies of the tactile sensation are based on the hand and arm, members that are involved in sound production and control in most instruments.

Chafe [30] described the psychophysics of vibrotactile sensation in humans by performing an experiment with a cellist wearing an accelerometer on a fingernail to control the oscillating nonlinear physical model of a brass instrument. Chafe's work emphasized the pertinence of vibrotactile feedback in new interfaces for musical expression. In music technology, Birnbaum and Wanderley [31], as well as Giordano and Wanderley [32], have previously studied applications of haptic feedback to musical instruments and controllers.

1.1.9 The Do It Yourself Community

Many DIY approaches for building multi-touch hardware are displayed on websites. Communities such as *NUI Group*[2] provide support to multi-touch enthusiasts. A technical guide with the descriptions of problems and their solutions, as well as assembly tips, was put together by Schöning et al. [33].

Smith et al.'s low-cost malleable system [22] builds upon Jeff Han's design [19], by reportedly reducing hysteresis and sensing touches of near-zero force.

A more advanced low-cost touch system is Bottoni et al.'s *TouchBox* [34], which used a single-touch screen and a DSP board. *Bricktable* [35] is a system that was assembled using low-cost materials. The authors provided a thorough explanation of the issues they faced, and solutions to problems such as infrared hot spots, low light diffusion in the projection, and the choice of a protective layer for the touch table.

Loviscach [36] reported a method using electronic components for converting an inexpensive single-touch screen overlay into a dual-touch system.

1.2 Music Mixing

1.2.1 Definition

We define DJing as the art or task of mixing music, commonly performed using turntables and musical software. Similarly, Carrascal and Jordà define audio mixing as "the adjustment of relative volumes, panning and other parameters corresponding to different sound sources, in order to create a technically and aesthetically adequate sound sum" [37], and Beamish et al. state that "A disc jockey (DJ)'s primary job is to play pre-recorde music at social occasions such as dances or weddings, and in performance venues like dance clubs. It was once sufficient to simply play one song after another, but at least one branch of DJing has evolved into an expressive art form where DJs mix song fragments using special interactive techniques, creating entirely new music in a live performance that is an auditory and visual spectacle." [38]. This type of DJing, referred to as scratching and turntablism, is not the focus of this thesis. Our interest resides in restricting the user to interactions

[2]http://nuigroup.com/forums, accessed 2010/11/17

that ensure an aesthetically pleasant sound. We do not hope to replicate DJing hardware in software so as to provide the same kind of complex control over sound, but rather to put in the hands of the average person some DJing tools (e.g., crossfader, frequency domain filtering) that would otherwise be out of their reach in terms of technical prowess or musical production expertise. For information about turntablism techniques, we refer the reader to [39] and [40]. A brief comparison of our software and traditional DJ equipment is presented in Section 5.4.4.

1.2.2 Musical Games

Ludic environments for the creation of music from predetermined sources can be traced back at least to 18th century dice games for the composition of "minuets, marches, waltzes, contredanses" in Europe [41]. These games allowed musical novices to compose music that could then be performed by musicians. These musical games were played by throwing two dice and using a lookup table to figure out which measure of the source composition to use in the new composition. Through careful (and restrictive) composition, a single piece of music could generate billions of possibilities. Played with two dice, the sum of which is always between two and 12 (giving 11 possibilities), and a game for 16 measures of music, there are 11ˆ16 possible outcomes, almost guaranteeing the unicity of each composition. It is worth noting that the probabilities of the different totals are not equal. Combined with the possibility of duplicates in the lookup table, this use of probabilities suggests a certain guidance to the composer of the game.

1.2.3 The Art of DJing

The task of controlling pre-recorded audio contents originated with the hardware solutions for radio transmission at the turn of the 20th century, but did not develop into a fully creative process until the advent of *musique concrète*, when Pierre Schaefer controlled the playback speed and used reverse playback as well as looping on his *Etude aux chemins de fer* [42] in the 1940s. The emergence of hip hop and DJ culture in the late 1970s cemented the role of turntables and mixers as legitimate instruments in this underground culture.

These DJ tasks can nowadays be assisted by software. Some products, such as *Native Instruments Traktor*[3], extend the functionalities of the initial DJ setup of two turntables

[3]http://www.native-instruments.com/#/en/products/?category=1316, accessed 2011/07/20

and a mixer with a combined hardware and software solution. The additional features include a loop recorder, sample triggers, audio effects, and "on-the-fly remixes". Another popular DJ tool, *Ableton Live*[4], allows users to arrange entire compositions, and has some of the features of *Traktor*, with the added functionality of mixing more than four tracks.

The features of a basic DJ setup are displayed in Table 1.1.

Feature	**Description**
Channels and tracks	DJs typically work with two different audio tracks. When more tracks are needed, they can be grouped in channels.
Crossfading	A single slider to control the weight of the two tracks being mixed.
Cues	DJs use cue points to skip to a particular part of a track. Cues do not necessarily have to fall on a beat, and playback continues after the cue.
Volume	Each track has a volume slider to control its amplitude.
Bass, mid, and high filters	Low pass, band pass, and high pass filters that can be applied on each track for quick frequency equalizing.
Audio effects	Aside from a simple equalizer, DJs also have access to audio effects such as a flanger, the gater effect, reverb, and a ring modulator.
Pitch shifter	DJs can change the pitch of a track using a slider without changing its tempo.
Tempo shifter	DJs can change the tempo of a track using a slider without changing its pitch.
Song syncing	DJs can sync the beat of different tracks. Traditionally performed by ear, this task is now automated in certain pieces of software.
Looping	DJs can create local loops inside a track. Loops can be manually defined, or use standard beat units.
Sequencing	Instead of recreating all elements of their mix during a performance, some DJs opt to use a sequencer to arrange pre-recorded musical elements, some of which may be of their own composition.

Table 1.1 The list of features offered by DJ equipment

[4]http://www.ableton.com/suite-8, accessed 2011/07/21

1.3 Hypothesis

We put forward the hypothesis that the use of user interface metaphors unrelated to conventional musical hardware, combined with innovative (non-WIMP) GUI controls, provides a basis for the development of multi-touch software that facilitates certain DJ tasks.

Our approach will be to use blending, or conceptual integration, to design new environments that will be less intimidating than professional music production systems by taking advantage of more familiar conceptual frames.

1.4 Thesis Contribution

The contribution of this thesis to the music technology community will be an investigation into the design of music mixing software using metaphor (a way of thinking about one domain in terms of another – see Section 4.1) and conceptual integration (similar to metaphor, but the two domains are applied to form a third and fourth one – see Section 4.1.3). We will also provide a taxonomy of multi-touch hardware, and a literature review of musical touch-based software environments.

1.5 Distribution of Chapters

This chapter discussed multi-touch technology, DJing, and our motivation. Chapter 2 describes a taxonomy of multi-touch hardware. Chapter 3 discusses the software considerations and possible GUI controls of multi-touch systems, as well as presenting a review of touch-enabled musical software. Chapter 4 investigates the concepts of metaphor and conceptual blending. Chapter 5 documents our software design approach. Chapter 6 concludes this document by summarizing our results and contributions, besides recommending further related work.

Chapter 2

Multi-touch Hardware: a Biaxial Taxonomy

This chapter introduces a taxonomy of multi-touch hardware based on two axes: the technology used to implement multi-touch capability, and the visual feedback provided. We also apply the taxonomy to existing devices, before introducing our own hardware.

2.1 Approach

We distinguish multi-touch devices from one another based on the technology used. We have identified the following techniques: camera-based solutions, capacitive sensing, resistive sensing, magnetic sensing, acoustic sensing, optical fibre, and dispersive signal technology. Although these methods describe the physical principles of interaction with the devices, a user might make different distinctions between them, essentially in the visual feedback provided. For this reason, we describe a bidimensional taxonomy: one axis is based on the technology used, while another refers to the visual feedback provided.

2.2 Previous Work

We point the reader towards Hurtienne and Israel's research [43] concerning taxonomies for tangible interfaces, and towards Kammer et al.'s work [44] for a taxonomy of multi-touch software. Hurtienne and Israel use orientational and ontological metaphors (see Section 4.1.2) to build their taxonomy, while Kammer et al. insist on technical features such as support for certain protocols and type of GUI widgets. Our taxonomy will focus on multi-touch rather than tangible interaction, and on hardware rather than software.

Well before the widespread use of multi-touch devices, Pickering [45] provided a review of technologies for touch sensing. Pickering describes systems using infrared energy, resistive sensing, capacitive sensing, and acoustic surface wave sensing. Pickering also mentions a system called “velocity touch sensing”, based on piezoelectric sensors but unable to track a moving finger on the surface.

In terms of hardware, Hinckley and Sinclair [46] provided an early taxonomy of touch-enabled devices at the turn of the twenty-first century, in addition to augmenting computer mice with capacitance sensors. This taxonomy identifies multi-touch devices as “multi-channel”. Its main axes are contact versus non-contact, and discrete versus continuous sensing.

2.3 Technological Taxonomy

2.3.1 Optical Solutions

Optical solutions provide a convenient approach to multi-touch sensing: the blobs’ positions and dimensions merely need to be extracted from camera frames, using common computer vision algorithms and filters, such as background removal (the difference between the live feed and a reference background with no activity). Some solutions use infrared energy to track blobs, while others use the visible spectrum. In addition to being widely available and affordable, most cameras inherently view the near infrared. In some cases, their infrared filter can readily be replaced by a visible light filter for better results. This makes this type of implementation a good choice for prototyping, the main obstacle then being the low frame rate of some of these cameras.

Front Diffused Illumination (FDI) is a method using the near-infrared spectrum and an infrared camera to detect finger positions. In it, infrared light from the environment is used to provide contrast on the touch surface where the fingers are placed. The camera vision software then needs to track the dark blobs. This is shown in Figure 2.1. This setup is so simple that it can be implemented with a piece of glass, thin paper, a webcam, and a cardboard box to host the camera. Its main drawback is the specific lighting environment required for proper functioning, although it is true that this restriction applies to most

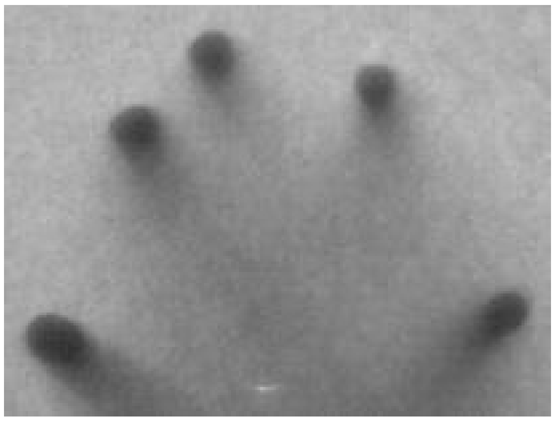

Fig. 2.1 Illustration of Front Diffused Illumination (courtesy of Christian Moore, NUI Group.)

camera-based systems.

Rear Diffused Illumination (RDI) is based on similar principles as FDI, but uses infrared sources placed below the touch surface, as opposed to infrared energy in the environment. Due to the touch surface transparency (it is often made of glass or acrylic), the infrared energy is not reflected back to the camera until a finger (or potentially another object) is placed on the surface. In this case, the camera will see white blobs, the opposite to black blobs in FDI. The table setup is shown in Figure 2.2. As with FDI, this approach requires specific lighting conditions. Its potential for the identification of fiducial markers on tangibles has made it the method of choice for the *reacTable* [9] and *Microsoft Pixelsense*[1] (previously *Microsoft Surface*).

A Laser Light Plane (LLP) can also be used to provide infrared energy to be reflected towards a camera. This method shines infrared energy on a plane slightly above the touch surface. When fingers appear on the surface, they reflect that energy towards a camera, and signal processing techniques allow the tracking of blobs; the latter can be done from above the table, as Crevoisier and Kellum showed [47], or below it, as in Montag et al.'s first-generation LLP table [48]. Interestingly, Crevoisier and Kellum's LLP setup was enhanced with "[...] acoustic onset detection in order to get precise timing information. In addition to fingers, our system can detect oblong objects striking the surface, like sticks and mallets, and it also measures the intensity of taps or impacts [...]" [47].

[1]http://www.microsoft.com/en-us/pixelsense/default.aspx, accessed 2012/08/02

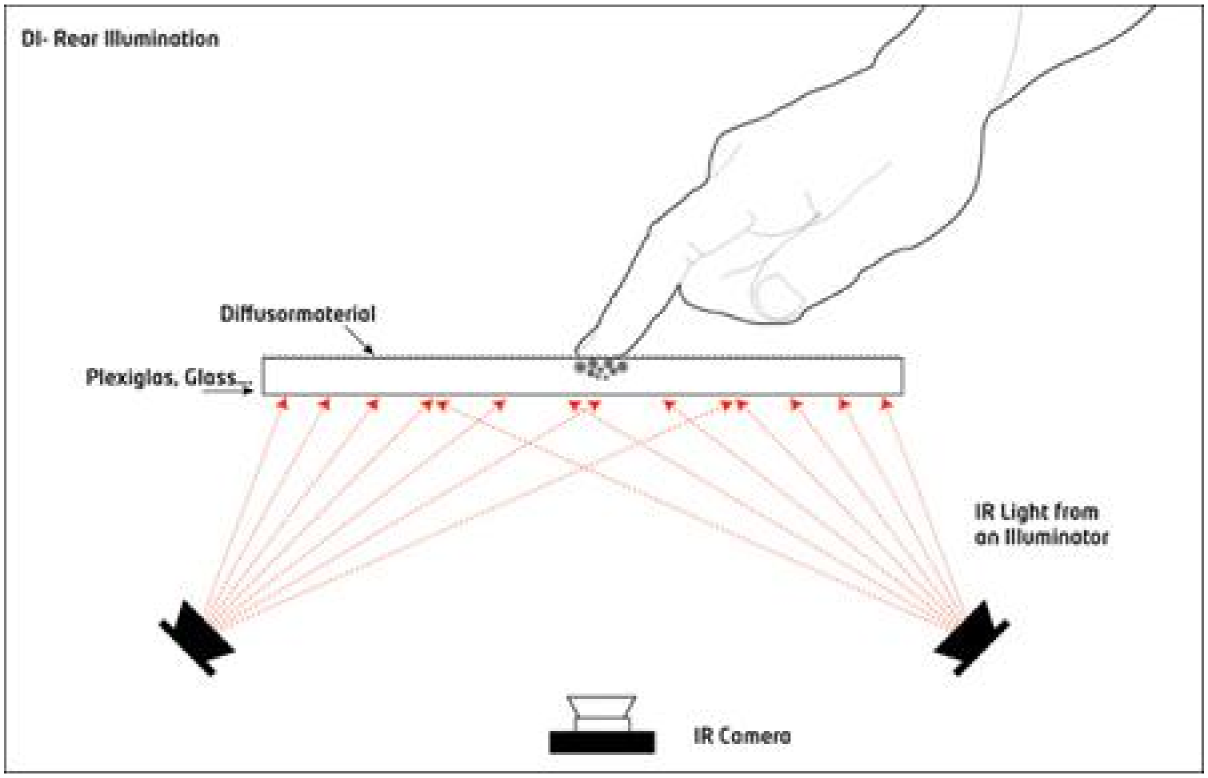

Fig. 2.2 Illustration of Rear Diffused Illumination (courtesy of Christian Moore, NUI Group.)

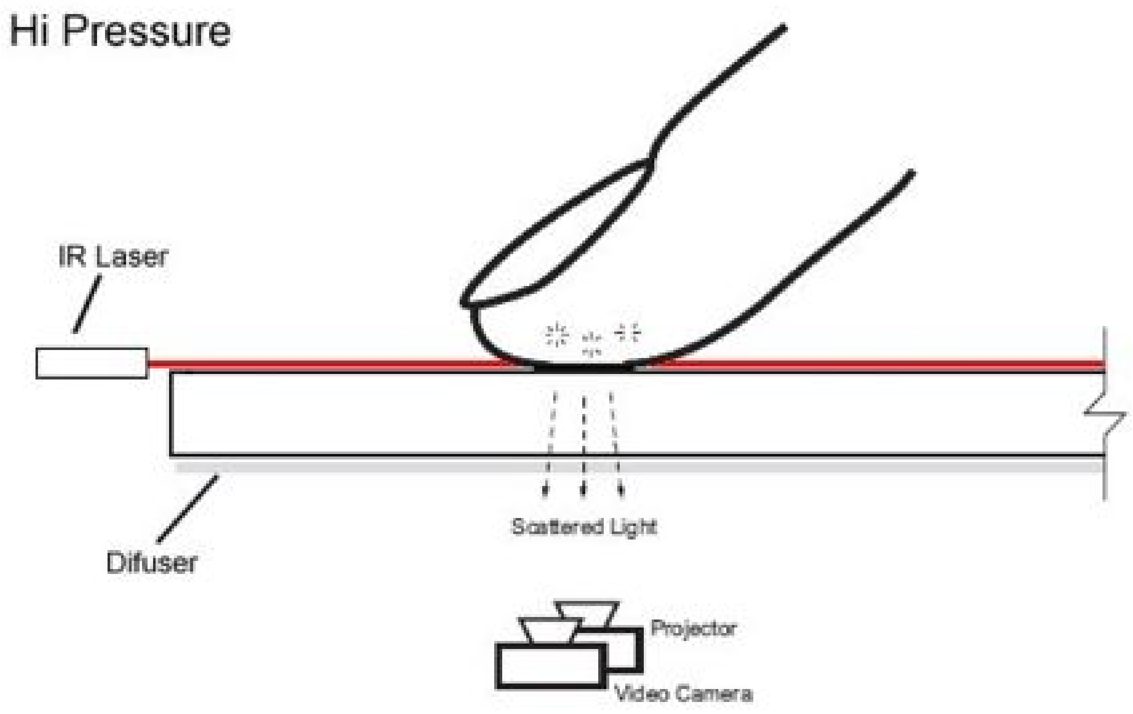

Fig. 2.3 Illustration of a Laser Light Plane (courtesy of Christian Moore, NUI Group.)

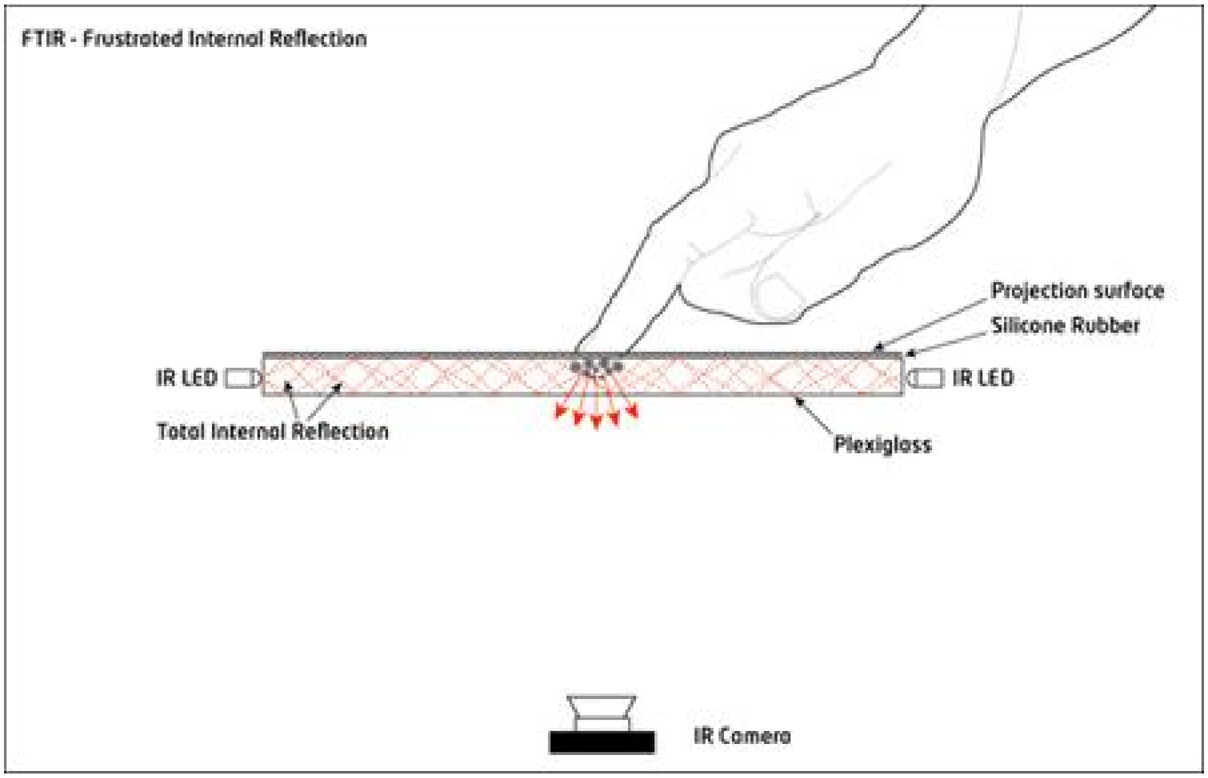

Fig. 2.4 Illustration of Frustrated Total Internal Reflection (courtesy of Christian Moore, NUI Group.)

