OrchestraTable: A Multi-touch Music Interface for the Surface Touch Table

Hanna Sanford

University of St Andrews

Submitted: Friday, 4 April 2014

ABSTRACT

Music and technology are ubiquitous in our culture. They have been intertwined through time to bring us to a point where any kind of music is available to us instantly. Technology has also become involved in the creation of music. Instruments are becoming increasingly high tech and detached from the concept of traditional music making devices. As new technologies emerge, they too are being used in new ways to create and record music. Experimentation is an important part of this process which ensures the new technology is able to fulfil its whole potential. Although touch devices progressed to become a quotidian feature of modern life, multi-touch technology is still developing. Thus it is beneficial to investigate a variety of divergent applications. This paper explores one of those applications by designing and implementing a multi-touch, table based musical instrument, then assesses it by comparing it to existing related work.

DECLARATION

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 7,519 words long, including project specification and plan.

Hanna Sanford

CONTENTS

INTRODUCTION

Music is a topic that fascinates us as a species. It can be used to bring communities together, trigger emotions, and tell stories. Because of its importance to us today, there is much research and debate on when the first music was created. Upon discovery of bone flutes found in the Hohle Fels cavern, south-west Germany, dating back as far as 40,000 years ago, a researcher, Professor Nicholas Conard of Tubingen University, said "It's becoming increasingly clear that music was part of day-to-day life"[17]. Some researchers have even gone so far as to argue that music originated in the time of the Neanderthals[16] or is an evolutionary adaptation[18]. However old it is, music is ubiquitous in our culture. It is referred to in some of the oldest roots of our society. One of the ancient epics, the *Aeneid*, begins "Arma virumque cano"[19] which translates to I sing of arms and of a man. Instruments have evolved along side out culture becoming the complex digital sound generation tools we have today.

When modern, electrical technology came into existence, the nature of technology's involvement in music changed. It was no longer just used in the creation of music but in the preservation and distribution of music. In 1877, the first recording of a human voice was made by Thomas Edison on the first phonograph[9]. From then, the obsession with new methods of recording and playing technology has continued from the invention of the jukebox in the 1890s, to the recording industry boom in the 1900s, to the invention of FM radio in 1933, and later the MP3 in 1990[9]. Parallel with these inventions, musical instruments have been evolving as well. What once was completely analog was being brought into the digital age— a guitar became an electric guitar and then a synthesiser imitating a guitar. Musical instruments have changed so much that some types of music no longer try to imitate analog instruments. In much pop music, there are few if any traditional instruments and any sound made is digitally generated by computers.

In this new age, the definition of an instrument is growing and changing. Where an instrument used to be a physical device with a single dedicated purpose, it now may just be one setting of a greater system. This system may also be definable as an instrument as according to the Merriam-Webster Dictionary, an instrument is simply a device that is used to make music[12]. In this paper, the system itself is described as an instrument because it is a device that can be used to produce music, but the different sound-texture options provided by the MIDI *instruments* are also referred to as instruments due to the MIDI terminology.

Each new sound or interface technology is examined with the potential of being used to create new ways of making music. Even seemingly unlikely technologies, are being examined as musical instruments such as First-Person Shooters[10]. In this way multi-touch technology seems to be an obvious choice for use in instrument development because the act of playing instruments has always been tactile. Multi-touch technology allows the user to control multiple things simultaneously, whether it is playing multiple notes, controlling different properties of the sound, or as in the case of this project, controlling multiple instruments.

While other multi-touch music systems exist, the possibilities have not been fully explored and therefore more systems must be created to explore new possibilities and configurations in order to optimise the use of multi-touch technology.

OBJECTIVES

The specification provided in this project was intentionally very broad and vague to encourage creativity. It said only that the project was to create an interface for music composition and interpretation. Therefore, as part of the design stage, I created design goals from which the objectives were elicited. These objectives were designed to allow the project to evolve while still ensuring the completion of the core of the project. Primary objectives were the barebones stage of the project, the secondary and tertiary objectives were optional extensions to the project that would be achieved if the project moved in that direction. For this reason, not all of the secondary and tertiary objectives were achieved.

Primary

1. Build a multi-touch interface which generates sound

2. Implement more fluid method of generating sound

3. Develop interface into an instrument

Secondary

4. Add percussion

5. Add additional instruments

6. Allow multiple instruments to be played simultaneously

7. Create method of generating sheet music or files from previous experimentation

8. Enhance sound quality and tone

Tertiary

9. Make sheet music modifiable live

10. Allow saving of states

11. Add chorus, verse, or bridge section markers

12. Form a band

Achievements

This project created a multi-touch application for the Samsung Sur40 table running Microsoft Surface. It consists of one or more control rings which play notes when a sector of the ring is touched and a staff which displays the notes as they are played (Objective 1). The control ring accepts multiple touch inputs at once allowing the user to play chords. The ring is designed to be controlled with one hand allowing the other hand to control a second instrument ring (Objective 6). The application can be used to create music; therefore, by the already stated definition, it is an instrument (Objective 3). The Midi library used in this application, Toub.Sound.Midi, has fairly realistic sounding instruments which meant there was little need to improve the sound quality or generate sound separately (Objectives 2 and 8). The different rings can be configured as different instruments and played simultaneously allowing the user or users to create a complete arrangement (Objective 5). As the user plays music on the application, visual representations of the notes are generated on a staff-like frame. The notes on the visual staff are represented by dots matching the colour of the note sector on the control ring and are placed on a line delineating that note. The time scale between notes is shown by the horizontal spacing between the dots. As more dots appear, the staff scrolls to show the most recent notes but allows the user to scroll back to previous activity.

As shown above, this project fulfilled all of the primary objectives set for it and three of the secondary objectives. The tertiary objectives mostly deal with post-play modification of the music made on the application. These were not achieved because the project moved more in the direction of facilitating live play rather than composition.

RELATED WORK

Since the invention and popularisation of multi-touch interfaces, people have been developing creative ways to generate music with them. Most of these applications have been for experimental purposes, exploring new ways to think about the concept of an instrument and researching how multi-touch technology could play a part. Many research projects have been developed to allow those who lack classical music training to easily create enjoyable music. Some like ToCoPlay[6] and MelodyMorph[7] allow simple compositions to be generated easily though their straightforward designs but prevent the user from being able to extend to live or multi-instrument performance. While they allow the user to interact with specific notes and so are able to play rudimentary pieces of music, they are limited as they cannot support too much complexity. Spaces[1] and WallBalls[8] both generate sound through creative methods that are entertaining to interact with and satisfying to even the least skilled user. However, it would not be able to be used for structured music composition as it is difficult to control specific note choice and timing.

Some projects such as Vuzik[2] take a more artistic approach which allows the user to generate sound through an interaction process similar to painting. This enables it to work as an instrument while presenting an elegant, artistic interaction method. Others have removed the goal of musical composition altogether. These projects are designed more like art installation with the aim to inspire thought and reflection rather than being a useful tool. One such project is Roots[1], which generates roots that grow from the touch point creating

different cycles of sound. Another, Ah!Text[1], goes through a few different programs using voices reading text to generate music.

Another common direction for multi-touch music interfaces is sound mixing. With digital, synthesised music increasing in popularity, music is not longer created just from physical instruments but from complex digital mixing software. Reactable[3] and Xanakis[9] use tokens on a touch table to generate different kinds of sound and beats which are mixed together to create a piece of music similar to that of club DJs.

Not all research projects have made music creation interfaces. Others like Isochords[4] attempted to use geometry to aid in the classification of musical structure. The mathematics of music has long been a tempting field for scientist as they seek to explain what about music causes it to have such an effect on us. Isochords[4] creates an artistic visualisation through an algorithmic representation of sound.

In this project, users can create or play music but not preserve it for playback. The design is created mostly for functionality rather than aesthetics aligning it primarily with the play and experimentation category, with ToCoPlay[6] and MelodyMorph[7]. However, it tries to better imitate traditional instrument sounds enabling it to be an interface potentially usable in a performing atmosphere.