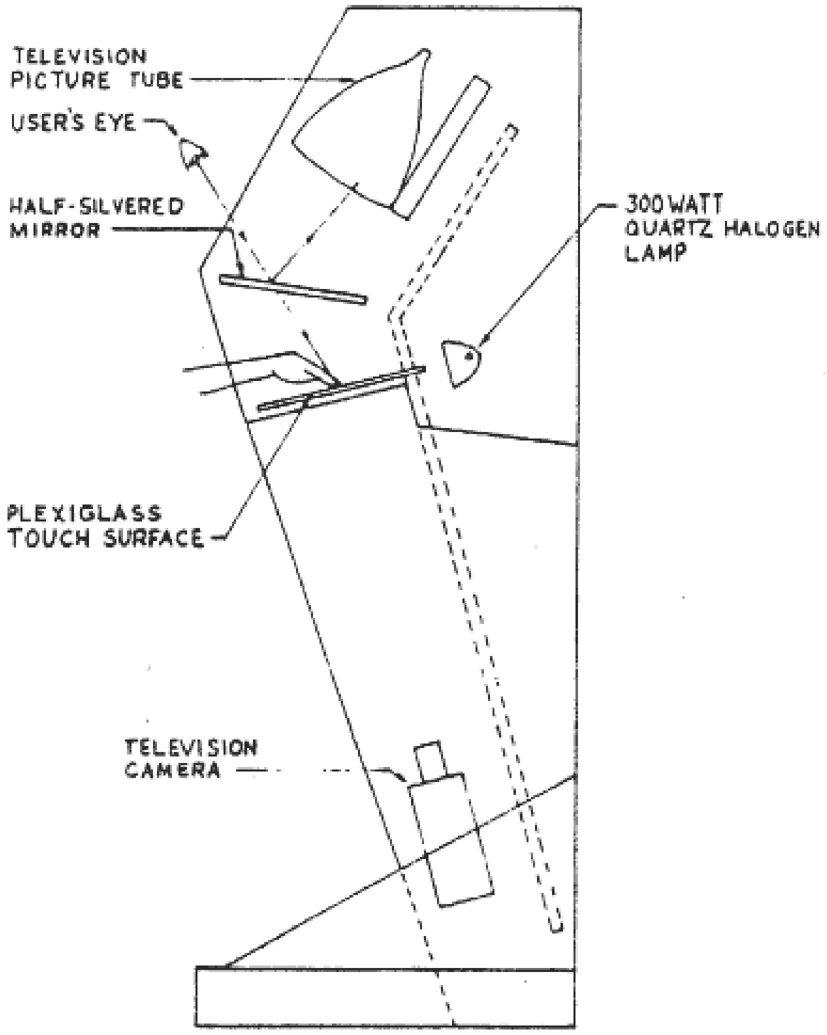

Fig. 2.5 Johnstone's Rolky [1] (courtesy of Eric Johnstone), an early implementation of FTIR

Frustrated Total Internal Reflection (FTIR) is a popular camera-based method for finger tracking. It consists in combining light energy with a sheet of glass or acrylic. Recent implementations [9] use the infrared spectrum, while early systems relied on visible light [1]. The diode energy is kept inside the glass or acrylic by the optical characteristics of the sheet and surrounding environment, until fingers reflect part of that energy towards a camera under the touch surface. FTIR is illustrated in Figures 2.4 and 2.5.

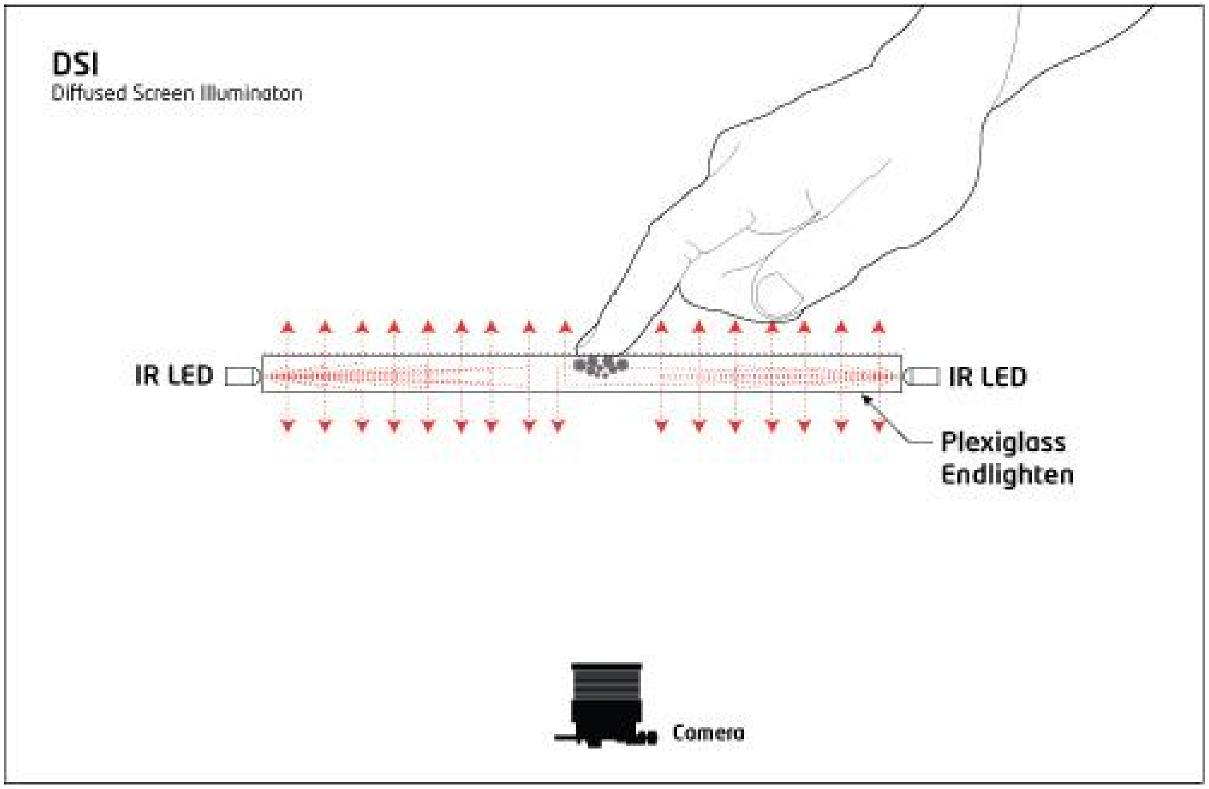

Fig. 2.6 Illustration of Diffused Surface Illumination (courtesy of Christian Moore, NUI Group.)

Diffused Surface Illumination (DSI) is another camera-based solution. It is very similar to FTIR in that it uses an array of infrared LEDs, but it varies in the type of acrylic used. The acrylic chosen has the property of spreading the infrared energy evenly throughout the sheet (see Figure 2.6), thereby giving clearer blobs than FTIR.

Stereoscopy is a technology that uses two cameras to identify the position of objects. When combined with a holographic film that reflects light at a certain angle, as in Microsoft Research's *TouchLight* [49], a partly transparent projection screen can be obtained. This is made possible by a holographic rear projection screen providing a 30°incline. In addition to blob tracking, *TouchLight* can scan documents placed on the touch surface and sense tapping through a piezoelectric sensor.

Multi-touch sensing using 3D cameras such as the *Microsoft Kinect*[2] is a more recent method of tracking finger movement. A more portable implementation of this principle is Harrison et al.'s *OmniTouch* [50], a shoulder-mounted system that used a picoprojector and a short-range depth camera. Years before 3D cameras appeared on the consumer market to facilitate finger tracking, Letessier and Bérard showed multi-touch sensing based on 2D image processing with the possibility of tracking more than 20 fingers [51].

2.3.2 Capacitive Sensing

Capacitive sensing measures changes in the capacitance of electronic circuits to identify the position of fingers. It is used by the *iPad*[3] and *DiamondTouch* [26], among other devices. The characteristics of capacitive sensing and the multiplexing of data were used by Dietz and Leigh to implement *DiamondTouch* a multi-user, multi-touch system.

Apple's devices use an array of capacitive sensors that detect the positions of points, while *DiamondTouch* combines antennas and capacitive coupling to detect which user is touching the surface. The latter system can only detect two fingers, one for each user. It is therefore more of a multi-user single-touch setup than a genuine multi-touch setup.

In the mid-1980s, researchers at the University of Toronto [13] implemented a recursive scanning algorithm for the quick identification of finger positions on a capacitive sensing setup.

Jones et al. designed a low-cost multi-touch surface using capacitive sensing [52], based on design goals related to spatial sampling frequency, cost, and size.

Capacitive sensing has the advantage of supporting several interaction techniques such as hovering gestures and shape-based manipulation [53]. *PocketTouch* [54] was an illustration of capacitive sensing through fabric for multi-touch interaction with a handheld device. It allowed users to type text using gesture recognition. A study of the signal strength for more than twenty fabrics was also included.

[2]http://www.xbox.com/en-CA/Kinect, accessed 2012/03/17
[3]http://www.apple.com/ipad/specs, accessed 2012/03/07

2.3.3 Resistive Technology

Jazzmutant's *Lemur* controller uses *PMatrix* resistive technology[4]. Like capacitive sensing, resistive technology can recognize objects such as styli and pens.

Freed reviewed [55] electronic circuits adapted for multi-touch sensing based on resistive technology. Particular emphasis was placed on shielding components and building energy-efficient circuits, major concerns in instruments meant for travel.

Wessel et al. [24] introduced the design and calibration of a force-sensitive system based on resistive technology. Of particular interest is the description of their method for calibrating the device, using a force hammer hitting the user's fingernails, in order to make the sensors linear. SLABS[18] is a similar system, also presented by Wessel; it was discussed in Section 1.1.3.

2.3.4 Magnetic Sensing

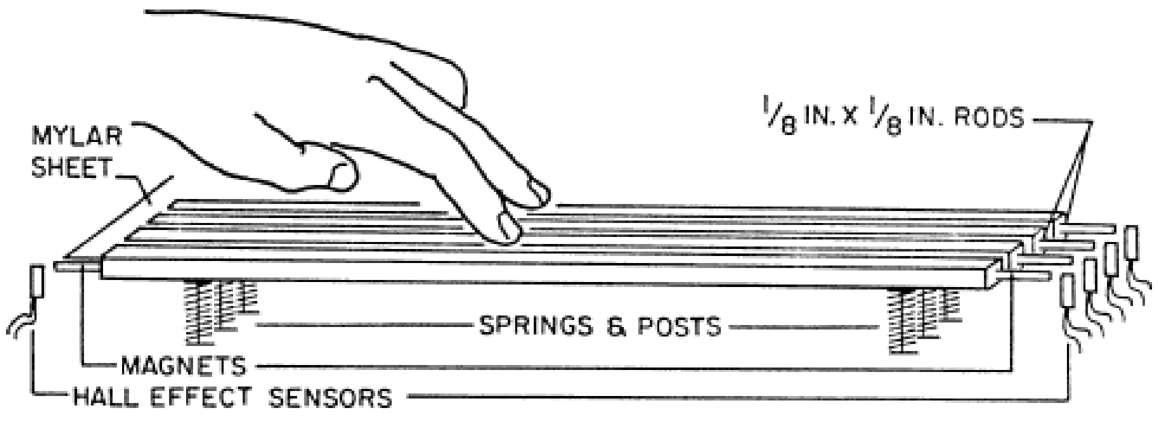

Fig. 2.7 Haken's Continuum [2] (courtesy of Lippold Haken)

It is also possible to implement multi-touch systems with magnetic sensing. The *Continuum* [2], designed at the University of Illinois, uses a Hall effect sensor, as is evident from Figure 2.7. An important feature of the *Continuum* is that it is genuinely pressure-sensitive,

[4]http://www.stantum.com/en/offer/pmatrix, accessed 2012/08/02

whereas camera-based solutions merely infer pressure from blob area. The *Continuum* is used to navigate a tridimensional timbre space.

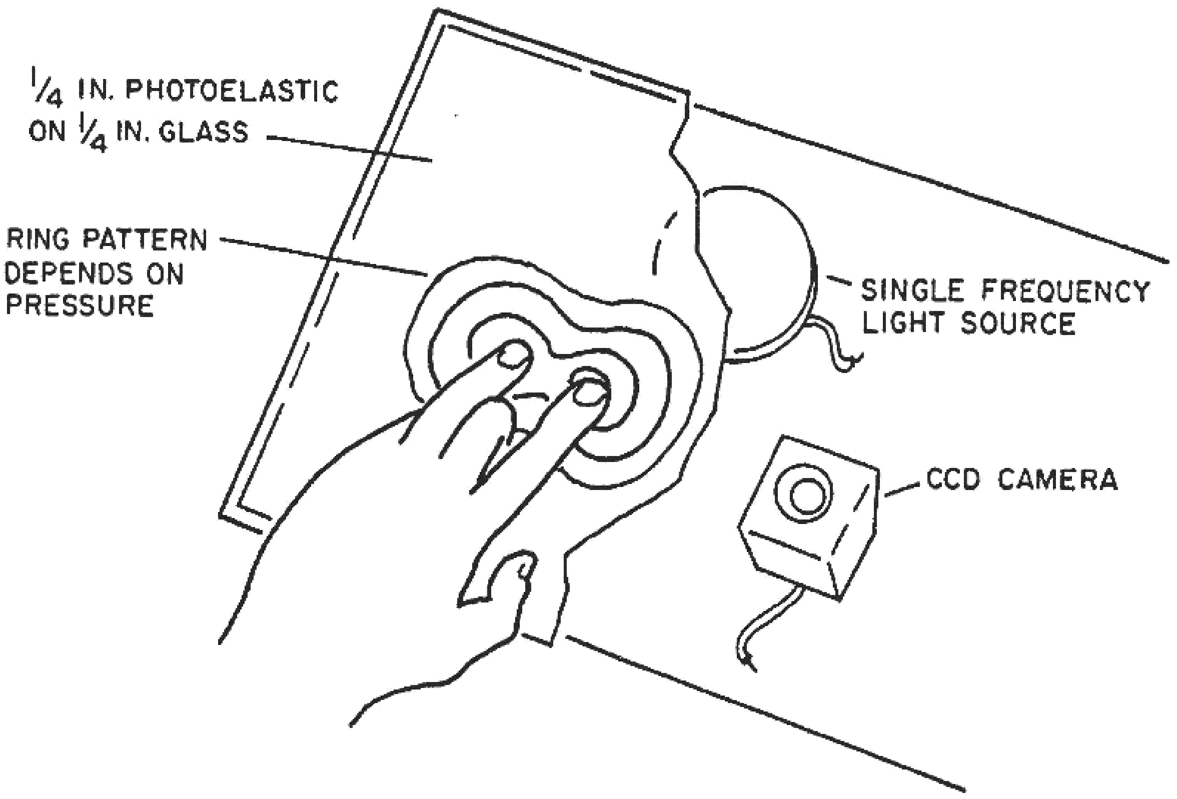
1/4 IN. PHOTOELASTIC
ON 1/4 IN. GLASS
RING PATTERN DEPENDS ON PRESSURE
SINGLE FREQUENCY LIGHT SOURCE
CCD CAMERA

Fig. 2.8 A 1983 prototype of the Continuum [2] (courtesy of Lippold Haken)

Interestingly, one of the initial designs of the Continuum used Rear Diffused Illumination, and another design used resistive technology (conductive rubber) [16]. An illustration of a 1983 prototype of the Continuum is shown in Figure 2.8.

2.3.5 Optical Fibre

Yet another approach to multi-touch technology consists in using the physical properties of optical fibre. The *MTC Express* [56], from the now-defunct company Tactex Controls, used technology licensed by the Canadian Space Agency to perform multi-touch sensing. A pressure-sensitive device, it had optical sensors that detected the deformation of the optical fibre by measuring the amount of light that went through the fibres, at a rate of 200 Hz.

2.3.6 Acoustic Sensing

Different single-touch systems use acoustic sensing to identify the position of fingers on a projection surface: *Sound Rose* [3] (shown in Figure 2.9), Herot and Weinzapfel's touch sensitive digitizer from Instronics Ltd. [11], and Albinsson and Zhai's device from Mass Multimedia, Inc. [57]. They are based on the propagation of waves when fingers hit the touch surface and move on it. This principle was explained and used as early as 1969 [58].

Combined with classifiers, a system based on acoustic sensing can even recognize the part of the finger that was used in multi-touch interaction, as in *TapSense* [59]. This system recognized the nail, tip, pad, and knuckle of the finger. It accomplished the classification using Support Vector Machines (SVMs). As well as human fingers, the authors indicated that tools of different materials could be used on such a surface.

2.3.7 Dispersive Signal Technology

Dispersive Signal Technology (DST) is a recent method developed by the 3M Company [60]. It uses the principle of dispersion, according to which the speed of travelling waves in a solid is a function of the underlying wavelength. Tapping the screen or moving a finger on it causes waves of different frequencies to travel through the surface, while piezoelectric sensors convert the mechanical energy to electrical energy for processing in order to estimate the position of fingers based on the phase differences in the incoming waves.

This technology works even with static objects placed on the surface; it is reportedly very resistant to dirt and grease. DST is slightly different from acoustic wave sensing: "Bending waves differ from surface waves in that they traverse through the thickness of the panel rather than the surface of the material, which provides several important advantages including enhanced palm rejection and superior scratch resistance." [60]

2.4 Visual Feedback Taxonomy

2.4.1 Front Display Devices

Given the long throw distance of commercial projectors up until recently, one can understand the motivation for projecting the visual feedback from above the touch surface. An issue with such a method is occlusion (see Section 1.1.7), as manual interaction on the surface will prevent the projected data from being properly displayed. This configuration

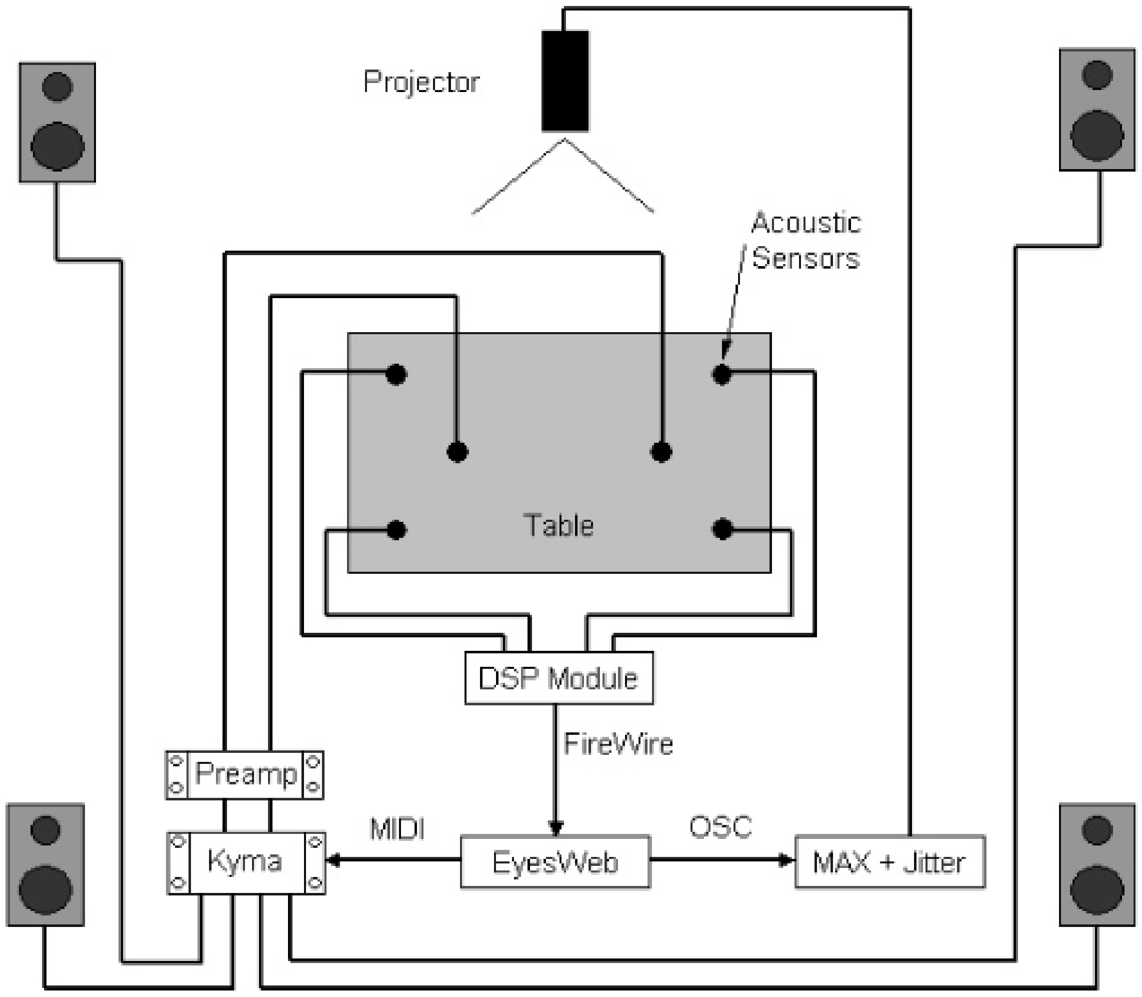

Fig. 2.9 Sound Rose, an acoustic sensing touch device [3] (courtesy of Alain Crevoisier, Cédric Bornand, Arnaud Guichard, Seiichiro Matsumura and Chuichi Arakawa)

has the advantage that any surface could potentially be used for projection, making it a very robust display setup.

2.4.2 Rear Display Devices

A rear display setup eliminates the problem of occlusion (see Section 1.1.7), but requires more maintenance and calibration, in addition to being harder to assemble with DIY means. Rear display setups require a projection surface. An innovative installation was Microsoft's switchable projection screen in *SecondLight*, which "can be made diffuse or clear under electronic control" [61] at a rate that is imperceptible to humans. This allows for the recognition of objects on the table and the projection of specific contents onto those objects, while regular visual feedback is displayed on the table. *SecondLight* supported the tracking of mobile projection surfaces, including the correction of the projected image so that the viewing position of the mobile surface can be changed in three dimensions.

2.4.3 No Display

In some cases, multi-touch technology is implemented without visual feedback. Sometimes visual feedback is not required for the system, although the affordability and availability of projectors mean that a touch-sensitive system can be made front-projected with relative ease. For some technologies [2] [56], rear projection provides a major challenge due to the nature of the sensing technology.

2.5 Applying the Taxonomy

In this section, we place selected hardware within this bidimensional taxonomy. In order to provide additional information about the hardware, we include the items below for each entry, whenever it is available:

A) The maximum number of fingers supported by the device;

B) The dimensions of the touch interface;

C) The sensing of hovering motion;

D) The support of fiducials or tangibles.

Table 2.1: Different multi-touch hardware solutions in our taxonomy

	Rear Display	**Front Display**	**No Display**
Camera-based: FTIR	**Han's TED2005 setup** [19], 2005 A) Limited by camera resolution B) 406mm x 305mm acrylic sheet C) No D) No	**The Rolky** [1], 1985 A) Limited by camera resolution B) "1/4 the area of a 24 inch diagonal CRT" C) No D) No	Unknown to author
Camera-based: RDI	**reacTable** [9], 2005 A) Limited by camera resolution B) 850mm diameter, 1024x768 pixels[5] C) No D) Yes	Uncommon because rear infrared illumination facilitates rear projection	**Haken's first Continuum prototype** [16], 1992 A) Continuous control B) Unknown C) No D) Possible in theory
Camera-based: LLP	**Montag et al.'s LLP prototype** [48], 2011 A) Unkown B) Unknown C) No	Unknown to author	**Crevoisier and Kellum's table** [47], 2008 A) Unknown B) Estimated to about 914mm x 610mm C) No

Continued on next page

[5]http://www.reactable.com/products/live/tech-specs, accessed 2011/12/08

Table 2.1 – *Continued from previous page*

	Rear Display	**Front Display**	**No Display**
	D) No		D) No
Camera-based: Stereoscopy	**Microsoft Research's TouchLight** [49], 2004 A) Unspecified (all contact on the plane can be sensed) B) The interface is of the order of several feet wide and tall. C) Yes D) Yes	Unknown to author	Unknown to author
Camera-based: Depth Sensing	Unknown to author	**OmniTouch** [50], 2011 A) At least two fingers B) Projected onto limbs and walls using a picoprojector C) Yes (clicking is detected) D) No	**Letessier and Bérard's system** [51], 2004 A) More than 20 B) 1000mm by 750mm C) No D) No
Capacitive Sensing	**iPad 3**[6], 2012 A) Maximum 11 fingers B) 246mm diagonal, 2048x1536 pixels	**DiamondTouch** [26], 2011 A) More than one finger per user B) About 800mm x 480mm	**PocketTouch** [54], 2011 A) Screenshots show three fingers B) 46mm x 97mm

Continued on next page

[6]http://www.apple.com/ipad/specs, accessed 2012/03/07

Table 2.1 – *Continued from previous page*

	Rear Display	**Front Display**	**No Display**
	C) No D) No	C) No D) Possible with customized objects	C) No D) No
Resistive Sensing	**Lemur**[7], 2004 A) Reportedly unlimited, in practice the surface gets crowded at 10 fingers B) 304mm diagonal, 800x600 pixels C) No D) No	Unknown to author	**Haken's second Continuum prototype**, 1992 [16] A) Continuous control B) Unknown C) No D) Possible if the objects cause a displacement of the rubber
Magnetic Sensing	Unknown to author	Unknown to author	**Haken's Continuum** [2], 1997 A) Continuous control, has the form factor of a keyboard (10 fingers) B) Full size: 1370mm x 190mm Half size: 720mm x 190mm[8] C) No D) No

Continued on next page

[7]http://www.jazzmutant.com/lemur_features.php, accessed 2012/08/01

[8]http://www.hakenaudio.com/Continuum/hakenaudiooverva.html, accessed 2011/12/09

Table 2.1 – *Continued from previous page*

	Rear Display	**Front Display**	**No Display**
Acoustic Sensing	**TapSense** [59], 2011 A) Shown using four fingers (adding fingers increases the risk of “timing collisions” [3]) B) 1100mm x 750mm C) No D) Yes (RDI prototype)	**Sound Rose** [3], 2006 A) One finger B) 800mm x 600mm C) No D) asd	Unknown to author
Optical fibre	Unknown to author	Unknown to author	**MTC Express** [56] A) Practical limit of about four fingers. B) 145mm x 95mm C) No D) Possible with materials of the proper weight and reflectivity
Dispersive Signal Technology	**3M Technology** [60], 2008 A) Unknown B) Scalable to sizes above 812mm diameter C) No D) No (tolerant to static objects)	Unknown to author	Unknown to author

The results of this classification are presented in Table 2.1, in which the technology axis corresponds to rows and the visual feedback axis to columns.

2.6 Building an FTIR and RDI Multi-Touch Table

This section details the construction of a multi-touch table that spans different regions of our taxonomy. At the start of the author's research as part of this thesis (September 2009), no affordable (around 1,000 CAD) touch devices with rear display and the form factor of a coffee table or desk were available on the commercial market. Tablets had not appeared on the market as consumer products either. As such, the author decided to build a touch table from scratch, with an input connection for display and an output connection for touch signals. The hardware could then be connected to a computer for software development.

The design and choice of materials was done pro bono by Olivier Plante, graduate student in Industrial Design at the Université de Montréal in Montreal, Canada. All electronics were designed and assembled by the author.

2.6.1 Electronic Components

The electronic components used in the system are:

- More than twenty KIE-7305 Infrared LEDs: 840 nm wavelength, 1.5 V at 50 mA;
- Optoma ES522 Projector: 2,800 Lumens, 1.2 m minimum throw distance, 4:3 aspect ratio;
- PlayStation Eye webcam: 640x480 pixels or 320x240 pixels, frame rate up to 100 FPS;
- A computer fan to provide air movement inside the table;
- Various resistors, capacitors, switches, and toggles, to provide energy for the FTIR and RDI setup.

The projector was chosen in the Fall of 2009 based on its low price, and despite its relatively long throw distance for the purpose of building a touch table.

2.6.2 Other materials

Other materials were used for the structure of the table:

- Russian cherrywood plywood for the bottom part of the table: chosen for its aesthetic quality and low price;
- Maple for the top part of the table: chosen for its strength and texture;
- Wood stain and varnish for a nice finish.

2.6.3 Prototyping

A first test needed to be done in order to validate the use of a mirror for the projection, and to estimate a reasonable throw distance using the mirror. This setup is show in Figure 2.10.

An initial prototype was built out of foam, in order to investigate the ergonomy of the project. This led us to rethink the form of the table.

2.6.4 Final Result

The final setup is shown in Figures 2.11.

Figure 2.12 shows the switch on the table that allows the system configuration to be changed between FTIR and RDI, while Figure 2.13 shows the VGA and USB connectors of the system. The VGA connector is an input to the table (to project visuals onto the table top), while the USB connector is an output (it sends the webcam image to a computer for processing).

2.7 Summary

This chapter introduced a biaxial taxonomy of multi-touch hardware. This taxonomy is based on the various methods of implementing multi-touch technology and the visualization playback of the hardware. It shows the variety of methods that can be used to provide multi-touch interaction. We placed existing hardware within this taxonomy, adding extra information (number of fingers sensed, dimensions, sensing of hovering gestures, and support for tangibles or fiducials) to each entry when possible.

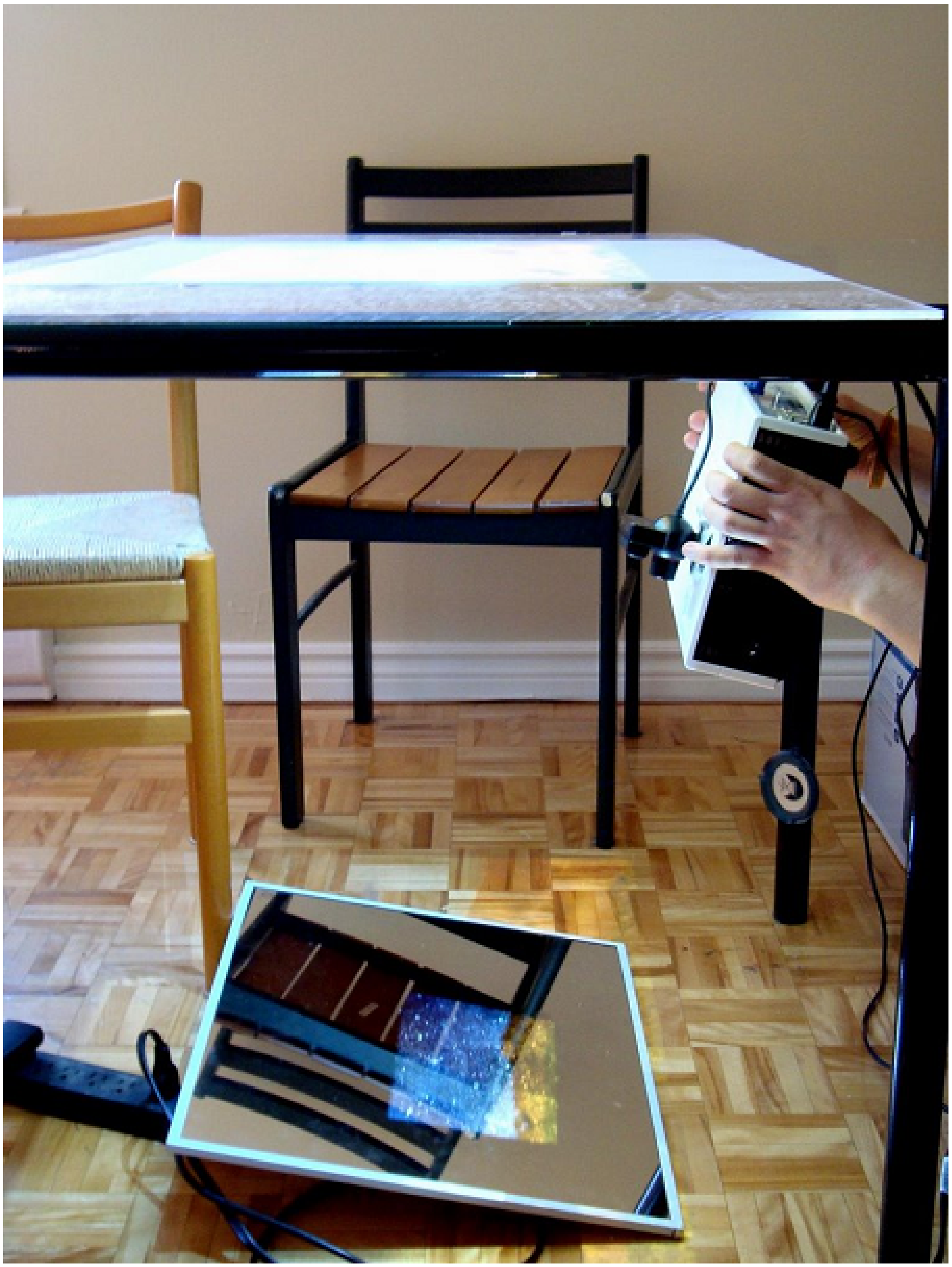

Fig. 2.10 Estimating the throw distance and use of a mirror in our setup

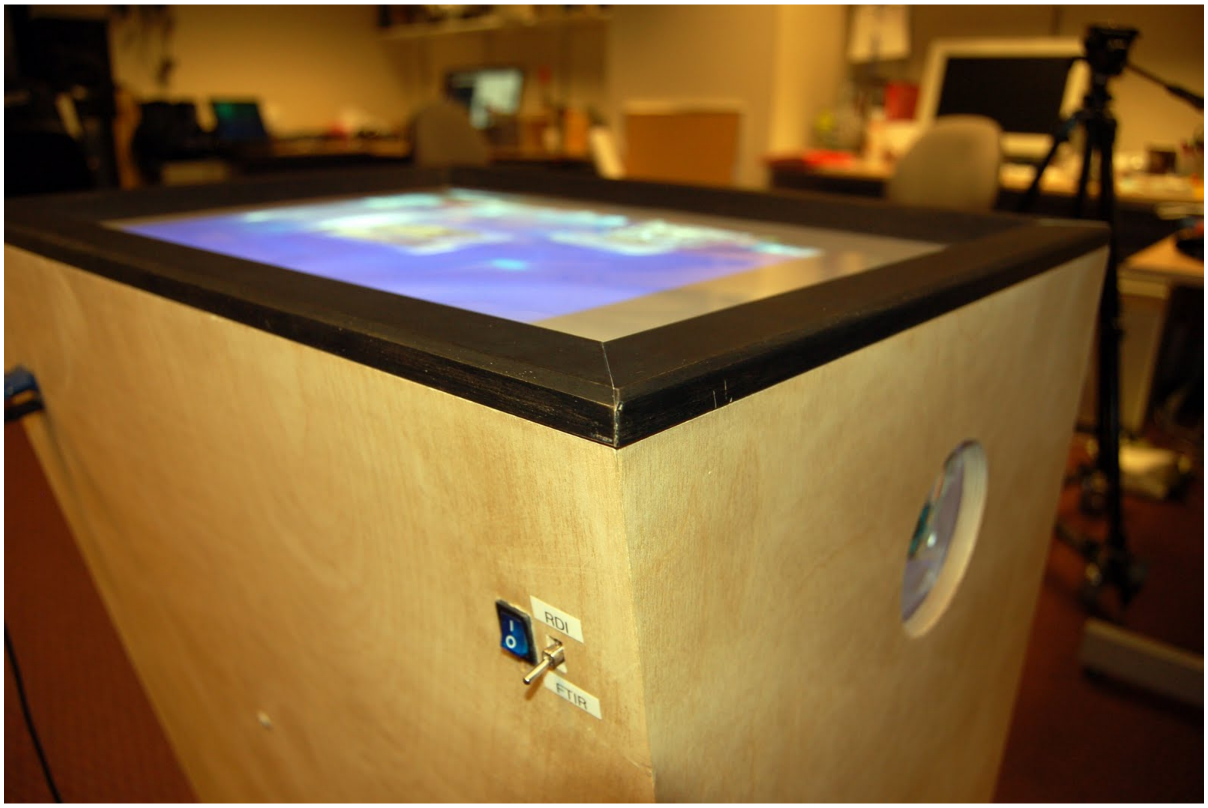

Fig. 2.11 A view at an angle of our final piece of hardware

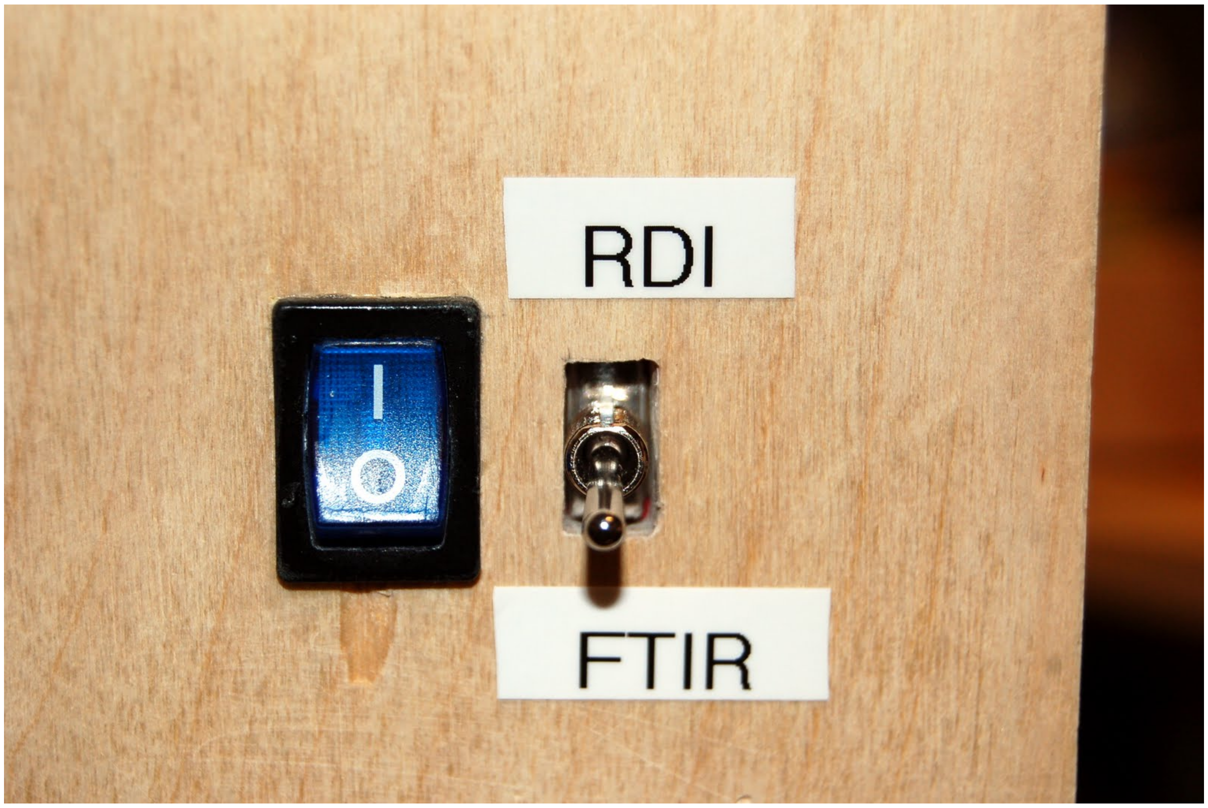

Fig. 2.12 The switch that allows the user to change between FTIR and RDI

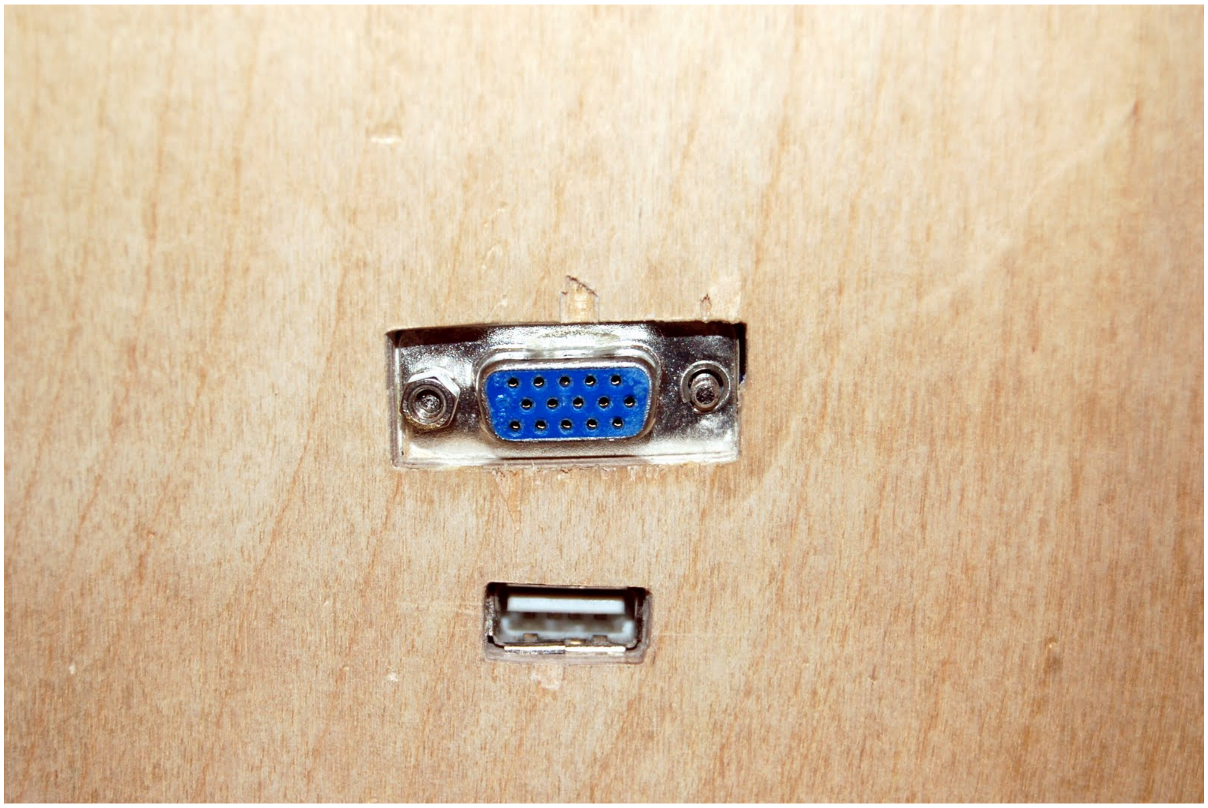

Fig. 2.13 The VGA connector for the projector, and the USB connector of the webcam

We also described our approach when designing our own DIY multi-touch hardware system. Our hardware platform supports both FTIR and RDI interaction, thereby allowing us to support the tracking of fiducials.