METHODOLOGY

Iterative Design And Engineering Process

The methodology used in this project was a combination of agile techniques and Design-Driven Development. The majority of the design was done at the beginning of the process with an overarching design plan. This started as broad idea sketches to breed idea creation. These sketches were than critiqued to narrow down the options to one final design. This design was then further developed and critiqued until a final complete design arose with UI elements and interaction techniques defined. This final design was used to build a paper prototype which could be manipulated to demonstrate the interaction techniques designed. I then brought this prototype to a meeting with my supervisor and we conducted a design critique in order to find any issues with the design before moving into the implementation phase. This process is described in more detail in the Design Stage section. Each week my supervisor and I would meet to perform a design critique and check on my progress. In the design critique, we would re-examine the initial design plan and compare it to what had been implemented, taking into consideration any problems that were encountered in the implementation and redesigning any aspects that proved themselves to be impractical or unsuitable for the program at that stage. We then set goals for what to achieve in the next week.

ETHICS

This project has no ethical implications or considerations as it does not involve any people or people's data.

DESIGN STAGE

Initially the specification for this project was left intentionally vague, described only as a multi-touch music interface. This left the design process very open for exploration across different platforms, different languages, and different designs. The first thing to be decided was what was trying to be achieved by this project and what angle would this project take. In order to decide this I had to review the current related work. After reviewing the papers described above, goals had to be set for the design of this project.

Design Goals

The main design goal for this project was to make a music interface that was friendly and usable to those with classical music training while maintaining the usability for those with no formal musical background. Other goals included enabling more options when creating a piece of music such as instrument variation and having the notes laid out in a way that communicated information about their relation to each other.

Design Process

From the design goals, I knew I was trying to create an interface based on classical music but with an accessible concept that would enable novice users to jump right in. This

inspired a number of initial ideas and sketches exploring different directions that will be described below. I allowed a portion of time to just investigate the options freely without judgement before starting to narrow down the direction towards one design. I then looked at each design idea and decided which aspects were strong and which were weak. This then led to another iteration of sketches which were more directed towards one solution (see Figure 1). These sketches described different aspects of the interface and how interactions would be performed. From these sketches, I built a paper prototype that would allow basic interaction to be demonstrated in a rudimentary way (see Figure 3). The prototype includes the table design with the staff and separate pieces for the control rings so they could be moved and rotated around the screen. There are also note pieces that can be placed on the staff and moved to demonstrate what happens when a sector is touched. Once the prototype was made, I moved into the implementation stage but the design continued to evolve based on limitations in the platform or feedback from testing of the application. For example, the staff had to be redesigned part way through because space limitations meant the original design would have appeared crowded. These changes were sketched out and critiqued in the weekly meetings with my supervisor (see Figure 2).