In the next chapter, we will consider software issues in multi-touch installations and review music-based touch-enabled applications.

Chapter 3

Multi-Touch Systems: Software Choices and a Review of Musical Applications

Having seen different approaches to implementing multi-touch hardware, we now turn to aspects of software that must be considered when designing multi-touch applications, some of which also apply to other controllers. We then present new GUI controls that could be used in multi-touch software, before exploring different categories of existing touch-enabled musical applications, based on the main task they facilitate: music exploration, sequencing (and production), synthesis, and editing.

3.1 Software Choices in Multi-Touch Systems

3.1.1 Modular Software Design for Computer Vision

Given the widespread use of webcam-based solutions to implement low-cost DIY multi-touch technology, it was only natural that video processing applications would emerge as black box solutions to provide multi-touch event information to accompanying software.

Software such as *Community Core Vision* (aka *tbeta*)[1] or *reacTIVision*[2] are solutions that integrate filtering, alignment, and calibration to identify both fingers and fiducials

[1]http://ccv.nuigroup.com, accessed 2010/11/17
[2]http://reactivision.sourceforge.net, accessed 2010/11/17

in multi-touch camera-based systems. The main benefit of such applications is that they remove the need to implement low-level features, at the cost of a more complicated installation pipeline. Since their source code is available, it is possible to integrate these solutions into your own software, but this might force the programmer to work in a very specific environment (as is the case with *Community Core Vision*, which uses a customized version of *OpenFrameworks*). These applications output the blob data to a network device, using the TUIO protocol [62] for instance, based on OSC communication.

A modular design allows the use of different software frameworks, possibly in different programming languages, on different machines running different operating systems. In this manner, computer vision operations can be separated from the GUI processing and audio playback, if needed due to limited hardware resources.

It is worth noting that modular software pipelines are not exclusive to computer vision software. Musical performance is often accomplished using different software environments, sometimes at the same time during performance. An example of a tool that facilitates this combination of software environments is *Max for Live*[3], which facilitates communication between *Max/MSP* and *Ableton Live*, two popular musician tools.

3.1.2 Libraries

Some software libraries provide an abstraction layer of touch events so that the blob information does not need to be interpreted, and instead appears more conveniently in the programming environment.

This is the case of *PyMT*[4], a library for Python. *LightTracker* [63] was a multi-touch programming framework for systems based on computer vision. It was compared to existing frameworks to highlight its features, namely its customizable filter pipeline, support for multi-threaded architectures, and own calibration module. The authors also provided a review of existing low-level (*BBTouch*, *Touch*, *CCV*, *movid*, *reacTIVision*) and high-level libraries (*DiamondSpin SDK*, *Microsoft Surface SDK*, *Squidy*, *PyMT*).

Kammer et al. have suggested a taxonomy for multi-touch software frameworks [44]. It has three axes: architecture (e.g., platform, support for gesture and TUIO), scope (e.g., support for tangibles, the parameters of each blobs), and features (e.g., GUI controls, ease of integration).

[3]http://www.ableton.com/maxforlive, accessed 2012/08/10

[4]http://pymt.eu, accessed 2010/11/17

Kellum and Crevoisier's *SurfaceEditor* [64] introduces the concept of a chain of event filters. Their types are parameter filters (looking at the characteristics of blobs), ordering filters (to prioritize certain events), count filters (to follow the number of fingers on a component, for instance), and temporal filters (to track the timing of events).

3.1.3 Selection Strategies

As with other input devices, different selection strategies are available in multi-touch systems. Potter et. al [65] present take-off, first-contact, and land-on, while Sears and Schneiderman [66] also mention the existence of other strategies requiring a second touch.

3.1.4 Gestures

Although blob position, size, and acceleration are simple parameters to use in software development, the recognition of gestures poses another challenge. Hoste [67] and Rubine [68] present research about defining and recognizing gestures in software. While Rubine's focus was on mouse input, Hoste expanded the concept to multi-touch interfaces. In discussing the rejection of unclear gestures, Rubine argues that "in applications without undo, rejection is preferable to misclassification and should be enabled" [68]. Minksy was already working on single-touch gestures in the 1980s to "introduce programming ideas to very young children" [25].

Moscovich and Hughes [69] presented findings that indicate the following principles in dual-finger manipulation in single-handed and double-handed gestures:

- "Unimanual multi-touch manipulation is compatible with a visual rotation task";
- "Two handed multi-touch manipulation is only compatible with an object manipulation task when there is a clear correspondence between the fingers and the manipulated control points";
- "Two hands perform better than one at tasks that require separate control of two points".

These concepts have been embraced by software designers of multi-touch devices, the most recognizable features of smartphones being the pinch-to-zoom and dual-finger rotation tasks. Furthermore, the aforementioned guidelines can help software designers in the choice of controls and tasks assigned to those controls.

Kammer et al. distinguish online gestures from offline gestures: "offline gestures are usually processed after the interaction is finished" [44].

Discussing video games, Wachs et al. identify gesture spotting ("distinguishing useful gestures from unintentional movement" [70]) and a fast response as essential requirements of such systems. Non-contact installations such as *SoundCatcher* [71] respect Wachs et al.'s "come as you are" [70] requirement. Wachs et al. also mention a trade-off between the number of gestures sensed by a system and the performance of its recognition algorithms.

3.1.5 Continuous Controls and Absolute Positioning

As opposed to computer keyboards, multi-touch devices offer the possibility of continuous parameter control. An important difference between multi-touch and WIMP interaction is also absolute positioning.

Continuous controls allow software designers to expose access to the finer details of parameters that can then easily be adjusted. This type of interaction is clumsily performed using a computer mouse, and computer keyboard keys only allow the user to navigate a given range by increments.

Because they often combine the control and display surfaces, multi-touch devices might need the user to focus his or her attention away from a potential audience. This can take away some of the immersiveness of a musical performances, although a clever choice of gestures[5] might make interaction possible without looking at the table.

3.1.6 Innovative Controls

After considering the characteristics and issues of multi-touch technology, we focus on innovative controls that are found in the literature. The common controls of basic WIMP interaction, namely sliders, buttons, menus, knobs, XY graphs, and text fields, can be used in multi-touch interaction, but more innovative controls also exist.

Following Jacob et al.'s hypothesis, we are of the opinion that "performance improves when the perceptual structure of the task matches the control structure of the device" [72]. We hope to exploit this relation of task and device by evoking familiar control structures

[5]http://www.gergwerk.com/work/design/design01.html, accessed 2012/03/11

through a careful choice of user interface metaphor. The integrality or separability of controls [72] can be exploited in multi-touch software, so as to sometimes combine or separate different parameter changes.

General Controls

Pie menus and marking menus [73] are now familiar to most computer users. *Stacked Half-Pie Menus* [4], an alternative to multi-level WIMP menus for touch tables, are meant for navigating nested hierarchic data structures and are designed to minimize the time required for choosing elements and the occlusion of menu items. They can support an unlimited number of menu entries. An illustration is presented in Figure 3.1.

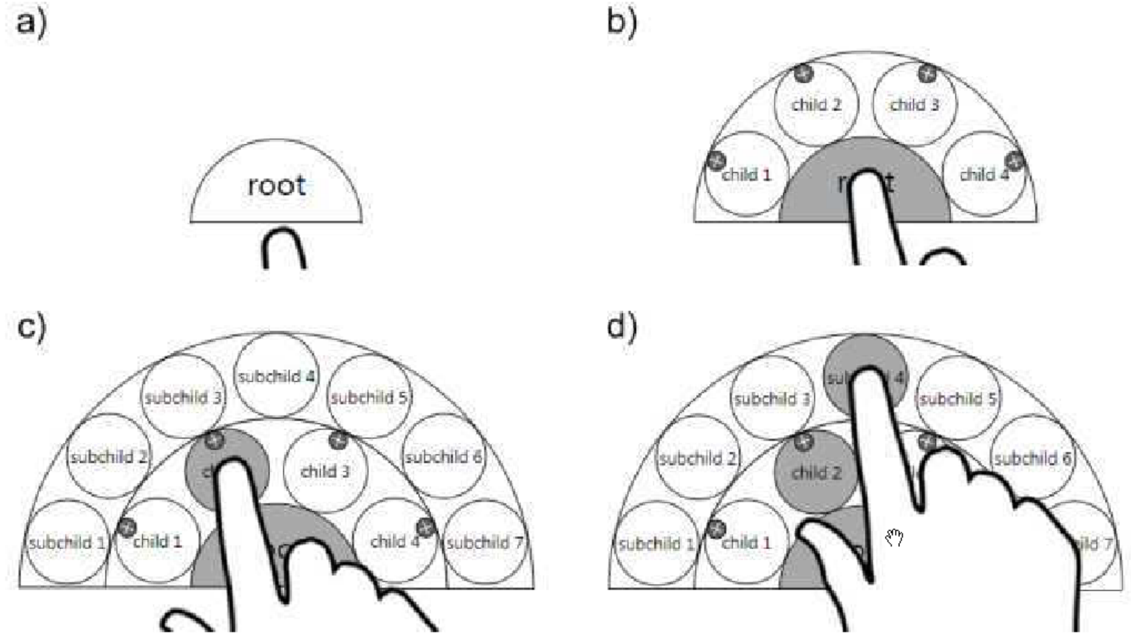

Fig. 3.1 Illustration of stacked half-pie menus [4] (courtesy of Tobias Hesselmann, Stefan Flöring, and Marwin Schmitt)

Geiger introduced the concept of border crossing, based on the action of plucking a string: "an event gets triggered when the pointer on the touch screen crosses a border" [74]. This control could be used on portable devices to provide page navigation in e-books: dragging the finger beyond a certain position would cause the next page to be loaded.

Cuebert's [75] background research revealed a need for the automated movement of controls. This type of programmed path can be more easily implemented in software than in hardware. It could be an interesting high-level addition to the controls of a multi-touch interface. *Native Instruments Traktor*[6] implements such a programmed path in automatically moving the crossfader to its extremes in one mouse click.

The *Pump* gesture [27] was presented as a solution to the issue of occlusion that is widespread in multi-touch interaction. It allows a cursor to be offset either below or above the finger of the user, in a traditional WIMP environment. This control is useful in touch-based interaction that still use a cursor, such as *TeamViewer*[7], a remote desktop application for mobile devices. It is also implemented in Android 4.0[8], Google's mobile operating system, when choosing between different suggested spellings or similar words.

Pixel-Level Selection

Techniques for pixel-level selection include the *Cross-Lever*, *Virtual Keys*, *Cross-Keys*, *2D Lever*, and *Precision-Handle* [57], or *Dual Finger Selections* (Offset, Midpoint, Stretch, X-Menu, and Slider) [5]. They are not necessarily required in all types of applications. In many situations, user interfaces can be designed with large controls, and gestures for zooming in can make it unnecessary to perform pixel-level selection. In situations where traditional operating systems have to be used or large data sets have to be explored, these selection methods do, however, provide a way of overcoming the difficulties of selecting small targets with a human finger.

Clicking

Simpress [5] is a method for emulating real pressure-sensitive buttons based on the area covered by the fingers. It is shown in Figure 3.2.

The *Fluid DTMouse* [76] is a multi-touch implementation of a regular WIMP mouse that avoids occlusion and handles issues such as distinguishing dragging from mouse-over motion.

[6]http://www.native-instruments.com/#/en/products/?category=1316, accessed 2011/07/20
[7]http://www.teamviewer.com/en/download/mobile.aspx, accessed 2012/03/18
[8]http://www.android.com/about/ice-cream-sandwich, accessed 2012/08/02

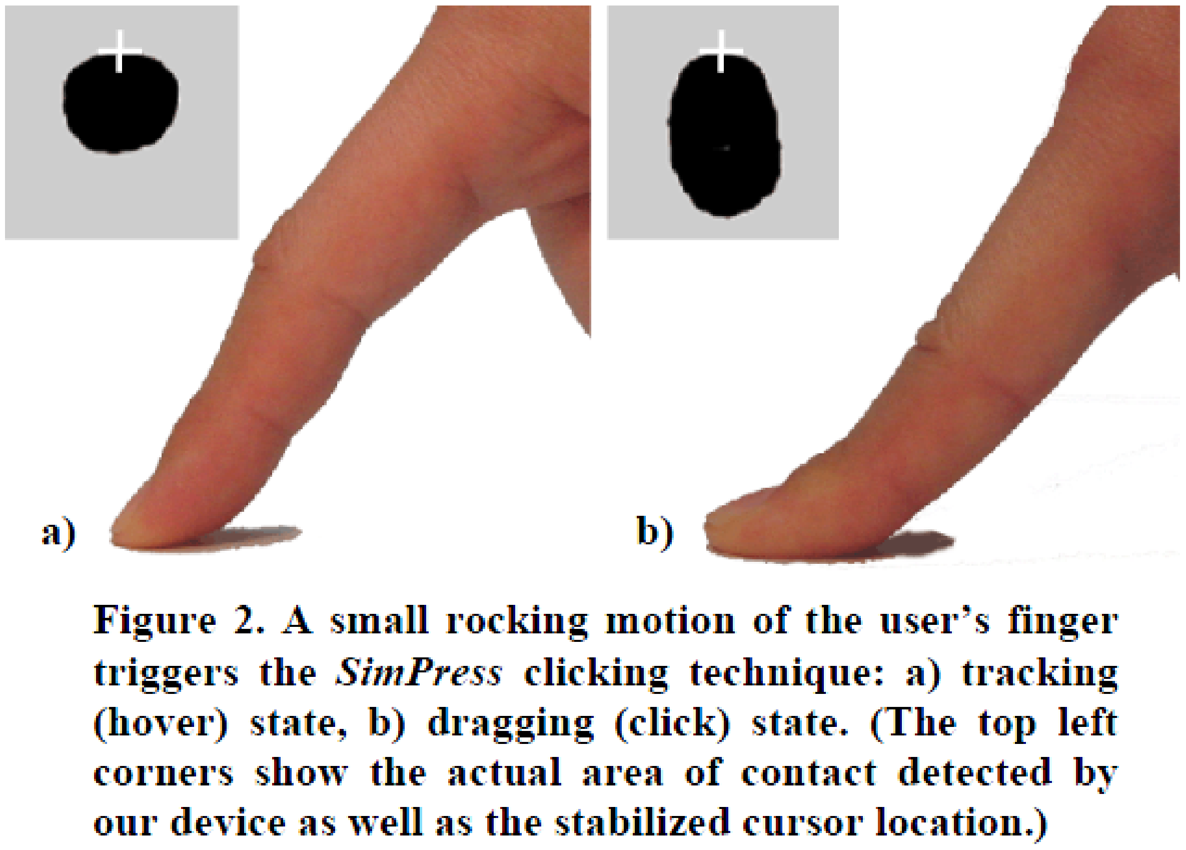

Figure 2. A small rocking motion of the user's finger triggers the *SimPress* clicking technique: a) tracking (hover) state, b) dragging (click) state. (The top left corners show the actual area of contact detected by our device as well as the stabilized cursor location.)

Fig. 3.2 Illustration of the *SimPress* technique, with accompanying explanation [5] (courtesy of Hrvoje Benko, Andrew D. Wilson, and Patrick Baudisch)

Zooming In and Out

Zliding [77] is a method for using pressure-sensitive controllers so as to zoom in and out of areas. It was presented in the context of audio and video editing. Ramos and Balakrishnan also introduced the *Zliding Wheel*, a knob control that allows users to “control the granularity of the wheel’s increments” [77].

DTLens [78] is a multi-user environment designed for the visualization of high-resolution geospatial data. As such, it provides the possibility of locking a region and invoking back a previously-zoomed area.

Rub-pointing [79] is a method for zooming in and out of a touch screen and selecting small targets; it uses a diagonal rubbing motion and the take-off selection strategy.

3.2 A Review of Musical Applications on (Multi-)Touch Systems

3.2.1 Music Exploration

In designing *Audioscapes*, “a framework for prototyping and exploring how touch-based and gestural controllers can be used with state-of-the-art content and context-aware visualizations.” [80], the usefulness of touch systems was presented in the field of Music Information Retrieval. Self-Organizing Maps (SOMs) were used to visualize large music libraries on different interfaces. The authors of *Audioscapes* do not discuss the use of multi-touch interaction, but we can easily imagine the use of the dual-finger pinch-to-zoom gesture to navigate large collections of music.

Torrens et al. [81] discussed different visualization techniques for large music libraries. A disc-like visualization system supporting playlist management was introduced. The other techniques are rectangle visualization (a single table with two metadata fields as axes) and tree-map visualization (similar to SOMs). One of the findings of the researchers is that tree-map visualizations are not well suited for displaying information about tracks and playlists. The use of a cursor and scroll bar indicate the focus of the authors on data visualization and not on user interaction, as multi-touch possibilities are not discussed by Torrens et al.

Researchers at the University of Victoria [82] developed an application on the *DiamondTouch* [26] hardware for the collaborative browsing of a music library organized by a SOM. SOMs are a data visualization technique that appears often in Music Information Retrieval literature when large metadata datasets need to be explored. *DiamondTouch* allows for multi-touch, multi-user interaction.

3.2.2 Music Control, Sequencing, Score Editing

Frießet al.'s *Collaborative Multi-touch Sequencer*[9] was a multi-touch sequencer using concentric rings. The *Stereotronic Multi-Synth Orchestra*[10] also used a user interface with concentric rings to implement a sequencer, which supported multi-touch interaction.

Crevoisier and Kellum designed a multi-touch LLP system [47] to control different virtual objects such as buttons, sliders, and XY graphs. These controls are common in sequencers, but no clear description of their audio environment is described.

Roots [83] was a multi-touch sequencer for the *Bricktable* [35] system that displayed expanding vines that were then "actively triggering sounds or loops associated with invisible zones on the screen" [35]. A combination of tangible objects and finger interaction allowed the user to control the playback of sound buffers in three modes: "either completely generative, semi-generatively, and/or fully controlled" [35]. The choice of an organic and generative element in the user interface was combined with the presence of several knobs that allowed the music playback parameters to be modified in real-time; this explicit access to parameters would have been much more difficult in the aesthetically rich generative environment.

Laney et al. [84] discussed a multi-touch, collaborative multi-user music control system in an analysis of the issues related to collaborative music production. The prototype shown allowed four users to control the playback of musical phrases on a tabletop interface, each having his or her instrument (bass, drum, keyboard, percussion). Beyond choosing an individual loop, the users do not seem to be able to influence other parameters of the sequencing or audio effects, as the focus of the authors was on collaborative musical performance within a creative process.

Scrapple [85] was a spectrographic sequencer based on an additive synthesizer. In it, the length of a long sewing table represents the time axis, while its width represents the frequency axis. Pieces of fabric of different sizes were placed on the sewing table to trigger notes of different lengths, as in a regular sequencer. Instead of tracking fingers, *Scrapple*

[9]http://www.dfki.de/its2010/papers/video/de131.wmv, accessed 2010/12/07

[10]http://vimeo.com/6859653, accessed 2010/12/07

generated a soundtrack based on the fabric markers placed on the table. It was therefore multi-touch in the sense that different fingers could affect the system concurrently.

Cuebert [75] was a multi-touch mixing board for sound engineers in theatres. At the core of its design was the replacement of several knobs by a touch screen, while leaving the general architecture of the console the same, due to the need for an instantaneous visual evaluation of the board's state by the user. We expect that, if the software interface of *Cuebert* is to maintain the look of a mixing board, the lack of tactile feedback may be a hindrance to adoption by technicians, because of the difference in fine control when displacing heavy mechanical knobs through noticeable friction (as is the case with most mixing desk sliders) and virtual objects with minimal friction. We agree with the authors' view that their interface promises added value in terms of automated motion and the "flexible communication of dynamic and context-sensitive content" [75].

WallBalls [86] enabled the playback of elements from a sample bank in an environment made of balls and walls. Although this environment does not implement friction and gravity, it allows loops and random trajectories. Not exactly a touch system, *WallBalls* uses an electromagnetic motion tracker with four styli in its interaction. The system is therefore multi-user and single-touch.

3.2.3 Music Synthesis

O'Sullivan and Boland [87] presented a functional analysis of multi-touch systems from the perspective of musical applications. Their research provides a good overview of multi-touch technology, but we are more interested in their discussion of sound symbolism for visual feedback during sound synthesis. In their environment, a round shape produces a "sine-type tone" [87], while a shape with pointy edges produces a sound with more harmonics.

The *reacTable* is an "electro-acoustic music instrument with a tabletop tangible user interface" [9], meant to be used by different collaborators. Running at 60 FPS and 640x480 pixels, the system provides rich visual feedback to the user. The list of possible objects on the table is: generators, audio filters, controllers, mixers, and clock synchronizers. An important feature of the *reacTable* is that this multi-touch device combines the tasks of editing a piece and playing it.

Sasaki et al.'s pressure-sensitive single-touch tablet [12] was used to control an FM synthesis system. *Luthier* [1], an instrument building application for the *Rolky*, allowed the synthesis of musical contents on a multi-touch device with visual feedback a few years later.

Davidson and Han presented a multi-touch sound synthesis engine with an emphasis on the “richly dynamic context” [88] of the graphical display. The authors argue that tangible systems “face challenges as the complexity of the environment increases” [88], and instead choose to offer the same behaviour using virtual puck-like GUI widgets. In the paper, the authors indicate an interest for the use of gestures using one or two hands for the control of physical models of musical instruments.

Cicconet et al. [89] used an augmented multi-touch system (a tabletop instrument with mounted guitar strings) to allow music improvisation on the blues scale. In addition to playing notes on an interface resembling a guitar neck, the polyphonic system provides access to note bending and a similar interaction mode, vibrato. Moog’s [14] keyboard-like controller with five degrees of freedom (see Section 1.1.3) is another instrument meant for electronic music synthesis. Never commercialized, it was later presented at NIME 2005 [15] by Eaton and Moog. In the 1950s, Le Caine developed the *Electronic Sackbut* [90], an electronics-based synthesizer that supported mutiple-finger input.

SurfaceMusic [91] implements three types of instruments: percussive, string, and wind. It uses OSC and ChucK, and also recognizes gestures. We praise Fyfe et al’s work for investigating non-traditional representations of instruments (all three appear as discs, with differences between instruments), and for the use of parameters derived from simple interaction: in the string instrument, “the angle of the strum gesture maps to pluck position and the speed of the strum maps to attack velocity in the physical model” [91]. Kuhara and Kobayashi [92] present an oscillator-based synthesizer that uses the kinetic information of a collision-enabled environment of virtual particles. Several attributes of the particles are used in the sound: linear velocity, angular velocity, shape, and collisions. The high-level control of a particle system results in an environment that does not require constant active participation from its user, a feature that is useful for ludic systems, as illustrated by the reported popularity of Kuhara and Kobayashi’s mobile device application with children.

3.2.4 Music Editing

Roma and Xambó’s waveform editor [93] was a tangible system used both to modify waveforms and for live music performance. The authors described their system as implementing the metaphor of a toolkit, wherein physical pucks are tools. Instead of presenting the palette of tools in an elongated array (as in standard image editing software), the palette was represented by the tangible objects, much in the way the *reacTable* presents the user with a tangible toolkit surrounding the physical device. This waveform editor also supported gestures: dual-finger zooming and single-finger scrolling.

3.3 Summary

In the first part of this chapter, we considered the software implications of working with multi-touch hardware. We saw that computer vision libraries allow multi-touch input to come from specialized applications (e.g., *Community Core Vision*), and that specific libraries can then be used to interpret the touch events in our software. The software designer should consider different selection strategies, the possibilities of continuous controls, and the support of gestures when designing his or her software. We also described a few innovative GUI controls that go beyond traditional WIMP interaction and will be considered in the software design of our DJing environment.

The second part of the chapter provided a literature review of musical applications for single-touch and multi-touch technology. We identified software that allows different musical tasks to be performed: music exploration, music control (as well as sequencers and score editors), music synthesis, and music editing. Although we focused on academic examples of multi-touch software, we point out the fact that software distribution platforms for mobile devices, such as the App Store (for Apple devices) or Google Play (for Android devices) contain various music-based applications. They are of various levels of control and complexity, and it should be noted that some software from expensive pieces of hardware has even been ported to those platforms (e.g., *reacTable mobile*[11] and *Liine*[12]).

Many of the commercial mobile device and tablet applications mimic a traditional DJing interface. This is the case of *djay*[13], for the *iOS* operating system, and *DJ Studio 4*[14] for *Android*, two popular apps for touch-enabled mobile devices. This type of software has the merit of bridging the knowledge gap between DJs and music lovers by providing affordable

[11]http://www.reactable.com/products/mobile/, accessed 2012/03/11
[12]http://liine.net/en/products/lemur/, accessed 2012/03/11
[13]http://www.algoriddim.com/djay-ipad/, accessed 2012/12/10
[14]https://play.google.com/store/apps/details?id=com.beatronik.djstudiodemo&hl=en/, accessed 2012/12/10

access to similar tools as those used by professional DJs. An app such as *Skillz*[15] goes even further in teaching music lovers about traditional DJing by leading them through DJ sets in a gaming context. However, our view is that there is more research and entertainment value in using DJing tools to facilitate music creation than in allowing users to perform the baby scratch on a touch tablet. The approach taken by *MixxMuse DJ*[16], an application that lets users sequence samples in a controlled, on-pitch and on-beat environment, is more in line with our idea of a ludic software application for DJ-like music creation.

The next chapter reviews the concept of metaphor and discusses conceptual integration (or blending) in the context of software design.

[15]https://itunes.apple.com/us/app/skillz-for-ipad/id456617069?mt=8/, accessed 2012/12/10
[16]https://itunes.apple.com/us/app/mixxmuse-dj-pro-hd/id373544145?mt=8/, accessed 2012/12/10

Chapter 4

Metaphors, Blending, and Software Design

This chapter reviews the concepts of metaphor and blending (or conceptual integration), and highlights the use of these concepts in multi-touch software systems for musical expression.

4.1 Image Schemata, Metaphors, and Blends

4.1.1 The Traditional View of Metaphor

Lakoff introduced the traditional view of metaphor with the following sentence: "The word "metaphor" was defined as a novel or poetic linguistic expression where one or more words for a concept are used outside their normal conventional meaning to express a "similar" concept." [94] In this traditional view, metaphors are based on similarity, and are purely a matter of grammar, syntax, and figures of speech. These fallacies had previously been exposed by Lakoff and Johnson [95], cognitive scientists and philosophers.

4.1.2 The Contemporary View of Metaphor

In their 1980 book [95], Lakoff and Johnson argues that metaphor is much more than a poetic tool. For them, it pervades everyday thought, and shapes the way we think, as this concept is rooted in our experience, both cultural and physical. Example such as Time is money and Happy is up respectively illustrate the cultural and physical origins of such metaphors. The authors distinguish conceptual metaphor from linguistic metaphor, the latter corresponding to the litterary use of the concept. Zbikowski discussed the elements involved in a metaphor: conceptual domain, image schema, and repeated patterns of embodied experience [96]. Lakoff's invariance principle determines that the transferred properties of the source domain retain as much of their initial characteristics as is allowed by the target domain.

Lakoff and Johnson argued that metaphors highlight certain aspects of the comparison while hiding others: parts of the metaphor are left "unused". They also discussed consistence and coherence between different metaphors that describe the same concept: "The two metaphors would be consistent if there were a way to completely satisfy both purposes with one clearly delineated concept. Instead, what we get is coherence, where there is a partial satisfaction of both purposes." [95] This distinction explains how it is possible to use different metaphors for the same concept.

Metaphors result in inferences: they are concepts derived from the choice of metaphor. For example, inferences for the metaphor Time is money are that time can be wasted or exchanged like any commodity. Inferences are also known as entailments [97].

Lakoff and Johnson present different types of metaphors that we list below:

- Structural metaphors (e.g., Argument is war) are used to structure one concept in terms of another one;
- Orientational metaphors (e.g., Happy is up) are rooted in physical and cultural experience. They use simple spatial concepts such as down or up;
- Ontological metaphors (e.g., The mind is a machine) are "ways of viewing events, activities, emotions, ideas, etc., as entities and substances";
- Personnification metaphors (e.g., A country is a person) relate concepts to human beings;
- Dead metaphors are marginal metaphors that appear in certain expressions (e.g., a head of cabbage, the foot of the mountain) but use only one aspect of the metaphor. Possibly derived from metaphors (or personnifications) prevalent in previous times, they now only survive through isolated idioms.

In the 2003 appendix to their book, Lakoff and Johnson argued that the lines between these different types of metaphors are blurry, and that in fact most metaphors belong to several such categories, and all are structural and ontological. This appendix also provides an extensive definition of conceptual metaphor: "In a metaphor, there are two domains: the target domain, which is constituted by the immediate subject matter, and the source domain, in which important metaphorical reasoning takes place and that provides the source concepts used in that reasoning. Metaphorical language has literal meaning in the source domain. In addition, a metaphoric mapping is multiple, that is, two or more elements are mapped to two or more other elements. Image-schema structure is preserved in the mapping — interiors of containers map to interiors, exteriors map to exteriors; sources of motion to sources, goals to goals, and so on."

4.1.3 Blending, or Conceptual Integration

Fauconnier and Turner [98] introduced blending, or "conceptual integration", as another way to combine seemingly unrelated concepts. While metaphor maps one conceptual domain to another, blending incites us to build an extra mental space, as well as an abstract space, that are combinations of both input domains. Blends are not always based on metaphor, but metaphors are a type of blend.

The four components of a blend are illustrated below (the example is ours):

- Two input spaces: they are mental spaces, each containing elements of different kinds (e.g., weather, people, concepts), and are structured by frames (prototypical situations). As an example, take mental space one: "the Canadian winter", yourself, and the frame "shoveling the car in the morning". Take mental space two as: "a Caribbean beach", an idealized (younger, fitter) you, and the mental frame of "a morning swim";

- Cross-space mappings: they are connections between counterpart in the input spaces. In our example, "the Canadian winter" is mapped to "a Caribbean beach", you are mapped to the idealized version of yourself, and "shoveling the car in the morning" is related to "a morning swim";

- A generic space: the space of shared elements and core links between the elements indentified by the cross-space mappings. This space would include extreme weather, granular elements (sand and snow) in the environment, and a morning ritual;

- The blend: a new mental space that has elements of both input spaces (and their mappings), plus new emergent structures taken from either input space but not shared between the two input spaces. In our example, the following inferences are possible in the blended space: shoveling sand from your Canadian driveway, getting a cold because of a freezing sandstorm, and melting sand in winter.

Blends can be single-scope or double-scope networks. In single-scope networks, the frame of only one of the input spaces is that of the blend, while in double-scope networks, the blend's frame is a combination of the frames in both input spaces.

The advantage of using blending over metaphor is the emergence of novel structures of thought (such as freezing sandstorms or melting sand in the preceding example). We intend to use this strength in the software design of our musical environment.

4.1.4 User Interface Metaphor and Blending for Software

User interface metaphor is the inclusion of graphical elements from a known domain in the user interface of some software. Such metaphors can exploit many aspects of the input space (e.g., operating systems using the office desktop metaphor include waste bins, folders containing documents, and a clock).

Alty et al. [99] presented a methodology for using metaphor in computer systems design and evaluating the effectiveness of such metaphors. Their analysis of metaphor was supported by set theory, a mathematical tool that enables the designers to quantify features that lie outside the intersection of both sets, in order to limit invalid inferences. The authors discuss the influence of "conceptual baggage", namely the inferences of the source domain that are left unused in the implementation of the metaphor.

The method proposed by Alty et al. to design computer systems using metaphor consists of six steps:

1. Identifying system functionality;
2. Generating and describing potential metaphors;
3. Analyzing metaphor-system pairings using set theory;
4. Implementation and analysis of representation, realism, and consistency;

5. Evaluation of the use of metaphor;

6. Feedback on design.

Imaz and Benyon [100] highlight several metaphors that are common in interaction with computers:

- Data is stored in "files" and "folders": ARCHIVED DIGITAL INFORMATION IS A PAPER DOCUMENT;
- Programs are written in a "language": MACHINE CODE IS A NATURAL LANGUAGE;
- Programs are "executed" on a computer: COMPUTERS ARE COGNITIVE BEINGS, in that they can execute tasks;
- We "cut" and "paste" text: DIGITAL TEXT MANIPULATION IS COLLAGE ART.

Imaz and Benyon mention the fact that the use of metaphor in the design of digital media is both praised for simplifying tasks (the desktop metaphor provides a more convenient way of accessing files than a command line does) and criticized (because emergent functions can be misleading, such as the dragging of a floppy disk onto a garbage bin to physically eject the floppy disk in early versions of Mac OS) [100].

Illustrating Lakoff and Johnson's view that metaphor both reflects and influences our way of thinking, Imaz and Benyon explain the transition from procedural languages to object-oriented languages as a shift from SOFTWARE IS A SEQUENCE OF STEPS to SOFTWARE IS A COLLECTION OF OBJECTS.

In the presentation of their concept "The tangible memories box", Hurtienne and Israel [43] explain their choices in terms of metaphors as well as image schemas. This method is used to justify design choices and highlight inferences.

Imaz and Benyon's book introduced a method for applying conceptual integration, or blending, to the task of designing software:

1. Understanding which input spaces and frames are involved (traditional software design analysis);

2. A lower level of abstraction: refine the concepts, take into account the constraints of technologies (design of the GUI and perceptual features, possible actions). It is an iterative process through various levels of abstraction.

4.2 Metaphor and Design

Saffer [101] argues that metaphor can be used in different ways in the design process: to redefine problems, as a research tool for unfamiliar areas, as inspiration during brainstorming, as a communication device (a common ground for everyone involved in a project), or applied to the design process itself (to compare it to another task).

Metaphor can be identified at different levels of an instrument: the *MetaMuse* [102] was initially said to use "the metaphor of rainfall to make the process of granular synthesis understandable", but the authors then identified three more refined metaphors (pouring, falling on a sink, and a virtual landscape).

Fels et al. [103] presented metaphor as a method of increasing the expressiveness of an instrument, placing it in a more expressive region of their graph of audience and player transparency axes by facilitating the understanding of the instrument by the user and audience. Fels et al.'s objective was to "facilitate the acceptance of novel controllers into the literature to allow for new forms of expression" [103]. Interestingly, the authors' *Glove-TalkII* system puts the performer in charge of choosing the metaphor, by training neural networks based on the user's movements.

Duignan et al. [104] analyzed two popular music production applications, *Reason* and *Live*, to identify and compare metaphors that appear in the applications. The authors highlighted the fact that the metaphors upon which these interfaces were designed might not be very meaningful to new users, given the fact that more and more of them have no experience of studio hardware, but might have interest and talent in music production. We strongly agree with this latter point, and justify our focus on conceptual blending and metaphor by hoping to simplify the learning process of a music production environment through the use of uncommon metaphors.

In their presentation of the *A20*, Bau et al. [105] explored three metaphors in the design of their interface: instrument building, composition, and expressivity of interpretation. The *A20* was "a polyhedron-shaped, multi-channel audio device that allows direct manipulation of media content through touch and movement, with various forms of aural and haptic feedback." It was a prototype meant to test out design ideas for interaction with a tangible audio device. Fishkin, while developing a bidimensional taxonomy for tangible interfaces [20], used the axes of embodiment and metaphor. The metaphor axis consists of the three combinations of "metaphor of noun" (related to the shape of the tangible object) and "metaphor of verb" (related to the motion of the tangible object) plus the absence of metaphor and the full use of a metaphor.

Wessel and Wright [106] discussed several metaphors that served as inspiration for their controller software: drag and drop, scrubbing, and dipping, although their work focuses more on a two-handed digitizing tablet than on multi-touch interfaces for fingers. We argue that these actions do not represent metaphor as we have presented them here, but are instead gestures.

Dahl and Wang [107] introduced *SoundBounce*, a piece for CCRMA's Mobile Phone Orchestra, based on the metaphor SOUND IS A BALL that can be thrown and passed around. Critical of their work, the authors argue that their simplistic choice of metaphor risks restricting the "active interpretation and multiplicity of understanding that good art allows" [107]. We praise this choice of metaphor for emphasizing the collaborative aspect of music performance, but insist on the fact that the source domain (a ball) is too poor to yield interesting inferences. Perhaps refining it (such as SOUND IS A BASEBALL or SOUND IS A STRESS RELIEF BALL) could generate richer inferences, by drawing upon schemas such as a sports game or bestowing a particular texture, weight, dimension, and possibly purpose (we anticipate users to quickly get bored of pointlessly passing a ball around) to the target domain.

4.3 Interface Metaphors in Touch-Enabled Musical Software

In this section, we mention musical applications with multi-touch interactionns, and briefly describe the frame being invoked by the (implicit or explicit) use of metaphor in user interface design. Our purpose is to show the variety of domains from which one can draw inspiration when consciously designing software with metaphor.

Installations using interface metaphors inspired from the natural world include *Small-fish*[1] (an image-based chamber music generator), *Zen Waves*[2] (a zen garden), and *Froggies*[3] (a frog pond). The *Roots*[4] application implemented a sequencer using a vine-like structure in the GUI, on a system supporting fiducials [83].

[1]http://hosting.zkm.de/wmuench/small_fish, accessed 2010/11/10
[2]http://www.youtube.com/watch?v=_H4WnZ5D_LQ, accessed 2010/11/10
[3]http://www.jtnimoy.net/itp/froggies/, accessed 2010/11/10
[4]http://www.youtube.com/watch?v=v9Yt5vUD2Xc, accessed 2010/11/17

The *SoundRose* installation [3] used the metaphor of roses on its display surface, and used the blob information to control various parameters of a sound synthesis engine.

ANTracks [108] used the metaphor of ants moving on a hexagonal grid to interact with a pitch pattern. Here, the passing of time was represented by the motion of ants on a grid instead of just being a playhead on a timeline, as is the case in most sequencers. The *Scrapple* project [85] used the metaphor of a sewing desk to perform spectrographic sequencing in an augmented reality environment. Instead of rethinking the passing of time, this project emphasized the laying out of sounds on the timeline. Being actual pieces of fabric, these sounds have different shapes and sizes. An unused aspect of the metaphor is the texture of fabric. Understandably, this feature is delicate to extract using cameras, but could be replaced by fiducial patterns on different pieces of fabric. The *Stereotronic Multi-Synth Orchestra*[5] used a metaphor of concentric rings to implement a sequencer.

Multi-touch environments using metaphors that are not related to the natural world include *Composition on the Table*[6] (spinning disks, an electronic circuit with switches, and sliding plates) and the *MUSICtable* [109] (geographic map). Another project involving geographic maps is *Weather Report* [35], in which the temperature of regions in the USA affected the timbre of sounds.

In the 1980s, Johnstone [1] introduced the metaphor of a storeroom for his Rolky installation: materials, tools, and operational devices are stored and invoked on the controller. Similarly, Nakatani and Rohrlich [21] present a metaphor for the design of software interfaces for "soft machines" (software with evolving GUI, as opposed to hardware) such as touch tables, based on a three-level structure:

- Tool bin: "the entire collection of tools available on a particular computer";
- Workshop: "a work environment specialized for a particular type of work or task [...]";
- Workbench: "analogous to a work surface or counter in the workshop where the actual work is done. On the workbench are tools needed just for the current task".

Similarly, the choices of Johnstone on one part, and Nakatani and Rohrlich on the other, are much akin to the choice of the desktop metaphor in operating systems. Instead

[5]http://vimeo.com/6859653, accessed 2010/12/07
[6]http://www.ntticc.or.jp/Archive/1999/+-/Works/conposition_e.html, accessed 2010/11/10

of focusing on a traditional shirt-and-tie or artistic office setting (file folders, recycling bins, collages), these user interfaces emulate environments more familiar to blue-collar workers or anyone who has ever done some kind of renovation work. We see this distinction as a powerful tool in catering to different segments of the global population. Beyond the fact that multi-touch technology has not reached (and will likely not reach) all remote populations who are unfamiliar with these settings, we think that our choice of user interface metaphor should be as intuitive as possible to all segments of population, particularly given that one of our objectives is the design of a ludic environment.