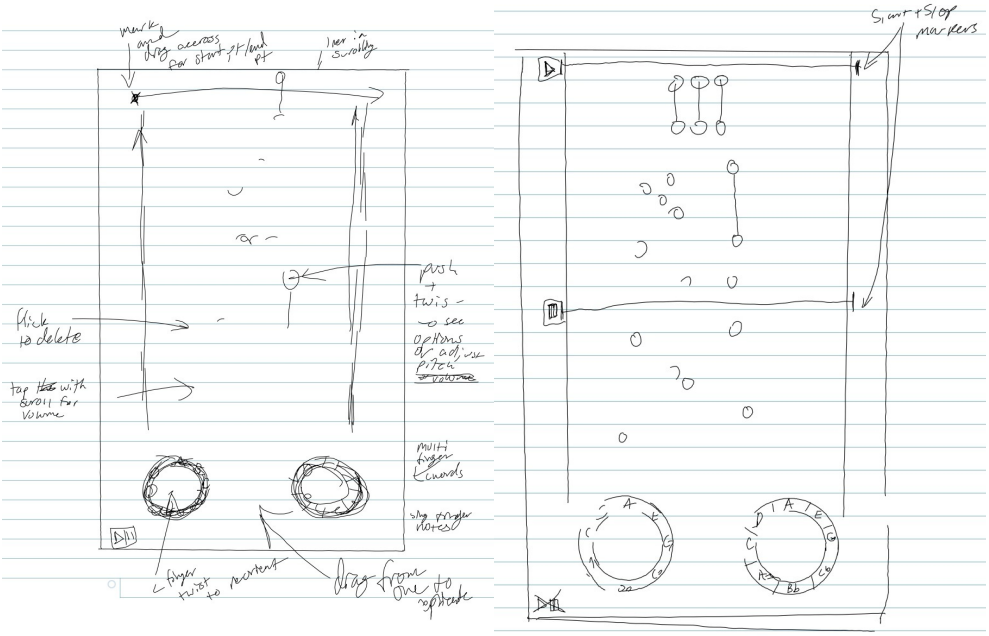


FIGURE 1: SKETCHES OF FINAL DESIGN IDEA

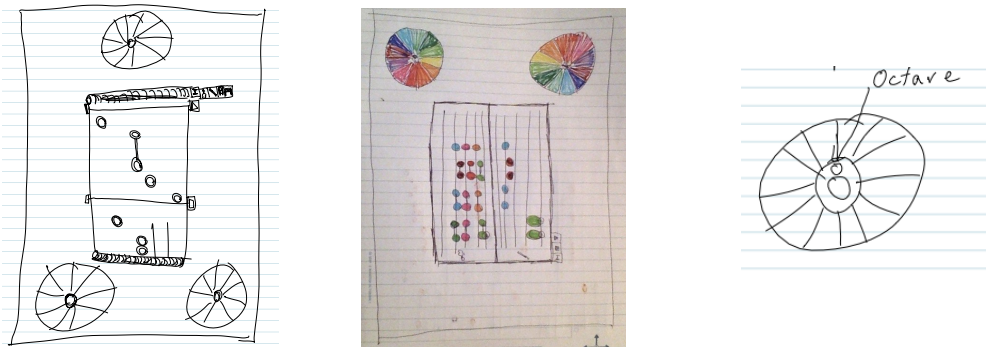


FIGURE 2: SKETCHES OF REDESIGNS DURING IMPLEMENTATION STAGE

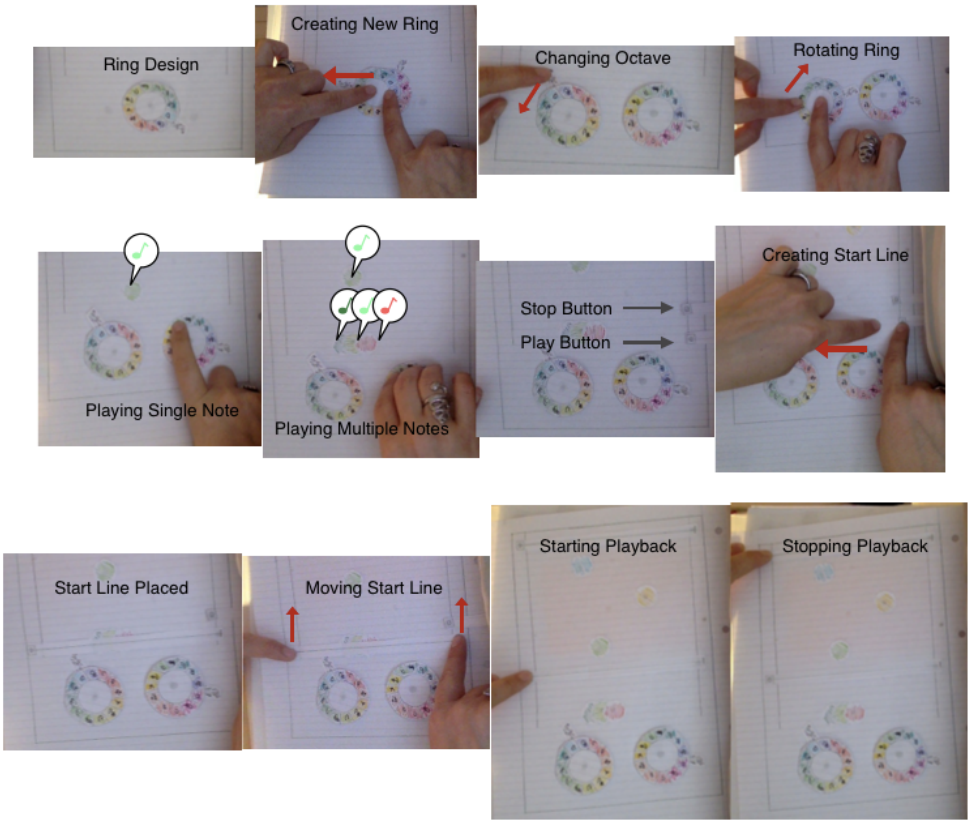


FIGURE 3: PAPER PROTOTYPE SHOWING INTERACTION METHODS

Design Alternatives

First Design Idea

One of the alternative design ideas from the initial design stage involved a classical music staff across the top with notes written as they are played. This was there to enable those with classical training to easily understand what was happening on the table and allow the music created on the table to be transferred to other instruments. Along the left side panel, it included a list of instruments in image form which can be scrolled trough by touching and dragging a bracket along the list. This configuration was both easily accessible in real time and displayed the current state in a quick and simple way to the user. In the centre section, the content would change to mimic that instrument depending on which is selected. For example, when the drum kit was selected it would display a drum kit layout which would make noise upon touch interaction with each drum (see Figure 4), and when the piano selected a simple touch keyboard would be presented (see Figure 5). This design would not allow a user to play multiple instruments at once and is more of a composition tool than a live instrument. Because of this, this design lacked the flexibility desired in this project.

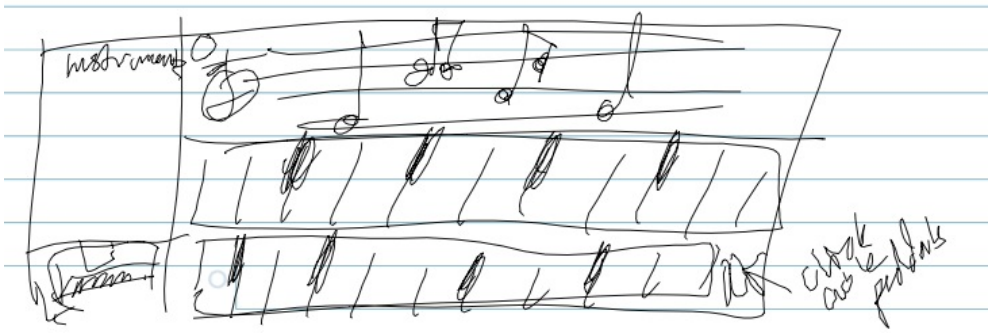


FIGURE 4: SKETCH OF FIRST IDEA IN PIANO VIEW

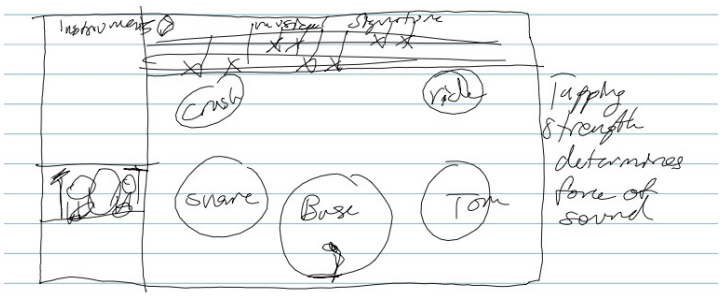


FIGURE 5: SKETCH OF FIRST IDEA IN DRUM VIEW

Second Design Idea

A second initial design idea involved dots that represent notes and could be dragged around the screen, tapped to be played, or linked to other notes (see Figure 6) to enable them to be played at once or in sequence. The dots would be located in a scrolling box at the bottom. This would prevent live play from being possible, but would be most accessible to the user and allow for easy manipulation of more complex attributes such as time spacing and attributes of the notes. In this design there were the same top and side panels as in the first, but there was also a separate screen for editing the music that appears on the staff above which provided tools to change the notes on the staff, insert rests, or change the type of notes (see Figure 7). This is accessed by tapping that staff. This page would enable more precise editing of a piece. Any changes made in the playing side would effect the staff and vice versa. This could be expanded to include markers designating which section of a piece of music is the chorus, verse, or bridge and have markers which could be attached to the notes to create certain types of effects (see Figure 8). This design also did not provide the flexibility desired but fulfilled goals in a different direction. Although this design was not selected for this project, another project, possibly on a tablet platform, could accomplish different goals using a similar design to this.

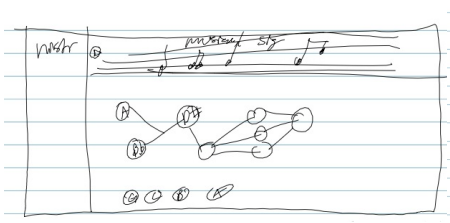


FIGURE 6: SKETCH OF SECOND IDEA IN PLAY MODE

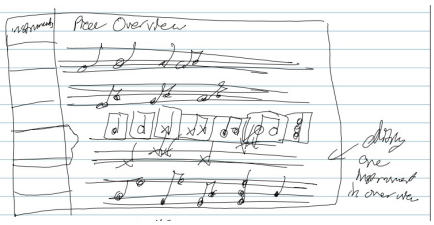


FIGURE 7: SKETCH OF SECOND IDEA IN EDIT MODE

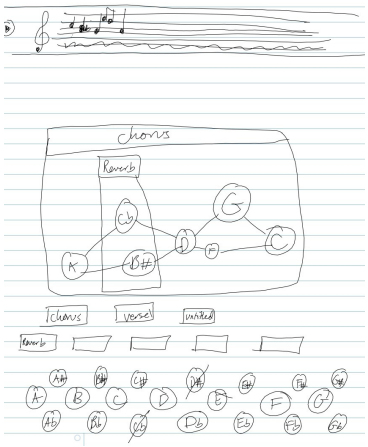


FIGURE 8: SKETCH OF SECOND IDEA WITH SECTION MARKER EXTENSION

Third Design Idea

A third, and most developed, alternative design idea had most of the table taken up by a scrolling panel where the notes could be dragged off the storage ring below the panel (see Figure 11). This arrangement provides some information about the notes that make good chords; a circle allows notes to be positioned so that major and minor chords form a triangle and other chords form other shapes (see Figure 10). Next to the the ring are a few slider controls to control volume or the octave of the notes on the ring. This provides very precise modification of the notes in real time but also prevents live play. Once a note is placed, different factors can be controlled through different gestures such as touching a note to play it (see Figure 12), touching and dragging to draw a line between notes which will play a slide, double tapping or another gesture to change properties of the note. There are also lines that can be placed at different points of the panel to mark where the playback will start and end (see Figure 9). When the playback is started by clicking on the button at the end of the start line, the notes would light up as the play line crosses over them and they are played (see Figure 13). To pause the playback the user would click the button at the end of the end line. This design was most similar to the one this project is based on though the final design was evolved to allow the application to be used more like a physical instrument requiring less set-up time. This design shared a bit too much with that of ToCoPlay[5], so it was not chosen for this project.

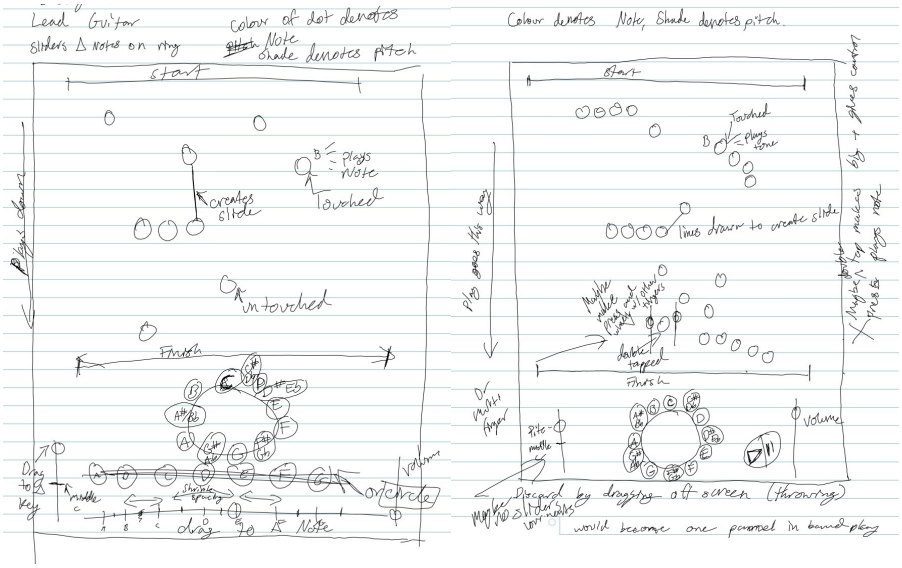


FIGURE 9: SKETCHES OF THIRD IDEA ANNOTATED WITH INTERACTION INFORMATION

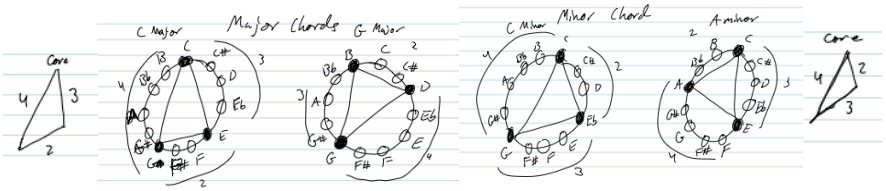


FIGURE 10: SKETCHES SHOWING THE SHAPES THAT MAJOR AND MINOR CHORDS MAKE

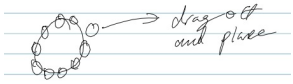


FIGURE 11: SKETCH SHOWING NOTE CREATION METHOD

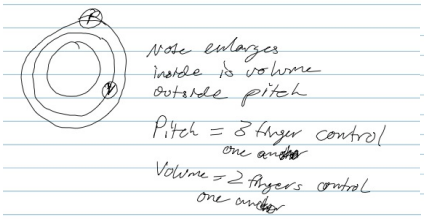


FIGURE 12: SKETCH SHOWING NOTE ATTRIBUTE SETTINGS

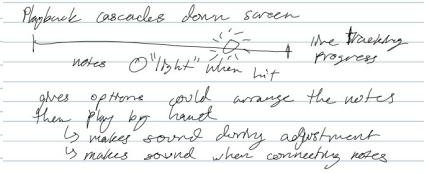


FIGURE 13: SKETCH SHOWING THAT NOTE FLASHES WHEN PLAYED

The Final Design

Design

The final design consists of a circular "control ring" (see Figure 14) which is divided into twelve sectors with a small circle in the centre and act as the main control point for the user. Each sector represents a note on the classical scale within an octave starting at middle C (see Figure 15). The colour of the centre circle informs the user which of the provided instruments is being represented by that ring. In the centre of the table is a panel, called the visual staff, which shows notes that have already been played (see Figure 16). Notes appear as a small dot on the visual staff. The notes scroll across the screen within the centre panel as more notes are played or when the user manipulates the scrollbar at the top of the panel allowing the previously played notes to be viewed. At the bottom of the visual staff, there is a square button that opens a menu showing the different instruments possible to choose a ring to represent.

FIGURE 14: THE CONTROL RING

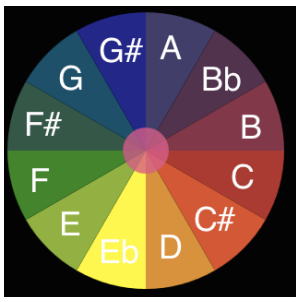


FIGURE 15: THE CONTROL RING LABELED WITH NOTES

FIGURE 16: THE VISUAL STAFF

Control Ring

The control ring can be moved, scaled, or rotated. Movement of the ring is controlled using a touch on the centre circle and a drag to the new location (see Figure 17). Rotation and scaling are controlled with two fingers (see Figures 18 and 19). One finger touches and remains on the centre circle while the other pinches towards, spreads away from, or rotates around the centre. To play notes on the ring, the chosen sector must simply be touched (see Figure 20). A note will play as long as a finger is touching it and will only end when the finger is lifted or moved off the sector. Touching multiple sectors will cause multiple notes to play at once (see Figure 21). When a note is played on the ring a circle will appear in the centre panel.