Wessel and Wright [106] presented various metaphors for use on their custom multi-touch hardware. These include mapping coordinates to 2D spaces, dipping (changing the dynamics of a sound), navigating the 2D space of a group of oscillators, granual synthesis, crossfading between four sound sources, and resonant filters.

Wessel et al.'s array of force-sensitive controllers [24] used the metaphor of a brick wall, but only in its aesthetic look. The metaphor of a painter's palette was used to provide users of the *WallBalls* system [86] access to the different tools and samples available. The palette metaphor is also employed in Roma and Xambó's waveform editor for live performance [93]. This metaphor is widespread in modern image editing software, where colours and other tools are laid out to the side of a canvas.

Wessel et al.'s [83] *AhText!* installation used a mandala metaphor, while the *Maps* application represented data in the style of a mosaic.

Bragdon et al. used three metaphors "to motivate gesture learning through positive reinforcement" [110] in a puzzle environment: buttons, springs, and physical props (such as a car wheel). The authors' motivation was to teach novice users a set of gestures. We praise the use of interface metaphors to facilitate the learning of a software environment.

Cicconet et al. [89] used the metaphor of a guitar neck to allow improvisation on the blues scale. A system presented by Malik et al. [111] allowed the multi-user control of a large wall display by manipulating the desktop metaphor with multi-touch gestures.

4.4 Summary

We have approached the concept of metaphor from the traditional poetic viewpoint, before showing through Lakoff and Turner's work that this concept is ubiquitous in human thought. A more recent development is the appearance of conceptual integration, or blending, which provides a structure for innovative concepts to emerge out links between conceptual spaces (metaphor being such a link). Having presented software engineering research in blending, we analyzed some multi-touch musical software environments with this newfound knowledge. This allowed us to document and qualify the used and unused parts of metaphors and blends.

We will now focus on using these concepts in designing our own multi-touch musical software environment. Alty et al. and Imaz and Benyon recommend different approaches for designing software with blends and metaphors. Instead of following these approaches literally, we will perform a more informal type of software design, informed by the strengths of conceptual integration and metaphor.

Chapter 5

TactoSonix: Software Design

This chapter presents the software design of *TactoSonix*, beginning by high-level considerations before presenting our design choices. Our software design methodology combines conceptual integration and elements of traditional software design.

5.1 Design Spaces as Informal Functional Design

One must choose where in the seemingly endless design spaces of interfaces the system will be placed. We shall discuss several such spaces: empirical classification, contexts, and Rowe's rough classification [112]. These considerations will give us high-level answers to questions that are frequently discussed in the functional design of software, and allow us to make design choices focused on our objective of providing an accessible platform for the creation of music based on tools with which the general public might not be familiar.

5.1.1 Empirical Classification

Drummond's [113] empirical classification builds upon Bongers' [114] and includes:

- Performer with system;
- Audience with system;
- Performer with system and audience;
- Multiple performers with a single system;

- Multiple systems interacting with each other and/or multiple performers.

Our system will be of the first type, a performer with the system, although the multi-touch platform will make it possible for multiple performers to interact with the software. As our immediate motivation is not to implement a professional solution for performance in front of audiences, we do not see a need to consider the second, third, or fourth types of system. We would nonetheless like to point the reader towards a system that implements those types of interaction: *Turntable.fm*[1] allows users to perform DJ sets for virtual rooms of real listeners. The fifth type of interaction in this classification could be of interest in further iterations of our software, in that we can imagine the possibility of a networked music creation environment in which users could, for instance, use as their own loops the audio output of other users' mixes.

5.1.2 Contexts in Interactive Computer Music

An important question that comes to mind when creating a musical application is what kind of sound control will be offered to the user. This software-related (in our case) choice has implications on the set of sounds or tones available, as well as on the design of the user interface. Wanderley and Orio [115] present a list of contexts in interactive computer music, including:

- Note-level control;
- Score-level control;
- Sound processing control;
- Traditional HCI controls: higher-level sound control (e.g., controlling a bank of oscillators or navigating a timbre space) via metaphor;
- Interaction in multimedia installations: when sound is part of a larger experience.

Unless we impose tonal restrictions, note-level control implies a minimum level of music theory knowledge from the user, which goes against our main motivation. We intend to provide both score-level control (as a sequencer does) and sound processing control (through audio effects) to users of our system. Although we will make use of metaphor, our software

[1]http://turntable.fm/, accessed 2012/12/10

will not fit the fourth context because of the conceptual complexity of such music technology systems to novices. Sound will be the main component of our system, therefore the fifth context does not apply to our application.

5.1.3 Rowe's Rough Classification

Rowe's rough classification [112] of interactive systems uses a combination of three dimensions:

- Score-driven vs performance-driven: whether or not the system follows "predetermined event collections, or stored music fragments" [112];
- Transformative vs generative vs sequenced: does the system create variants of existing material, synthesise audio from basic sets (scales, duration sets), or control the sequencing of prerecorded contents?;
- Instrument vs player paradigms: is the system designed as an extension of the performer, or as a virtual player?

The system we foresee will be performance-driven (in that loops will be loaded from a library), sequenced (the audio loops are prerecorded), and will focus on the player paradigm (we are developing a software system to be run on hardware with the form system of a coffee table or desk, and possibly on tablets).

5.2 Nonfunctional Design

We now explore features that are not immediately related to the behaviour of the system, but that affect its operation nonetheless.

5.2.1 Multi-user

Multi-user systems include the *reacTable* [9] and *DTLens* [78]. These systems do not recognize different users, as is possible with *DiamondTouch* [26], but are rather designed for concurrent visualization and interaction by different people.

The concept of private space [116] for voting purposes or the visualization of personal data has also been explored by some researchers in collaborative environments.

Our system will not be designed for multiple users, but the use of tabletop-sized FTIR and RDI multi-touch technology means that in practice, several users will be able to interact with the system.

5.2.2 Latency and Frame Rate

Depending on the type of interaction required, stricter requirements on the latency of a controller will be imposed. An accepted lower limit for percussive instruments is 10 ms [87][106], meaning 100 FPS.

Our system will focus on the higher-level control of recorded musical contents, not on note-level control, for all tracks. As such, we do not anticipate the need to aim for the aforementioned lower limit on latency. The typical frame rates of contemporary cameras, namely 30 fps to 60 fps, should be more than enough for our application.

5.2.3 Discrete vs Continuous Controls

Wessel and Wright [106] highlight a need for the continuous control of parameters, and present their Open Sound Control (OSC) protocol as an improvement over MIDI in that it offers the synchronized transmission of events as well as a mechanism for atomic updates. It is worth noting that their focus is on "improvised interactive live performance, in collaboration with improvising acoustic musicians" [117].

We understand the dullness of event-based musical systems and stress the importance of continuous controls in our software.

5.3 Designing with Blends

In this section, we present two metaphors for a user interface dedicated to DJing. Lakoff identified metaphor in fields as varied as mythology and foreign policy [94]. We use his approach in the field of music, more specifically in the task of controlling the playback of recorded audio samples.

5.3.1 Design Choices

In order to provide a ludic platform for the high-level control of pre-recorded music, we decide to facilitate certain tasks so that the user can focus on more creative aspects of

DJing. For instance, all tracks will have their beats matched, and loops will always be properly synced. The sound library will be predefined and not customizable by the user, which will ensure that all loops are of the same time signature and tempo. Additionally, crossfading will be automated.

5.3.2 Metaphor One: Making music is cooking

The first metaphor we study is MAKING MUSIC IS COOKING. We illustrate the link between DJing and cooking[2,3] by pointing at DJ/hip-hop terms such as “blend”, “mix”, “fresh”, and “cut”.

The user will be presented with a symbol representing a burner or cooking pot, in which elements will be added. Table 5.1 details our choice of inferences for this metaphor, while table 5.2 explains how the features of DJ equipement are implemented.

Inference	**Implementation details**
Combining elements creates a coherent whole	As with cooking, different elements are necessary to create a good result. In this case, loops are split in three categories: percussions, bass, and lead instrument.
Hotter means louder	The proximity of loops to the centre of their burner will determine the amplitude of the sound.
Different pots have different elements	While cooking, people naturally cook different meals in different pots. For us, these pots will represent individual channels of music.

Table 5.1 An explanation of the MAKING MUSIC IS COOKING metaphor

[2]http://www.omelette.net.au/release/gourmet_scavenger, accessed 2011/02/20

[3]http://www.allmusic.com/album/pass-the-peas-the-best-of-the-jbs-r486032, accessed 2011/02/20

Feature	**Availability**	**Implementation details**
Channels and tracks	✓	Each burner will represent a channel of audio, and each cookable element will be a track on these channels.
Crossfading	✓	Crossfading will be automatic when moving towards other burners.
Cues	✗	Not implemented.
Volume	✓	The closer a cookable element will be to the centre of the pot, the louder it will be.
Bass, mid, and high filters	✓	A lowpass filter will be provided by the position of a cookable element within the pot.
Audio effects	✗	Not implemented.
Pitch shifter	✗	Not implemented.
Tempo shifter	✗	Not implemented.
Song syncing	✓	All tracks will be in sync.
Looping	✓	A section of each pot will be split in four to provide looping of four lengths: one, two, four, or eight beats.
Sequencing	✓	By adding and removing elements from the pot, the user will change the sequencing of the tracks.

Table 5.2 The implementation of DJ equipment features in the cooking metaphor. See Section 1.1 for an explanation of these features.

5.3.3 Metaphor Two: Making music is gambling

In this metaphor, the user is responding to cues from non-human players, as his or her role is that of a croupier watching over a group of gamblers. Random elements, in the form of card draws, will impact the sound. The user will not be performing croupier tasks literally, since he or she will be able to perform tasks such as modifying the amount bet by gamblers.

Table 5.3 details this metaphor, while table 5.4 explains how each DJ equipement feature could be implemented. Note that only metaphor one is actually implemented in this thesis.

Inference	**Implementation details**
Randomness	Any gambling task involves some kind of randomness. The choice of track that will be assigned to gamblers will be random.
The volume is the bet amount	Increasing and decreasing the amount that is bet changes the volume. We justify this choice by the fact that more money in jeopardy means more expectations, energy, or stress.
Gamblers are adressed in sequential order	Much like a croupier has to interact with gamblers one at a time, the DJ could be forced to address the audio channels one at a time.

Table 5.3 An explanation of the MAKING MUSIC IS GAMBLING metaphor. See 1.1.

5.4 Implementation

We now present our implementation of the first metaphor presented, MAKING MUSIC IS COOKING. It is the result of our reflexion about metaphor-based user interface, and also uses innovative GUI elements. We call our multi-touch DJing software *TactoSonix*, in expectation of the development of new metaphors using the same framework.

5.4.1 Metaphor and User Interface

Table 5.2 documents our implementation of certain DJ tools within the MAKING MUSIC IS COOKING metaphor.

Feature	**Availability**	**Possible implementation details**
Channels and tracks	✓	Different gamblers would represent different audio tracks, and different tables of gamblers will represent audio channels.
Crossfading	✓	Crossfading would be controlled by the speed at which the croupier moves from one table to the next.
Cues	✓	Could be implemented by making the joker card appear.
Volume	✓	By adjusting the amount bet by players (including the croupier), the user will be able to change the volume of each track.
Bass, mid, and high filters	✓	By cheating on a gambler, the croupier could apply these filters to the matching track.
Audio effects	✓	The Jack, Queen, King and Ace cards could be used to represent audio effects.
Pitch shifter	✗	We do not deem this effect absolutely necessary.
Tempo shifter	✗	We do not deem this effect absolutely necessary.
Song syncing	✓	All tracks will be in sync.
Looping	✓	As long as a gambler is at the table, his or her track will keep looping as part of the main mix.
Sequencing	✓	By controlling the players at the table (kicking them away, replacing them, keeping them), the croupier is in charge of sequencing.

Table 5.4 The implementation of DJ equipment features in the gambling metaphor. See Section 1.1 for an explanation of these features. Note that this metaphor is not implemented in this thesis.

The first view of the Making music is cooking software environment is shown in Figure 5.1. When the user enters the system, a kitchen burner (or cooking pot) is shown, its transparency pulsating at a constant rhythm. The point on the edge of the burner moves discretely at the same rhythm, in a clockwise manner between 16 positions. It indicates the current beat within a 16-step sequencer. The diagram in the top left corner of the interface shows which of the four burners the user is currently watching.

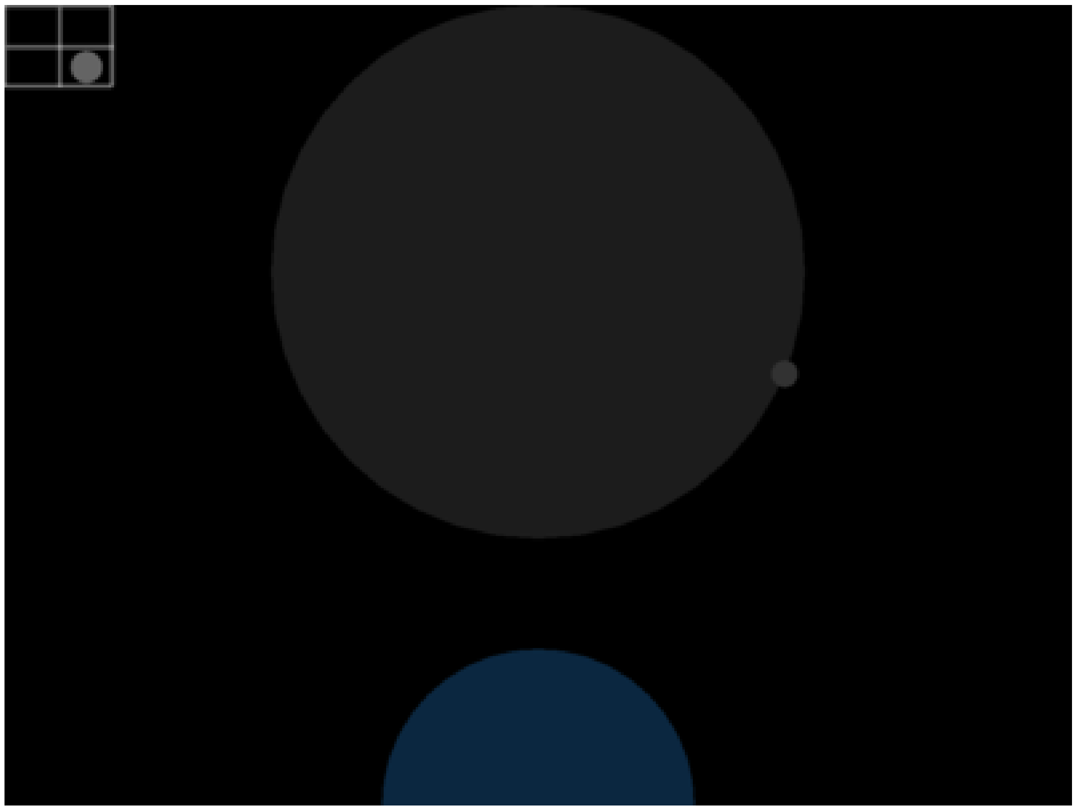

Fig. 5.1 The first screen of the TactoSonix cooking metaphor environment

By exploiting the metaphor of a kitchen top and using the top left diagram, we wish to lead the user towards an intuitive expectation of more than a single such burner. By dragging a single finger on the surface while not touching the burner, the user will be able to navigate between the different stovetops, as shown in Figure 5.2.

The blue half-circle at the bottom of the GUI is always shown, and represents the root node of a *Stacked Half-Pie Menu* (SHPM) [4] (see Section 3.1.6). This menu opens a first level of nodes, from which a second level can be triggered. An illustration of the

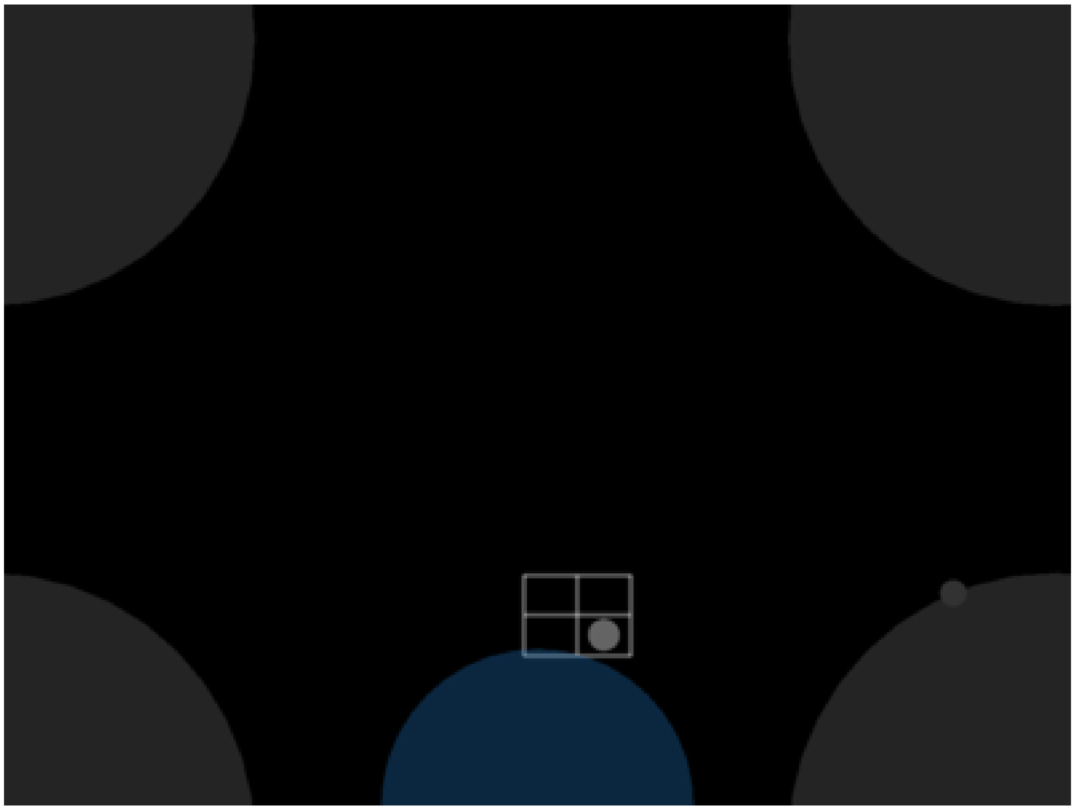

Fig. 5.2 A focus position resulting in an approximately equal contribution of each audio channel

unwrapped menu is Figure 5.3. The first level has three polygons with a different number of sides representing three groups of samples: percussions (square), bass (pentagon), and lead instrument (hexagon). The second level of nodes represent the samples themselves. As such, all nodes at the second level have the same number of sides. They differ by colour: red represents country music, purple represents funk, blue represents hip hop, and green represents jazz.

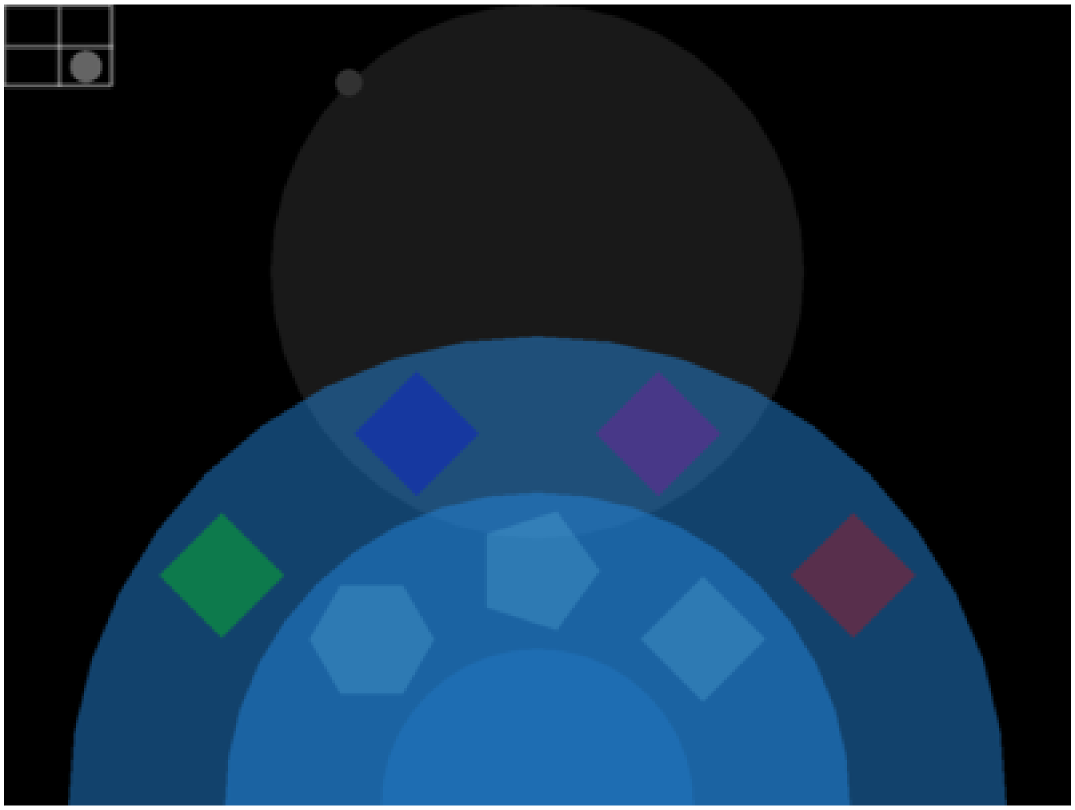

Fig. 5.3 The unwrapped *Stacked Half-Pie Menu* showing a choice of audio tracks. The first menu represents – from left to right – lead instruments, bass, and percussions. The second menu represents – from left to right – jazz drums, hip hop drums, funk drums, and country drums.