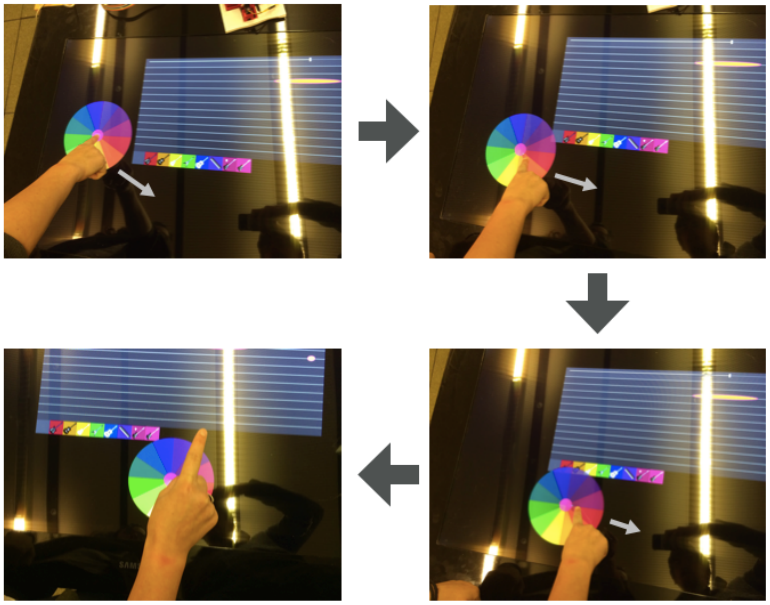


FIGURE 17: SEQUENCE DIAGRAM OF MOVING THE CONTROL RING

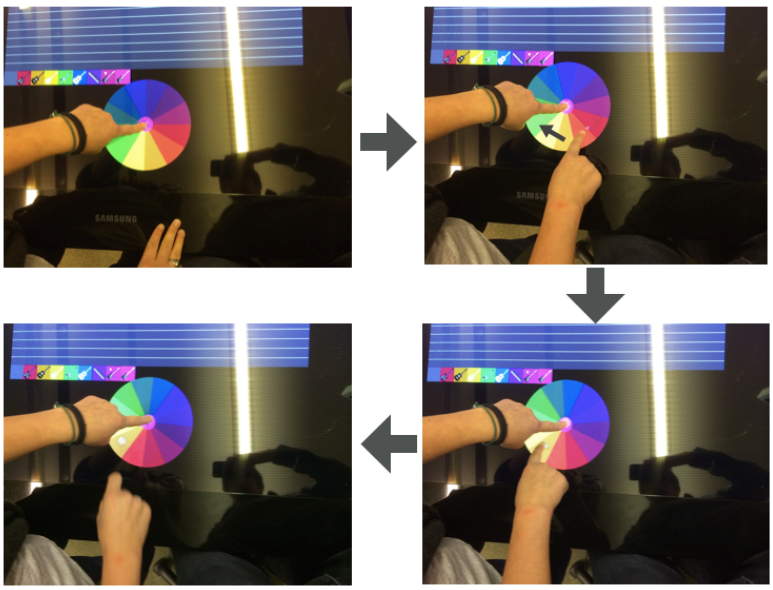


FIGURE 18: SEQUENCE DIAGRAM OF ROTATING THE CONTROL RING

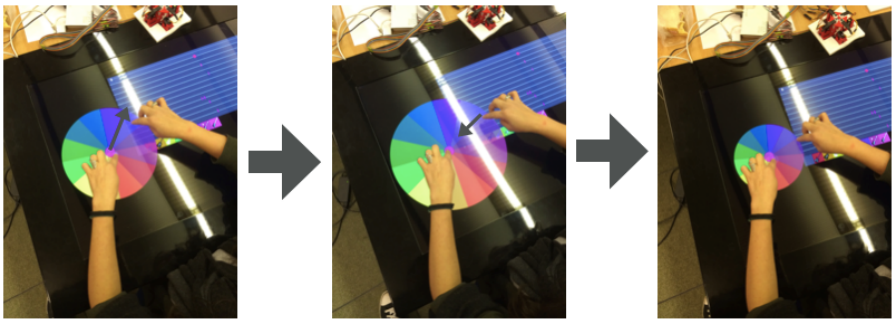


FIGURE 19: SEQUENCE DIAGRAM OF SCALING THE CONTROL RING

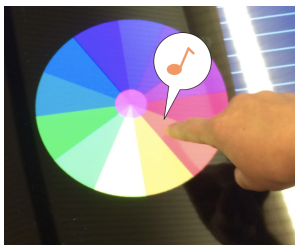


FIGURE 20: DIAGRAM OF A NOTE BEING PLAYED

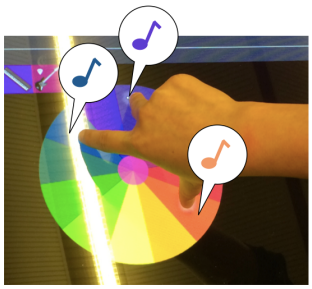


FIGURE 21: DIAGRAM OF MULTIPLE NOTES BEING PLAYED

Note Visualisation

Visual Staff

The central panel is referred to as the visual staff (see Figure 16). It holds the visual representation of the notes that have been played. It consists of a main panel with twelve horizontal lines, one for each note, a scrollbar at the top, and a square button at the bottom. As notes are played they appear as coloured dots on the line representing the note that was played. The outside colour of the dot tells the user which instrument the note was sent from and the inside colour tells which note the dot represents (see Figure 22). As more notes are played the visual staff scrolls automatically to show the current progress but the scrollbar at the top can also be touched and dragged back or forward to look at other sections of the visual staff (see Figure 23). The button at the bottom opens a menu of instruments available to the user when touched. Touching one of the instrument buttons will create a ring of the matching colour (see Figure 24). To delete a ring the user must drag the centre circle of the ring over the matching button (see Figure 25). The ring will be deleted upon release. The spaces between the note dots on the visual staff represent the time between when the notes were played.

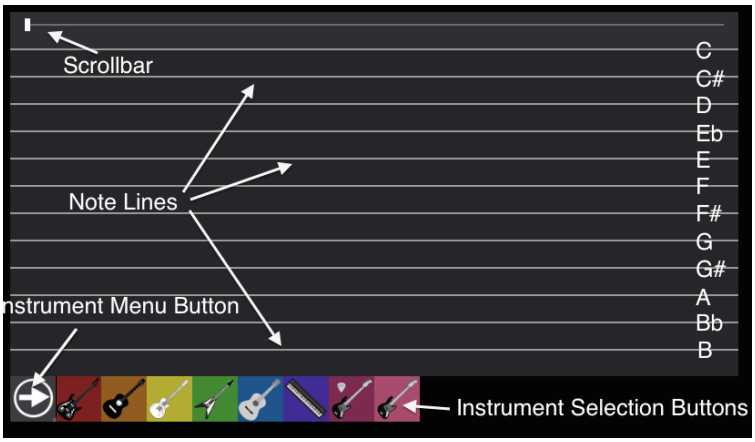


FIGURE 22: LABELED DIAGRAM OF VISUAL STAFF

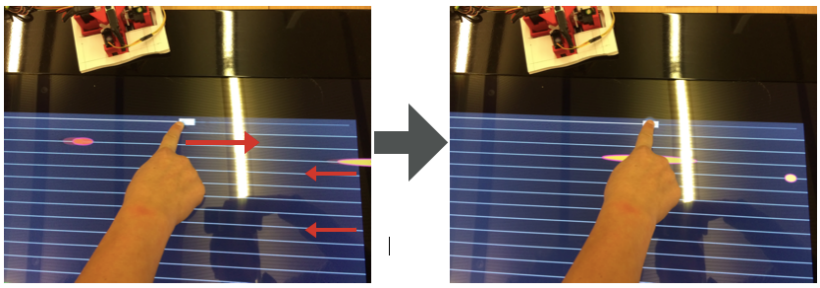


FIGURE 23: SEQUENCE DIAGRAM OF SCROLLING ON THE VISUAL STAFF

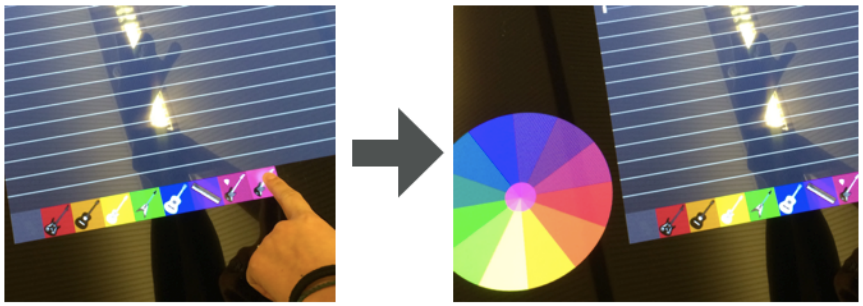


FIGURE 24: SEQUENCE DIAGRAM OF CREATING A NEW INSTRUMENT RING

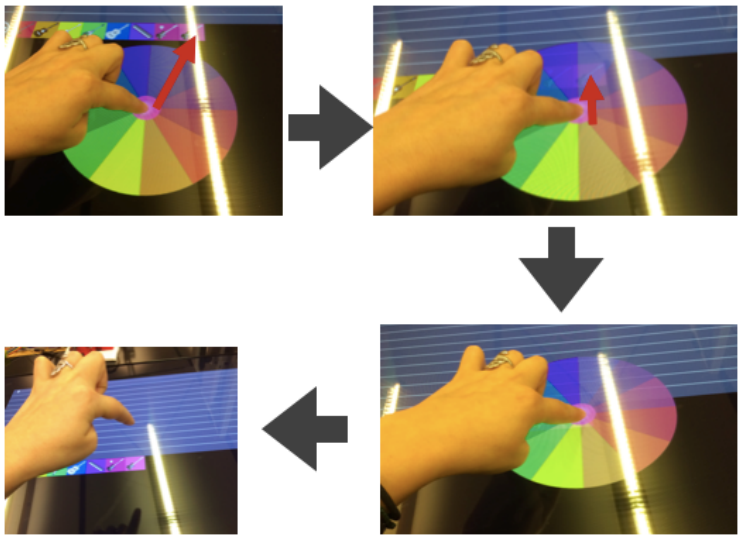


FIGURE 25: SEQUENCE DIAGRAM OF DELETING AN INSTRUMENT RING

Dealing with Multiple Instruments

The multiple instruments are distinguished using colouration, and the centre circle of the control ring for that instrument will match the colour of the button representing it. The notes played by that instrument will also have the matching outside colouration (see Figure 26). Multiple instruments can be played simultaneously as they are represented by different rings which allow multiple simultaneous touches. The rings can be moved and changed

independently from each other and played simultaneously. The multi-touch behaviour allows chords to be played on rings using multiple fingers (see Figure 21). Two rings for the same instrument will have the same colouration and will not be distinguishable from each other (see Figure 27). They are instead treated like multiple people playing one physical instrument.

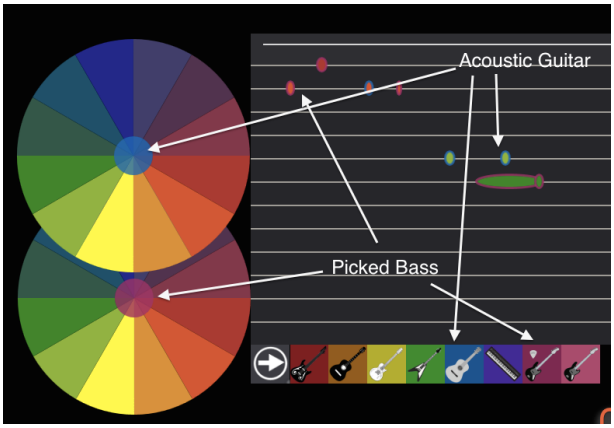


FIGURE 26: LABELED DIAGRAM TO SHOW INSTRUMENT COLOURATION

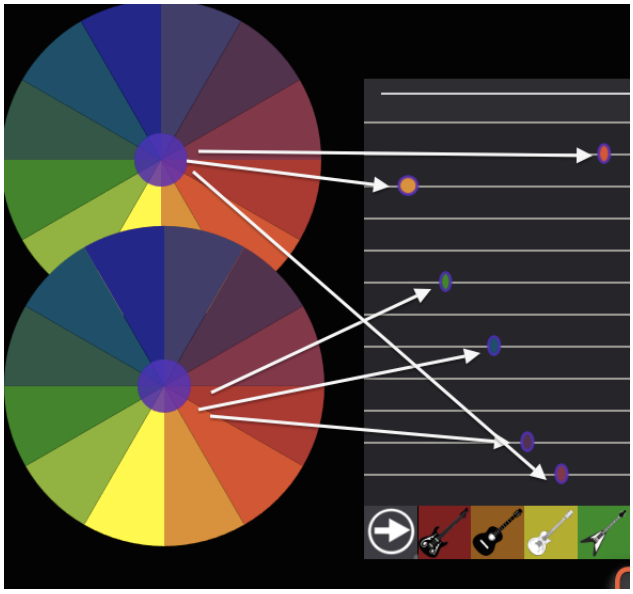


FIGURE 27: LABELED DIAGRAM TO SHOW THAT TWO RINGS FOR THE SAME INSTRUMENT WILL HAVE THE SAME COLOURATION

REQUIREMENTS AND SPECIFICATION

The following requirements were based on the final design and design modifications that arose from the iterative design critiques throughout the process. Some requirements moved away from the original secondary and tertiary objectives.

Interface Requirements

- **Requirement 1:** Interface
  System requirement. Importance: very high
  The software must include an interface for the Sur40 table.

- **Requirement 1.1:** Multi-touch Interface
  System Requirement. Importance: very high
  Interface must support multi-touch interaction.
- **Requirement 1.2:** Informative Interface
  System Requirement. Importance: high
  The interface must provide a method of interaction which facilitates understanding of basic musical relationships.
- **Requirement 1.3:** Easy to Use
  User Requirement. Importance: high
  The user should be able to use the interface without formal training or too steep a learning curve.
- **Requirement 1.4:** Interface Flexibility
  System Requirement. Importance: intermediate
  The interface must be flexible so that it is useful to different types of users with different musical backgrounds.

Sound Requirements

- **Requirement 2:** Sound
  System requirement. Importance: very high
  Application must be able to generate sound.
  - **Requirement 2.1:** Dynamic Sound
    System Requirement. Importance: very high
    Application must be able to generate sound dynamically based on user input.
  - **Requirement 2.2:** Realistic Sound
    System Requirement. Importance: low
    The system must be able to generate realistic sound which effectively imitates instruments.

Interaction Requirements

- **Requirement 3:** Instrument
  System Requirement. Importance: high
  Application will represent or mimic the behaviour of a traditional instrument.
  - **Requirement 3.1:** Instrument Play
    User Requirement. Importance: high
    The user must be able to play notes based on multi-touch interaction.
  - **Requirement 3.2:** Chords
    User Requirement. Importance: high
    User must be able to play multiple notes at once to form chords.
  - **Requirement 3.3:** Multiple Instruments
    System Requirement. Importance: low
    The system must have multiple instrument options.
  - **Requirement 3.4:** Simultaneous Play
    User Requirement. Importance: low
    User or users must be able to play multiple instruments simultaneously.

Feedback Requirements

- **Requirement 4:** Visual Feedback
  User Requirement. Importance: high
  User must be able to see what is being played and what has been played in visual format.
  - **Requirement 4.1:** Easy to Read Representation
    System Requirement. Importance: high
    The system must present visual feedback is an accessible way that is not overcrowded or illegible.
  - **Requirement 4.2:** Classically Inspired Visual Representation
    System Requirement. Importance: intermediate
    Visual representation of notes should imitate classical representations.

Back-End Requirements

- **Requirement 5:** Implementation
  System Requirement. Importance: very high
  Design must be implemented into a usable application.
  - **Requirement 5.1:** Independent Implementation of UI Elements
    System Requirement. Importance: high
    UI elements must be fully encapsulated in order to make them usable in other applications.
  - Requirement 5.2: Simple Implementation
    System Requirement. Importance: high
    System must be implemented in the simplest way possible to reduce the weight of the application of the platform.

IMPLEMENTATION

At the beginning of the implementation stage, decisions had to be made about which platform and language would be used in the project. Two main options were considered. In the first, the UI could be written in Processing, with the back-end written in Java using a Java MIDI library and a Java multi-touch library which would interact with the Windows 7 multi-touch interface and run on the Windows 7 side of the table. In the second, the whole application would be written in Visual Studio, in C#, using a C# MIDI library and the Surface multi-touch commands and run on the Microsoft Surface side of the table. The second option was selected because it was a more self-contained system with fewer external libraries reducing the complexity of the implementation necessary. During the coding process a Mercurial repository was used to maintain version control and preserve previous work for rollback if needed.

Architecture

This application is divided between nine classes (see Figure 28). The class layout is as follows: the main control classes are the *Manager* and *UIWindow;* the MIDI library is controlled from the *Music* class; the UI elements are represented by the *Ring*, *VisualStaff*, and *InstrumentButtons* classes; the colour and image information is stored in the *Colour* class; and the note data is stored in the *Note* and *Staff* classes. The UI element events send information to the back-end using delegates which are set up to call methods in other classes with the given parameters. The *Manager* class holds the back-end of control ring, including the methods which play the notes by calling methods in the *Music* object it holds. The *UIWindow* class links the front- and back-end. It also initiates the Surface window and holds the methods for the configuration and manipulation of the UI elements. The methods stored in the *Manager* and *UIWindow* classes are passed to the UI element classes using delegates. The delegates allow a separation to be obtained between the UI elements and the back-end, enabling multiple exterior methods to be linked to one delegate in the UI class. This allows one UI event to trigger multiple exterior actions. This architecture was designed so the UI elements could be copied as a class and put into another project and used in a completely different way. The back-end is also kept as separate as possible from the rest of the project in order to make the entire structure compartmentalised in a way that different back-end classes could be written and plugged in with only one class, *UIWindow,* having to be modified.

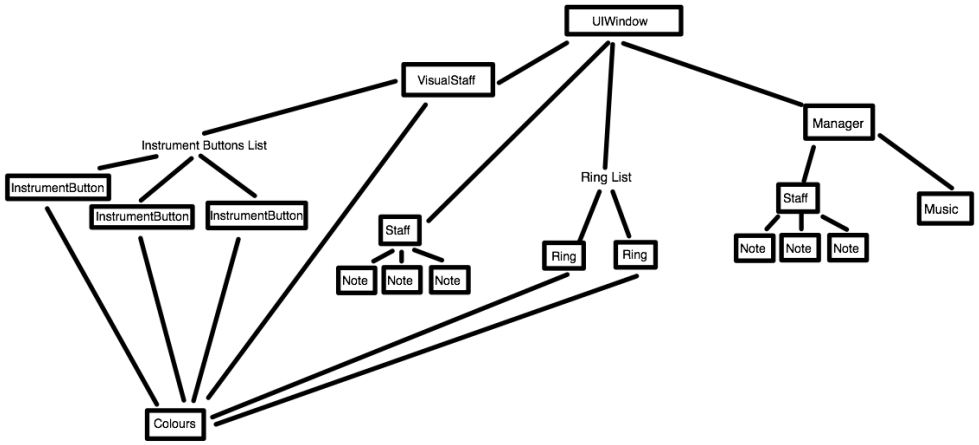


FIGURE 28: DIAGRAM OF THE CLASS STRUCTURE OF THE PROJECT

The Front-End

UI Elements

The front-end of this application is written on the C# side of the Visual Studio C#/WPF file. This made more complex programming possible than the more limiting XAML side. The UI elements are each written as separate Canvas elements with geometric elements making up their structure.

The Ring Class

The control ring has a class, *Ring*, which contains the constructor to sets up the event handlers, *touchDown*, *touchUp*, *touchMove*, for the centre ellipse and the ring itself. In these handlers, delegates, such as *processTouchDown*, call methods in the other classes of the application to carry out actions like playing a note or moving the ring. The different actions are triggered using touch events such as the *touchDown* event which is checked against each of the sectors to determine which was chosen, then calls the *touch_Down_Ring* delegate (see Figure 29). The events are triggered for a specific element, and therefore, when global touch coordinates were needed, the event had to be passed up to a higher level. This too is conducted through a delegate, *processGetTouchPoints* or *processGetTouchPointsStaff*. The UI elements use delegates in order to keep them as flexible and separate as possible for future use. The *Ring* class also contains methods that mathematically lay out the sectors on the ring. None of the application functionality is held within the *Ring* class.