5.4.2 Interaction and Visual Feedback

Once open, the software will display a collapsed *Stacked Half-Pie Menu*, as is visible in Figure 5.1: with one touch or click, the user unrolls the next level of the menu. Trying to expand the leaf node of a SHPM, the user will discover that these nodes can be drag-and-dropped. Dropping such a loop node randomly makes it disappear, but dropping it on the pulsating burner illustration will keep the node alive, and will start playing the associated loop.

Once dropped on a burner, a node can be moved around. Doing so will both affect the sound and bring up visual feedback about its track. This is shown in Figures 5.4 and 5.5. The first image shows the country drum loop (represented by a red square) drawn with a red curve going through its centre. This line's width and opacity are mapped to the volume of the track. The star-like symbol inside the visual feedback represents the cutoff frequency of the lowpass filter applied to the track. The rounder the shape, the lower the cutoff frequency. The vertical position of the node within the stovetop determines the cutoff frequency of the low-pass filter (higher means almost no filtering, lower means low cutoff frequency). This choice of representation is inspired by the work of O'Sullivan and Borland in exploring correlations between graphical controller shape and audio spectrum [87]. O'Sullivan and Borland focused on sound synthesis and used a rounder shape to represent a single tone, and a more pointed shape to represent a richer sound with many harmonics. We instead choose to represent the cutoff frequency of a low-pass filter.

The crosshair visible in Figure 5.4 appears because the user dragged a node close to the centre of the kitchen burner. This has the effect of looping the specific track over a number of beats. The number of beats of this loop is determined by the quadrant in which the node is: the top-right quadrant will trigger loops of one beat, the bottom-right quadrant will trigger loops of two beats, the bottom-left quadrant will trigger loops of four beats, and the top-left quadrant will trigger loops of eight beats.

One can then understand that a simple drag-and-drop of nodes may affect various parameters at the same time: they can be made "integrable" [72] if they are perceived as such. In our implementation, a user may affect up to three parameters (volume, low-pass cut-off frequency, and loop length) with a single drag motion. Multi-touch interaction means that with just 3 fingers, up to nine parameters can be confused. This integrability of parameters forces the user to interact within a constrained environment, and serves our purpose by both facilitating the mixing of music and taking full advantage of multi-touch technology.

The user may only play one track per instrument type: we force the sound to be coherent by not allowing concurrent loops serving the same purpose (percussions, bass, or lead instrument) to overlap.

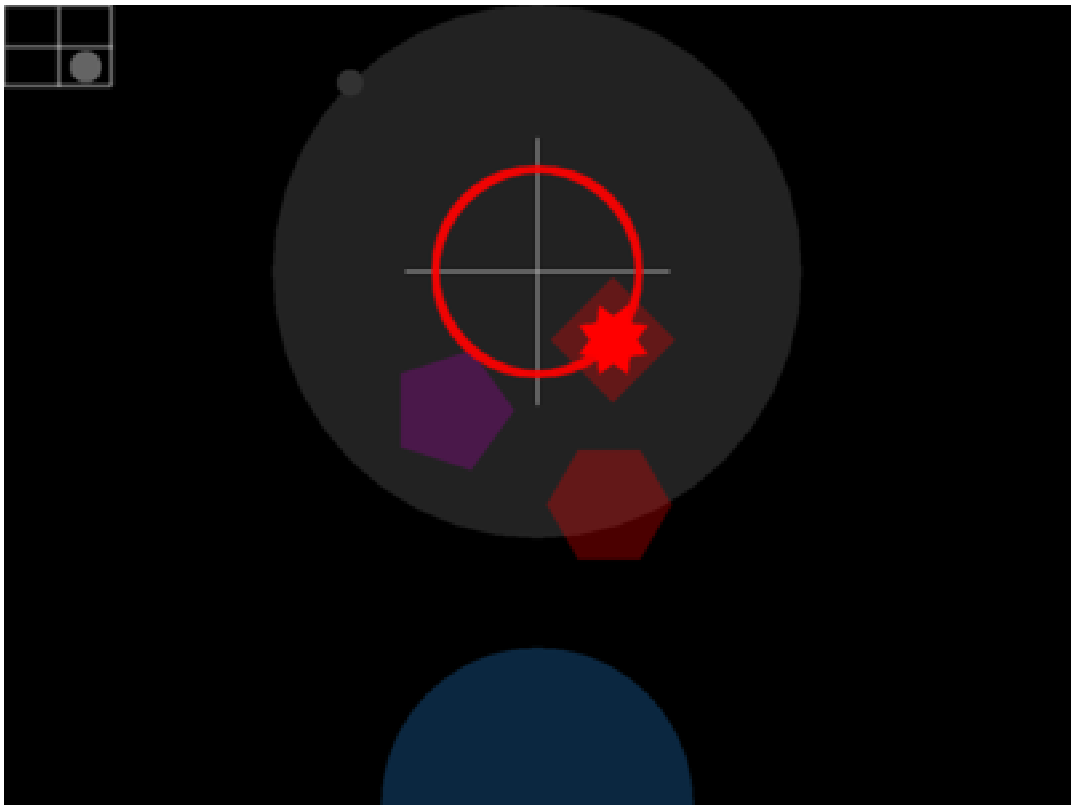

Fig. 5.4 Dragging a node around (the red square) displays visual feedback about the associated audio track's volume, low-pass cutoff frequency, and loop length.

5.4.3 Audio Environment

The audio samples of our musical software environment are categorized based on type of instrument (percussions, bass, or lead instrument) and musical genre (country, funk, hip-hop, jazz). By isolating musical genre and instrument, our choice of loops has resulted in a software environment that facilitates the cross-genre exploration of music.

The creation of musically coherent content is enforced by restrictions to the number of concurrent loops playing and by locking all the starts of samples to fall musicon beat. We leave room to creation by letting loops start on any beat (and not just on the first beat of the sequencer) and implementing a feature through which new sounds may be generated:

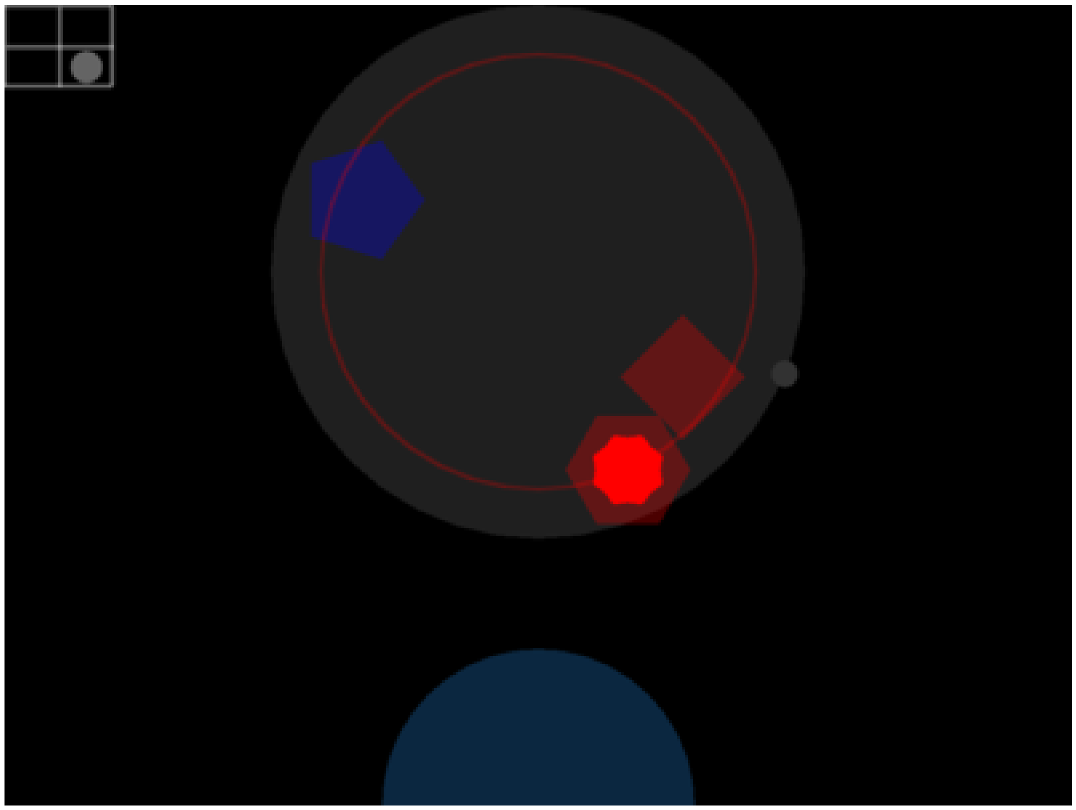

Fig. 5.5 Dragging a node around (the red hexagon) displays visual feedback about the associated audio track's volume and low-pass cutoff frequency, looping being disabled due to the node being outside the looping zone of the stovetop. The visual feedback showing that the node is not looping at one, two, four, or eight beats, is that no crosshair is visible in the cooking pot, as opposed to Figure 5.4.

the looping feature, which can drastically change the melody.

5.4.4 Comparison with Traditional DJing Equipment

This software environment is not meant as an implementation of or a substitute for DJing equipment, but as an introduction for anyone to some of the tools used by DJs. It leaves out certain features such as beat matching, pitch shifting, and tempo shifting for technical reasons (implementing low-level sound processing algorithms across different platforms was not the focus of this thesis), while implementing other features in a much different manner. For instance, crossfading is performed by dragging one finger across the background in our software. DJs perform this task using a linear potentiometer that has mechanical characteristics (e.g., weight and mid-position lock) with which a DJ is highly familiar; a DJ sometimes switches between the crossfader's two extreme positions at a rate that would be unfeasible using our software environment. Instead of providing the same level of control, we offer a simplified version of crossfading.

This software, through its user interface metaphor inspired from cooking, provides an environment for the exploration of musical concepts common in music production but not necessarily among the general public: sequencing, audio filtering, looping, and the mixing of several audio channels.

5.5 Technical Details

TactoSonix is an add-on for OpenFrameworks[4], a C++ programming framework. It provides several GUI elements that can be used in the development of multi-touch software in order to replace traditional GUI elements. For the software documentation of our system, we point the reader to Appendix A.

TactoSonix is available at http://www.idmil.org/projects/tactosonix.

5.6 Summary

In this chapter, we narrowed down our software's design by placing the musical system within different classifications for interactive environments. We then described brainstorming ideas about user interface metaphor in the form of inferences and a checklist of features,

[4]http://www.openframeworks.cc, accessed 2012/08/08

before presenting the implementation of the first proposed metaphor, MAKING MUSIC IS COOKING. This novel, informal approach to software design seems to us to best fit the brainstorming process, given that it does not provide all the implementation details that arise when exhaustive software documentation is required.

The next chapter will conclude this thesis with a discussion of our hypothesis, contributions and further possible work.

Chapter 6

Conclusions and Recommendations

This thesis presented a historical overview of touch-enabled technology and a short presentation of the task of DJing in Chapter 1, before analyzing methods for the implementation of multi-touch functionality and introducing a taxonomy of multi-touch hardware in Chapter 2. Chapter 3 discussed software-related considerations in multi-touch technology and reviewed existing multi-touch musical applications, while the concepts of metaphor and conceptual integration were introduced from the perspective of software design. *TactoSonix*, our framework for the development of novel user interfaces geared towards music production, is the subject of Chapter 5. The current chapter concludes this thesis with a discussion of our hypothesis, contributions, and possible further work.

Our hypothesis was "that the use of user interface metaphors unrelated to conventional musical hardware, combined with innovative (non-WIMP) GUI controls, provides a basis for the development of multi-touch software that facilitates certain DJ tasks, with the aim of obtaining musically satisfying results with a steeper learning curve than standard DJ equipment" (see Section 1.3).

Our implementation of such software is presented in *TactoSonix*[1]. Our software uses the *TactoSonix* framework to implement a blend – a product of conceptual integration (see Section 4.1.3) – based on the metaphor MAKING MUSIC IS COOKING.

We present our software as a first attempt at implementing innovative user interface metaphors for the purpose of mixing music. A contribution to the field of music technology, and more particularly to multi-touch software for musical applications, is an exploration

[1]http://www.idmil.org/projects/tactosonix, accessed 2012/08/07 (see Section 5.4)

into informal software design using conceptual integration and user interface metaphor. The proposal of a new taxonomy for multi-touch hardware is another contribution; it can be used to choose between different technologies when a touch-enabled system needs to be designed, and is accompanied by examples of hardware placed within the taxonomy. Our literature review of musical touch-enabled applications provides a new reference for multi-touch technology researchers and the music technology community.

The objective of our software being to provide an accessible interface for DJing, our software allows users to combine different musical genres with very little understanding of music theory. The design of *TactoSonix* was inspired by the concepts of metaphor and blending, which we hope to see used more often in the context of graphical user interface design. We applied this approach to the design of two novel metaphors, MAKING MUSIC IS COOKING and MAKING MUSIC IS GAMBLING, and implemented the former.

Limitations

The taxonomy we introduced provides a detailed overview of methods for the implementation of multi-touch functionality. We recognize the fact that it does not discuss implementation costs, or that it might become obsolete as new technologies arise. The taxonomy presented in Table 2.1 could also be improved by adding to it some of the missing examples.

The use of metaphor and conceptual integration for software design generates innovative ideas that can lead to innovative user interfaces. Such an approach is not as refined as traditional software design methods, in that it only provides general guidelines for the implementation of the metaphors. Many decisions that must be taken to finalize the software lie outside the scope of blending for software design. Despite this limitation, we argue that the power of conceptual integration to generate novel ideas out of separate concepts justifies its use, if only for the brainstorming steps of software design.

Our analysis did not focus on other figures of speech such as analogy [118], simile [118], or anaphora [41]. We invite other researchers to explore the potential application of other figures of speech to software design and music.

Further work

Several improvements could be made to our software implementation of the Making music is cooking metaphor, including control over more audio effects and a more thorough analysis of feedback options for all parameters. Performing a user study could help us formally evaluate our hypothesis. Implementing new user interface metaphors, such as Making music is gambling (see Section 5.3.3), would help highlight the power of conceptual integration for this purpose.

Developing a mobile version would be an accessible challenge, given that *TactoSonix* runs on OpenFrameworks, and that OpenFrameworks has support for both Google and Apple devices. In terms of aesthetics, we understand that the user interface and audio samples would greatly benefit from some artistic input. We invite other researchers and industry members to start developing novel interfaces using conceptual blending in order to see new ideas emerge out of the more interesting features of two conceptual frames.

The author has worked on two other software environments related to music. The first one was a port of *Different Strokes* [119] to OpenFrameworks for multi-touch support and easier cross-platform integration. *Different Strokes* was designed by Mark Zadel, an IDMIL colleague of the author, and allows users to control the playback and sequencing of audio samples with an interface based on drawn curves. The second piece of software is *FaceQuencer*[2], a camera-based sequencer that reads fiducials. It was presented by the author at *Music Hack Day Montreal 2011*[3], a weekend of music programming and electronics, where it was voted crowd favourite[4]. Combining these two environments with *TactoSonix* could lead to a new immersive multi-platform DJing environment for webcam and tablet input.

[2]http://www.idmil.org/software/facequencer, accessed 2012/08/08

[3]http://montreal.musichackday.org/2011/index.php?page=Main+page, accessed 2012/08/08

[4]http://createdigitalmusic.com/2011/10/face-sequencers-sonic-databases-automatic-dub-remixes-more-montreal-music-hackday-hacks/, accessed 2012/08/08

Appendix A

Software documentation

This section includes basic documentation for our source code in the form of graphics. It will be most helpful to people with experience in computer science, to understand the structure of our thought within the *OpenFrameworks* environment.

For the complete source code, detailed documentation, and installation instructions, please refer to http://www.idmil.org/projects/tactosonix (accessed 2012/08/13).

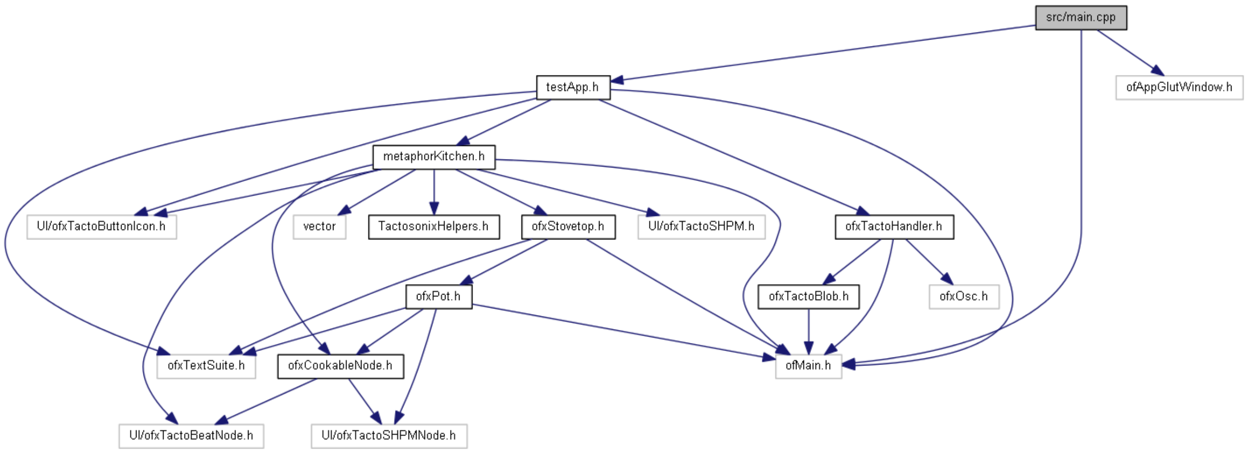

Fig. A.1 Dependency graph for main.cpp, the entry point for TactoSonix. Note the presence of the *metaphorKitchen* class, which most importantly includes a *Stacked Half-Pie Menu* (class *ofxTactoSHPM*) and a stovetop (class *ofxStoveTop*). This graph shows how TactoSonix was designed to allow for the implementation of multiple metaphors.

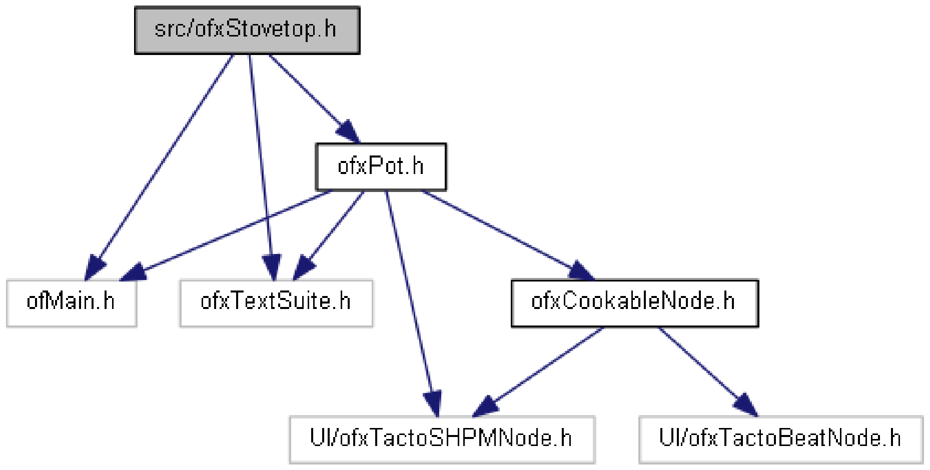

Fig. A.2 Dependency graph for *ofxStovetop*, the class that implements a stove top in *TactoSonix*. Note the fact that the stovetop class includes the *ofxPot* class: in our model, a stovetop is a cooking pot *ofxPot* (or kitchen burner) as well as some stove information (see the top-left corners of Images 5.4 and 5.5)

References

[1] E. Johnstone, "The rolky: a poly-touch controller for electronic music," in *Proceedings of the International Computer Music Conference*, pp. 291–5, 1985.

[2] L. Haken, K. Fitz, P. Wolfe, and P. Christensen, "A continuous music keyboard controlling polyphonic morphing using bandwidth-enhanced oscillators," in *Proceedings of the International Computer Music Conference*, 1997.

[3] A. Crevoisier, C. Bornand, A. Guichard, S. Matsumura, and C. Arakawa, "Sound rose: creating music and images with a touch table," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 212–5, 2006.

[4] T. Hesselmann, S. Flöring, and M. Schmitt, "Stacked half-pie menus - navigating nested menus on interactive tabletops," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pp. 173–80, ACM, 2009.

[5] H. Benko, A. D. Wilson, and P. Baudisch, "Precise selection techniques for multi-touch screens," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 1263–72, ACM, 2006.

[6] A. v. Dam, "Post-wimp user interfaces," *Communications of the ACM*, vol. 40, no. 2, pp. 63–7, 1997.

[7] W. Buxton, R. Hill, and P. Rowley, "Issues and techniques in touch-sensitive tablet input," in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pp. 215–24, 1985.

[8] G. W. Fitzmaurice and W. Buxton, "An empirical evaluation of graspable user interfaces," in *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, pp. 43–50, ACM, 1997.

[9] S. Jordà, M. Kaltenbrunner, G. Geiger, and R. Bencina, "The reactable," in *Proceedings of the International Computer Music Conference*, pp. 579–82, 2005.

[10] R. T. Murphy and L. R. Appeal, “Evaluation of the plato iv computer-based education system in the community college,” *SIGCUE Outlook*, vol. 12, no. 1, pp. 12–28, 1978.

[11] C. F. Herot and G. Weinzapfel, “One-point touch input of vector information for computer displays,” in *SIGGRAPH Computer Graphics*, vol. 12, pp. 210–6, 1978.

[12] L. Sasaki, G. Fedorkow, W. Buxton, C. Retterath, and K. Smith, “A touch-sensitive input device,” in *Proceedings of the International Computer Music Conference*, pp. 293–7, 1981.

[13] S. K. Lee, W. Buxton, and K. C. Smith, “A multi-touch three dimensional touch-sensitive tablet,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 21–5, 1985.

[14] R. A. Moog, “A mulitply touch-sensitive clavier for computer music,” in *Proceedings of the International Computer Music Conference*, pp. 601–5, 1982.

[15] J. Eaton and R. Moog, “Multiple-touch-sensitive keyboard,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 258–9, National University of Singapore, 2005.

[16] L. Haken, R. Abdullah, and M. Smart, “The continuum: a continuous music keyboard,” in *Proceedings of the International Computer Music Conference*, pp. 81–4, 1992.

[17] R. Pinkston, J. Kerkhoff, and M. McQuilken, “A touch sensitive dance floor/midi controller,” in *Proceedings of the International Computer Music Conference*, pp. 224–5, 1995.

[18] D. Wessel, “Hands on - a new work from slabs controller and generative algorithms,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 242–5, 2009.

[19] J. Y. Han, “Low-cost multi-touch sensing through frustrated total internal reflection,” in *Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, pp. 115–8, ACM, 2005.

[20] K. P. Fishkin, “A taxonomy for and analysis of tangible interfaces,” *Personal Ubiquitous Computing*, vol. 8, no. 5, pp. 347–58, 2004.

[21] L. H. Nakatani and J. A. Rohrlich, “Soft machines: a philosophy of user-computer interface design,” in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 19–23, ACM, 1983.

[22] J. D. Smith, T. C. N. Graham, D. Holman, and J. Borchers, "Low-cost malleable surfaces with multi-touch pressure sensitivity," in *Proceedings of the Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pp. 205–8, 2007.

[23] M. Gao and C. Hanson, "Lumi: live performance paradigms utilizing software integrated touch screen and pressure sensitive button matrix," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 58–9, 2009.

[24] D. Wessel, R. Avizienis, A. Freed, and M. Wright, "A force sensitive multi-touch array supporting multiple 2-d musical control structures," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 41–5, 2007.

[25] M. R. Minsky, "Manipulating simulated objects with real-world gestures using a force and position sensitive screen," in *SIGGRAPH Computer Graphics*, vol. 18, pp. 195–203, 1984.

[26] P. Dietz and D. Leigh, "Diamondtouch: a multi-user touch technology," in *Proceedings of the 14th annual ACM symposium on User Interface Software and Technology*, pp. 219–26, ACM, 2001.

[27] C. Sousa and M. Matsumoto, "Study on fluent interaction with multi-touch in traditional gui environments," in *Proceedings of TENCON - IEEE Region 10 Conference*, pp. 1–4, 2007.

[28] Y. Jansen, T. Karrer, and J. Borchers, "Mudpad: tactile feedback and haptic texture overlay for touch surfaces," in *ACM International Conference on Interactive Tabletops and Surfaces*, pp. 11–4, ACM, 2010.

[29] R. Verrillo, "Vibration sensation in humans," *Music Perception*, vol. 9, no. 3, pp. 281–302, 1992.

[30] C. Chafe, "Tactile audio feedback," in *Proceedings of the International Computer Music Conference*, pp. 224–5, 1993.

[31] D. M. Birnbaum and M. M. Wanderley, "A systematic approach to musical vibrotactile feedback," in *Proceedings of the International Computer Music Conference*, pp. 397–404, 2007.

[32] M. Giordano and M. M. Wanderley, "A learning interface for novice guitar players using vibrotactile stimulation," in *Proceedings of the 8th Sound and Music Conference*, 2011.

[33] J. Schöning, P. Brandl, F. Daiber, F. Echtler, O. Hilliges, J. Hook, M. Löchtefeld, N. Motamedi, L. Muller, P. Olivier, T. Roth, and U. von Zadow, "Multi-touch surfaces: a technical guide," tech. rep., Technical University of Munich, 2008.

[34] P. Bottoni, R. Caporali, D. Capuano, S. Faralli, A. Labella, and M. Pierro, "Use of a dual-core dsp in a low-cost, touch-screen based musical instrument," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 394–5, 2007.

[35] J. Hochenbaum and O. Vallis, "Bricktable: a musical tangible multi-touch interface," in *Proceedings of the Berlin Open Conference*, 2009.

[36] J. Loviscach, "Two-finger input with a standard touch screen," in *Proceedings of the 20th annual ACM symposium on User Interface Software and Technology*, (1294239), pp. 169–72, ACM, 2007.

[37] J. P. Carrascal and S. Jordà, "Multitouch interface for audio mixing," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 100–3, 2011.

[38] T. Beamish, K. Maclean, and S. Fels, "Manipulating music: multimodal interaction for djs," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 327–34, ACM, 2004.

[39] K. F. Hansen and R. Bresin, "Mapping strategies in dj scratching," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 188–91, 2006.

[40] K. F. Hansen, *The acoustics and performance of DJ scratching.* PhD thesis, School of Computer Science and Communication, Royal Institute of Technology, Stockholm, Sweden, 2010.

[41] L. Zbikowski, *Conceptualizing Music: Cognitive Structure, Theory, and Analysis.* Oxford University Press, 2005.

[42] C. Palombini, "Machine songs v: Pierre schaeffer: from research into noises to experimental music," *Computer Music Journal*, vol. 17, no. 3, pp. 14–9, 1993.

[43] J. Hurtienne and J. H. Israel, "Image schemas and their metaphorical extensions: intuitive patterns for tangible interaction," in *Proceedings of the 1st international conference on Tangible and Embedded Interaction*, (1226996), pp. 127–34, ACM, 2007.

[44] D. Kammer, G. Freitag, M. Keck, and M. Wacker, "Taxonomy and overview of multi-touch frameworks: architecture, scope and features," in *Proceedings of the Workshop on Engineering Patterns for Multitouch Interfaces*, 2010.

[45] J. A. Pickering, "Touch-sensitive screens: the technologies and their application," *International Journal of Man-Machine Studies*, vol. 25, no. 3, pp. 249–69, 1986.

[46] K. Hinckley and M. Sinclair, "Touch-sensing input devices," in *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, pp. 223–30, ACM, 1999.

[47] A. Crevoisier and G. Kellum, "Transforming ordinary surfaces into multi-touch controllers," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 113–6, 2008.

[48] M. Montag, S. Sullivan, S. Dickey, and C. Leider, "A low-cost and low-latency multi-touch table with haptic feedback for musical applications.," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 8–13, 2011.

[49] A. D. Wilson, "Touchlight: an imaging touch screen and display for gesture-based interaction," in *Proceedings of the 6th international conference on Multimodal Interfaces*, pp. 69–76, ACM, 2004.

[50] C. Harrison, H. Benko, and A. D. Wilson, "Omnitouch: wearable multitouch interaction everywhere," in *Proceedings of the 24th annual ACM symposium on User Interface Software and Technology*, pp. 441–50, ACM, 2011.

[51] J. Letessier and F. Brard, "Visual tracking of bare fingers for interactive surfaces," in *Proceedings of the 17th annual ACM symposium on User Interface Software and Technology*, pp. 119–22, ACM, 2004.

[52] R. Jones, P. Driessen, A. Schloss, and G. Tzanetakis, "A force-sensitive surface for intimate control," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2009.

[53] J. Rekimoto, "Smartskin: an infrastructure for freehand manipulation on interactive surfaces," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems: changing our world, changing ourselves*, pp. 113–20, ACM, 2002.

[54] T. S. Saponas, C. Harrison, and H. Benko, "Pockettouch: through-fabric capacitive touch input," in *Proceedings of the 24th annual ACM symposium on User Interface Software and Technology*, pp. 303–8, ACM, 2011.

[55] A. Freed, "Novel and forgotten current-steering techniques for resistive multitouch, duotouch, and polytouch position sensing with pressure," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2009.

[56] R. Jones, "Mtc express multi-touch controller," *Computer Music Journal*, vol. 25, no. 1, pp. 97–9, 2001.

[57] P.-A. Albinsson and S. Zhai, “High precision touch screen interaction,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 105–12, ACM, 2003.

[58] A. M. Hlady, “A touch sensitive x-y position encoder for computer input,” in *Proceedings of the November 18-20, 1969, fall joint computer conference*, pp. 545–51, ACM, 1969.

[59] C. Harrison, J. Schwarz, and S. E. Hudson, “Tapsense: enhancing finger interaction on touch surfaces,” in *Proceedings of the 24th annual ACM symposium on User Interface Software and Technology*, pp. 627–36, ACM, 2011.

[60] 3M, “Technology profile - dispersive signal touch technology,” 2008.

[61] S. Izadi, S. Hodges, S. Taylor, D. Rosenfeld, N. Villar, A. Butler, and J. Westhues, “Going beyond the display: a surface technology with an electronically switchable diffuser,” in *Proceedings of the 21st annual ACM symposium on User Interface Software and Technology*, pp. 269–78, ACM, 2008.

[62] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza, “Tuio: a protocol for table-top tangible user interfaces,” in *Proceedings of the 6th Internationall Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005.

[63] A. Gokcezade, J. Leitner, and M. Haller, “Lighttracker: an open source multi-touch toolkit,” in *Proceedings of the ACM International Conference on Advances in Computer Entertainment Technology*, 2010.

[64] G. Kellum and A. Crevoisier, “A flexible mapping editor for multi-touch musical instruments,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2009.

[65] R. L. Potter, L. J. Weldon, and B. Shneiderman, “Improving the accuracy of touch screens: an experimental evaluation of three strategies,” in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 27–32, ACM, 1988.

[66] A. Sears and B. Shneiderman, “High precision touchscreens: design strategies and comparisons with a mouse,” *International Journal of Man-Machine Studies*, vol. 34, no. 4, pp. 593–613, 1991.

[67] L. Hoste, “Software engineering abstractions for the multi-touch revolution,” in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, vol. 2, pp. 509–10, ACM, 2010.

[68] D. Rubine, “Specifying gestures by example,” *SIGGRAPH Computer Graphics*, vol. 25, no. 4, pp. 329–37, 1991.

[69] T. Moscovich and J. F. Hughes, "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*, pp. 1275–83, ACM, 2008.

[70] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," *Commun. ACM*, vol. 54, no. 2, pp. 60–71, 2011.

[71] G. Vigliensoni and M. M. Wanderley, "Soundcatcher: explorations in audio-looping and time-freezing using an open-air gestural controller," in *Proceedings of the International Computer Music Conference*, 2010.

[72] R. J. K. Jacob, L. E. Sibert, D. C. McFarlane, and J. M. Preston Mullen, "Integrality and separability of input devices," *ACM Transactions on Computer-Human Interaction*, vol. 1, no. 1, pp. 3–26, 1994.

[73] G. Kurtenbach and W. Buxton, "User learning and performance with marking menus," in *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, pp. 258–64, ACM, 1994.

[74] G. Geiger, "Using the touch screen as a controller for portable computer music instruments," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 61–4, 2006.

[75] N. Liebman, M. Nagara, J. Spiewla, and E. Zolkosky, "Cuebert: a new mixing board concept for musical theatre," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 51–6, 2010.

[76] A. Esenther and K. Ryall, "Fluid dtmouse: better mouse support for touch-based interactions," in *Proceedings of the working conference on Advanced Visual Interfaces*, pp. 112–5, ACM, 2006.

[77] G. Ramos and R. Balakrishnan, "Zliding: fluid zooming and sliding for high precision parameter manipulation," in *Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, pp. 143–52, ACM, 2005.

[78] C. Forlines and C. Shen, "Dtlens: multi-user tabletop spatial data exploration," in *Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, pp. 119–22, ACM, 2005.

[79] A. Olwal and S. Feinter, "Rubbing the fisheye: precise touchscreen interaction with gestures and fisheye views," in *Conference Supplement of the ACM Symposium on User Interface Software and Technology*, pp. 83–4, 2003.

[80] S. R. Ness and G. Tzanetakis, “Audioscapes: exploring surface interfaces for music exploration,” in *Proceedings of the International Computer Music Conference*, pp. 303–6, 2009.

[81] M. Torrens, P. Hertzog, and J.-L. Arcos, “Visualizing and exploring personal music libraries,” in *Proceedings of the 5th International Conference on Music Information Retrieval*, pp. 421–4, 2004.

[82] S. Hitchner, J. Murdoch, and G. Tzanetakis, “Music browsing using a tabletop display,” in *Proceedings of the 8th International Conference on Music Information Retrieval*, 2007.

[83] J. Hochenbaum, O. Vallis, D. Diakopoulos, J. Murphy, and A. Kapur, “Designing expressive musical interfaces for tabletop surfaces,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 315–8, 2010.

[84] R. Laney, C. Dobbyn, A. Xamb, M. Schirosa, D. Miell, K. Littleton, and N. Dalton, “Issues and techniques for collaborative music making on multi-touch surfaces,” in *Proceedings of the 7th Sound and Music Computing Conference*, (Barcelona, Spain), 2010.

[85] G. Levin, “The table is the score: an augmented-reality interface for real-time, tangible, spectrographic performance,” in *Proceedings of the International Computer Music Conference*, 2006.

[86] G. Partridge, P. Irani, and G. Fitzell, “Let loose with wallballs, a collaborative tabletop instrument for tomorrow,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 78–81, 2009.

[87] L. O’Sullivan and F. Boland, “Tailoring tabletop interfaces for musical control,” in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pp. 331–4, ACM, 2010.

[88] P. L. Davidson and J. Y. Han, “Synthesis and control on large scale multi-touch sensing displays,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 216–9, 2006.

[89] M. Cicconet, I. Paterman, L. Velho, and P. Carvalho, “On multi-touch interfaces for music improvisation: the blues machine project,” tech. rep., Vision and Graphics Laboratory, National Institute of Pure and Applied Mathematic, 2010.

[90] T. Rhea, “The sackbut blues: Hugh le caine: Pioneer in electronic music by gayle young,” *Computer Music Journal*, vol. 19, no. 4, pp. 96–9, 1995.

[91] L. Fyfe, S. Lynch, C. Hull, and S. Carpendale, "Surfacemusic: mapping virtual touch-based instruments to physical models," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 360–3, 2010.

[92] Y. Kuhara and D. Kobayashi, "Kinetic particles synthesizer using multi-touch screen interface of mobile devices," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 136–7, 2011.

[93] G. Roma and A. Xambó, "A tabletop waveform editor for live performance," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 249–52, 2008.

[94] G. Lakoff, *Contemporary Theory of Metaphor*, ch. 11, pp. 202–51. Cambridge University Press, second ed., 1994.

[95] G. Lakoff and M. Johnson, *Metaphors we live by.* Chicago: University of Chicago Press, 1980.

[96] L. M. Zbikowski, "Cross-domain mapping.," *Conceptualizing Music*, vol. 1, pp. 63–96, 2002.

[97] P. Barr, R. Khaled, J. Noble, and R. Biddle, "A taxonomic analysis of user-interface metaphors in the microsoft office project gallery," in *Proceedings of the Sixth Australasian conference on User interface - Volume 40*, pp. 109–17, Australian Computer Society, Inc., 2005.

[98] G. Fauconnier and M. Turner, *The way we think: conceptual blending and the mind's hidden complexities.* Basic Books, 2003.

[99] J. L. Alty, R. P. Knott, B. Anderson, and M. Smyth, "A framework for engineering metaphor at the user interface," *Interacting with Computers*, vol. 13, no. 2, pp. 301–22, 2000.

[100] M. Imaz and D. Benyon, *Designing with blends: conceptual foundations of human-computer interaction and software engineering.* MIT Press, 2007.

[101] D. Saffer, "The role of metaphor in interaction design," Master's Thesis, The School of Design, Carnegie Mellon University, Pittsburgh, PA, USA, 2005.

[102] A. Gadd and S. Fels, "Metamuse: metaphors for expressive instruments," in *Proceedings of the International Conference on New interfaces for Musical Expression*, (1085206), pp. 1–6, National University of Singapore, 2002.

[103] S. Fels, A. Gadd, and A. Mulder, "Mapping transparency through metaphor: towards more expressive musical instruments," *Organised Sound*, vol. 7, no. 2, pp. 109–26, 2002.

[104] M. Duignan, J. Noble, P. Barr, and R. Biddle, "Metaphors for electronic music production in reason and live," in *Proceedings of the 6th Asian Pacific Conference on Computer Human Interaction* (M. Masoodian, S. Jones, and B. Rogers, eds.), vol. 3101 of *Lecture Notes in Computer Science*, pp. 111–20, Springer Berlin / Heidelberg, 2004.

[105] O. Bau, A. Tanaka, and W. E. Mackay, "The a20: musical metaphors for interface design," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 91–6, 2008.

[106] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 1–4, National University of Singapore, 2001.

[107] L. Dahl and G. Wang, "Sound bounce: physical metaphors in designing mobile music performance," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 178–81, 2010.

[108] B. Wöldecke, C. Geiger, H. Reckter, and F. Schulz, "Antracks 2.0 - generative music on multiple multitouch devices," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 348–51, 2010.

[109] I. Stavness, J. Gluck, L. Vilhan, and S. Fels, "The musictable: a map-based ubiquitous system for social interaction with a digital music collection," in *Proceedings of the International Conference on Entertainment Computing*, vol. 3711 of *Lecture Notes in Computer Science*, pp. 291–302, Springer Berlin / Heidelberg, 2005.

[110] A. Bragdon, A. Uguray, D. Wigdor, S. Anagnostopoulos, R. Zeleznik, and R. Feman, "Gesture play: motivating online gesture learning with fun, positive reinforcement and physical metaphors," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces 2010*, pp. 39–48, ACM, 2010.

[111] S. Malik, A. Ranjan, and R. Balakrishnan, "Interacting with large displays from a distance with vision-tracked multi-finger gestural input," in *Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, pp. 43–52, ACM, 2005.

[112] R. Rowe, *Interactive Music Systems: Machine Listening and Composing.* MIT Press, 1993.

[113] J. Drummond, "Understanding interactive systems," *Organised Sound*, vol. 14, no. 2, pp. 124–33, 2009.

[114] B. Bongers, *Physical Interfaces in the Electronic Arts Interaction Theory and Interfacing Techniques for Real-time Performance.* IRCAM, 2000.

[115] M. M. Wanderley and N. Orio, "Evaluation of input devices for musical expression: borrowing tools from hci," *Computer Music Journal*, vol. 26, no. 3, pp. 62–76, 2002.

[116] M. Wu and R. Balakrishnan, "Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays," 2003.

[117] D. Wessel, M. Wright, and J. Schott, "Intimate musical control of computers with a variety of controllers and gesture mapping metaphors," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 171–3, 2002.

[118] L. A. Wozny, "The application of metaphor, analogy, and conceptual models in computer systems," *Interacting with Computers*, vol. 1, no. 3, pp. 273–83, 1989.

[119] M. Zadel and G. Scavone, "Different strokes: a prototype software system for laptop performance and improvisation," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 168–71, 2006.