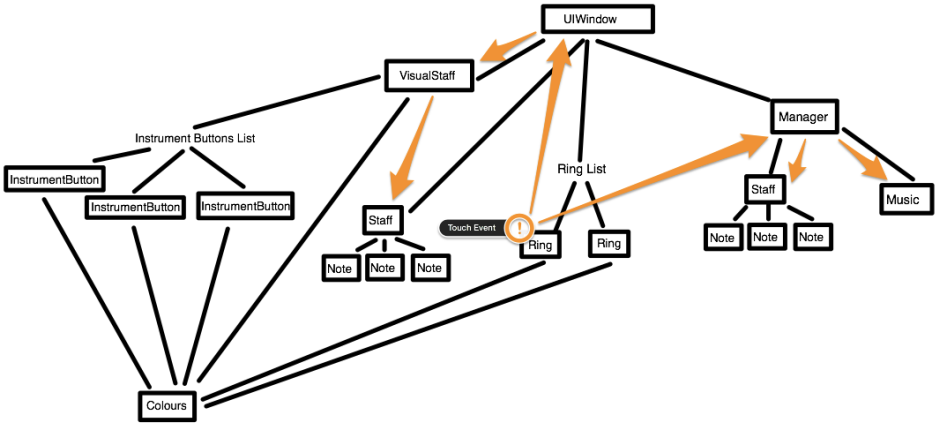


FIGURE 29: DIAGRAM OF HOW THE RING TOUCH EVENT NAVIGATES THE CLASS STRUCTURE

The VisualStaff Class

The *VisualStaff* class holds a bit more functionality but only what relates to changes within itself. Inside this class is a *Staff* object which contains a list of the notes that have been played stored in *Note* objects, a *SurfaceSlider* object, and the UI elements that make up the staff. It contains methods to setup the staff, an event handler for the *SurfaceSlider, onScroll,* and the *createNote* method which is called by the *Ring* delegate *processTouchDown*. This method creates the visual representation of the note and is one of three methods that are called by that delegate. The *onScroll* method changes the current locations of the notes stored in the *Staff* but maintains information of the true location in the *Staff* data structure. This along with checks to ensure the note is in the current viewing area allow the notes to be scrolled across the screen (see Figure 30). The instrument menu touch events are handled in the VisualStaff's *touch_Down_Button* method by changing the visibility of the InstrumentButtons using the *openMenu* delegate, but the InstrumentButton touch events are handled in a higher level classes because the *VisualStaff* does not have the information to handle their action.

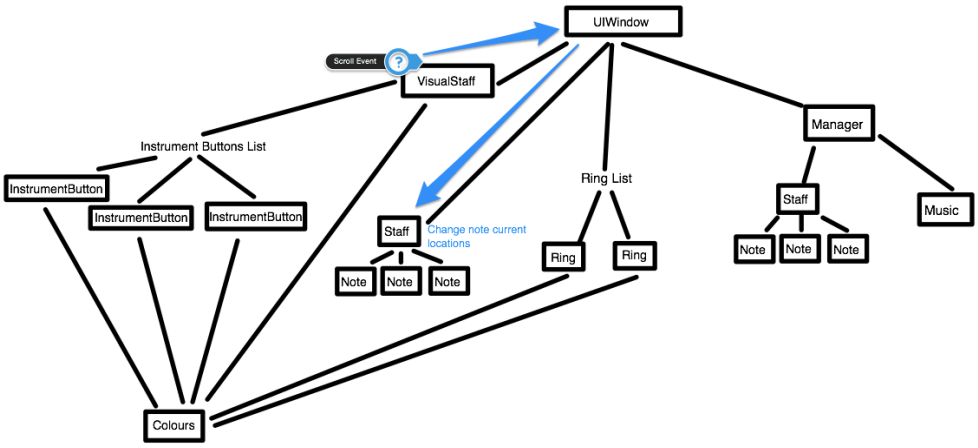


FIGURE 30: DIAGRAM OF HOW THE SCROLL EVENT NAVIGATES THE CLASS STRUCTURE

The UIWindow Class

In the *UIWindow* class, a *Manager* object is held to carry out the back-end functionality as well as a list of the current *Ring* objects, a list of the *InstrumentButton* objects, and the *VisualStaff*. This class contains event handlers for the top level touch events and the methods called by the *Ring* class's *processMove, processScale*, and *processRotate* delegates. It also creates the method which creates a new *Ring* object, deletes a *Ring* object, creates a *VisualStaff* object, configures the *InstrumentButton* list, and the method that is called by the *VisualStaff*'s delegate to open the instrument menu (see Figure 31). It also contains the methods called by the *processGetTouchPoints* and *processGetTouchPointsStaff* delegates in the *Ring* class. This class is the top level class and deals with the UI element operation and referring any functional activity to the *Manager* class.

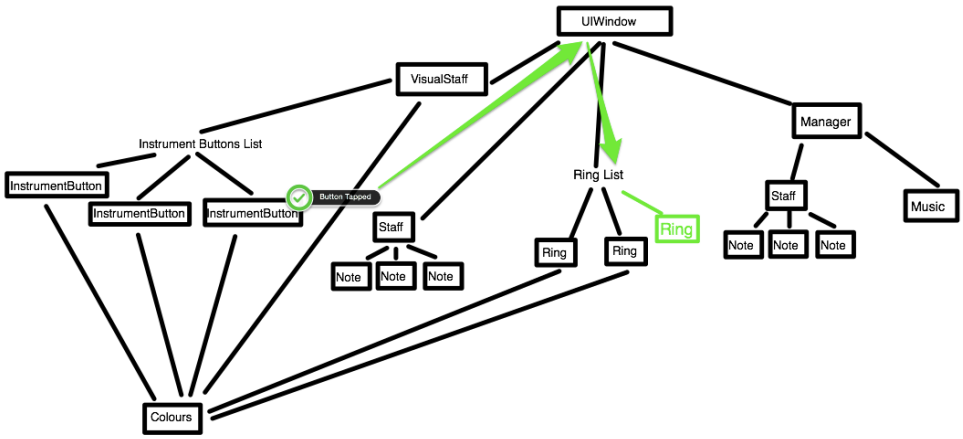


FIGURE 31: DIAGRAM OF HOW THE BUTTON TAPPED EVENT NAVIGATES THE CLASS STRUCTURE

The Back-End

Note Storage

The note information, with the exception of the UI information, is stored in *Note* objects which are held by the *UIWindow* class in a *Staff* object. The *Staff* object is made up of a list of *Note* objects. In each *Note* object is held the pitch, the length of the note, the start time, the end time, and the instrument that played it. Some of this information is not currently used but would have been needed for some of the objectives that were not achieved.

Music

The music in this application was generated using the MIDI library, Toub.Sound.Midi[11] from the Computer Science department at the University of Calgary. This library was chosen because it is written for Visual Studio, has clear and concise documentation, and was simple to use yet flexible enough to achieve what was needed. A note could be simply played using *NoteOn* and *NoteOff* commands which could be configured using default MIDI instruments or modified further to use more complex settings. In addition, other libraries that were tried worked on the Surface simulator but had trouble working on the table. Though the more complex settings in this library are not used in this project, using a library with the options enables for further extensions to be made if desired.

EVALUATION

Colouration

The colouration of the different elements in this application was a significant design decision. While the colours needed to be aesthetically pleasing, it also had to be usable. The colours have to be distinguishable from each other both when next to each other on the control ring and when separated on the visual staff. For this reason, I initially looked at ColorBrewer[15] to select colours but was not able to choose the number of colours necessary. Instead I used Adobe Kuler[14]. I created three different colour schemes: a dark one, a light one, and one with alternating colours from the two. These were tested on the implemented system and found to have different effectiveness. The dark colours were the least distinguishable both on the control ring and the visual staff. Between the light and mixed, however, there was a crossover. The light colours were less distinguishable on the control ring but better on the visual staff, whereas the mixed colours were easier to distinguish on the control ring but harder on the visual staff. This was evaluated primarily by me as no formal testing was conducted as a part of this project, however, it would be an interesting extension to experiment with more colour schemes to optimise usability or even include settings to aid those suffering from colour blindness.

Testing the Interface

This project involved no formal assessment or evaluation only the design and implementation of the system. To ensure different elements were performing as expected during the implementation phase, two methods were used. Much of the implementation was done on a separate computer because it was portable and more convenient to work on, so the Surface SDK 2.0 Simulator was used to test small changes. Touch inputs could be simulated using the SDK's Input Simulator which uses various types of mouse button combinations to act as touch inputs. When more thorough testing was necessary the application was run on the table either through Visual Studio on the Windows 7 side or through the Surface launcher on the Surface side. This allowed me to get more accurate feedback on how accurate the touch inputs were and how multi-touch was working. Through this process, I discovered that the sensitivity of the table was quite high as it was registering touch events from parts of the user's hands and arms when they were not touching the table but hovering over it. This kind of interaction severely interfered with the application's performance because many touch inputs were being read in all over the ring where no touches were actually occurring.

DISCUSSION

Achievements

Other musical multi-touch applications fall into a few different categories: note-based interfaces, synthesiser interfaces, transitional interface, and unstructured interfaces. Transitional interfaces provide notes on a gradual scale making it difficult to select a precise note such as in Spaces[1] and Vuzik[2]. Synthesiser interfaces allow for simple generation of digital sound similar to that used by DJs like in Reactable[3] and Xanakis[8]. Unstructured interfaces are more experimental than functional as they make it difficult to play specific notes or make music but instead generate more artistic sound as in WallBalls[7], Ah!Text[1], and Roots[1]. Note-based interfaces allow the user to choose which notes to play and what order to play them in like in ToCoPlay[5] and MelodyMorph[6]. This interface is most like the note-based projects because the user can play individual notes on the classical scale though it differs because unlike others, play can be conducted in real-time with less setup required. Other interfaces in this category cannot be performed on without dragging notes into place and arranging them into chords and patterns beforehand. With this application, a user can approach the table, pick an instrument and immediately begin playing music. It can be used as an instrument which would allow for more sophisticated play with practice in order to become familiar with the note placement and the shapes of different types of chords.

Value

With emerging technologies, each new project that approaches the situation differently is valuable. Each project acts like a brainstorming sketch, on a larger scale, that can help foster new ideas about how the technology can be used to the greatest benefit of the users. This project examines new interaction techniques for musical multi-touch applications which combine visualisation techniques with the note-based methods of previous work. In this way, it bridges one of the gaps that exists between the related works. Using this method, an application can both teach the user and allow them to experiment.

Difficulties Encountered

In this project some of the design had to be modified based on problems that were encountered during implementation. The design of the ring changed based on the fact that in order to make the coloured sections around the ring, sectors had to be made from Path objects meaning the entire sector was one unit and could not be made partially translucent. Other changes such as the number of settings based on the ring changed as well due to the size of the ring on the table. This change was made because there did not seem to be enough space to comfortably fit the controls. Another change that was made was to the visual staff design which changed from having a separate section for each instrument being used to having them all on one with different colours because with the number of instrument options the staff would have been too crowded. The complexity of the application was limited by the amount of time programming the basic interface took. Another factor that caused problems was the differences that exist between the Surface Simulator and the actual Surface platform. The window size, colours, and even reaction to touch inputs were different due to the difference of the hardware running it. The C# method of recording touch points based on the element not a global coordinate system made the

scaling and rotation of the ring a much larger challenge than it should have been. Without a touch drag event, scaling needed to have global point coordinates to prevent the scaling from being effected by the touch point leaving the control ring boundaries. Because they share an interaction technique, the scaling method interferes with the rotation in a way that makes them both ineffective, even though they both work independently. I discovered that, unfortunately, the Surface touch API does not provide touch IDs like some other multi-touch platforms. This led to the problems of ring movement and note play which caused the rings to jump positions if multiple rings' centres were touched at once and notes to not stop if the touch slides onto another sector, to not have a quick fix. These issue, along with the scrolling bug were never dealt with.

Reflection

Despite these challenges, much was accomplished in this project. I started with no experience with the language or platform and was able to create a functional application that fulfilled the primary objectives laid out at the commencement of the project. At this point in the project, I feel I have a much better idea of what is possible, what works, and what does not. Looking back and knowing what I do now, I would probably change some aspects of the original design and perhaps take a different approach, but I am still satisfied with what I have been able to achieve in the time allotted. There are some issues I was never able to deal with properly like the octave changing controls and post-play editing, but my goals were always going to be greater than my ability to achieve them. I feel I have gained insight into the designs of this type of system and would like to have an opportunity to apply this knowledge to a similar project in the future.

Future Work

Further extensions could be added to this project to broaden the uses for the application. With the rest of the secondary and tertiary objectives, such as percussion, replay, and ways to export pieces played on the table as sheet music, this application could be expanded to be a fully functional composing platform as well as an instrument. Adding functionality into the design to handle octave changes in a quick way that could be used during performance would also enable the instrument to be able to play more complex pieces live. Another extension that would allow the user to modify the notes after they have been played, possibly using interaction with the visual representations, would make the application better for composing music. As a touch application, the appeal could certainly be improved by adding more sophisticated graphics and "juice"[13] such as sound effects, a paper scroll-like appearance for the visual staff, and effects like a ripple effects when objects are dropped or genie effects when rings are created from the instrument bar or deleted. This project, however, focussed more on the functional and interaction attributes of the application rather than the graphics and feel.

CONCLUSION

This project has created a multi-touch interface for the Samsung Sur40 table that can produce sound in methods that mimic traditional instruments. The control ring allows the user to see basic relationships between the notes on the classical scale without classical training and is simple to use. The visual staff displays the notes as they are played providing information about the note, length, and instrument as well as allowing viewing of previously played notes through the scrollbar. The control ring can be moved, scaled, and rotated to make it more usable for the user at any position around the table. New rings for different instruments can be created and deleted using the collapsable instrument menu allowing users to work with multiple instruments simultaneously. Multiple notes from one or more rings can be played simultaneously without interfering with each other allowing the user to play chords or harmonies. The application is usable in live play but could be adapted to be useful in a composition setting.

Unfortunately, I was not able to implement all the design ideas I would have liked. I was never able to implement controls to change octaves or modify notes after they were played. Nor was I able to implement playback of previous composition or saving of states. One of my favourite ideas that I was unable to implement was the ability to export something that was played on the application as a document in the form of sheet music. This would have allowed the application to be used not only to experiment during composition but to actually create a piece of music that could be replicated on other instruments. With more specialised skills I would have been able to accomplish some of these things.

Overall this project created a functional music interface that employs interaction techniques that diverge from other projects of its genre. I hope others exploring this topic will find my work here useful.

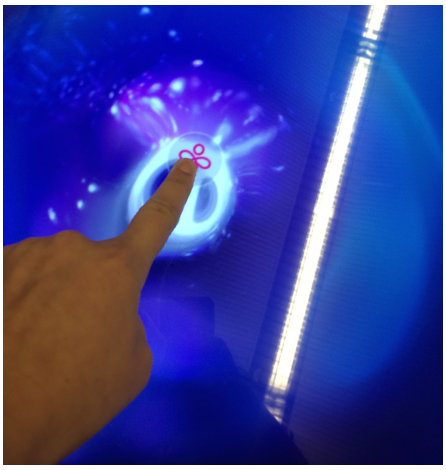
APPENDIX

MANUAL

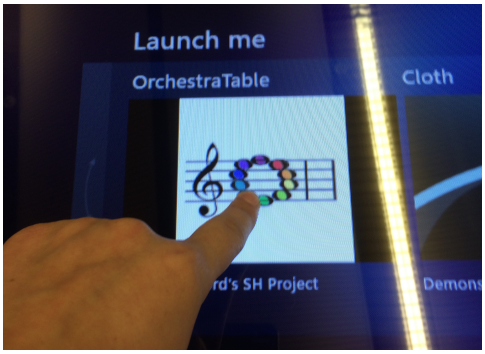
User Manual

Running the Application

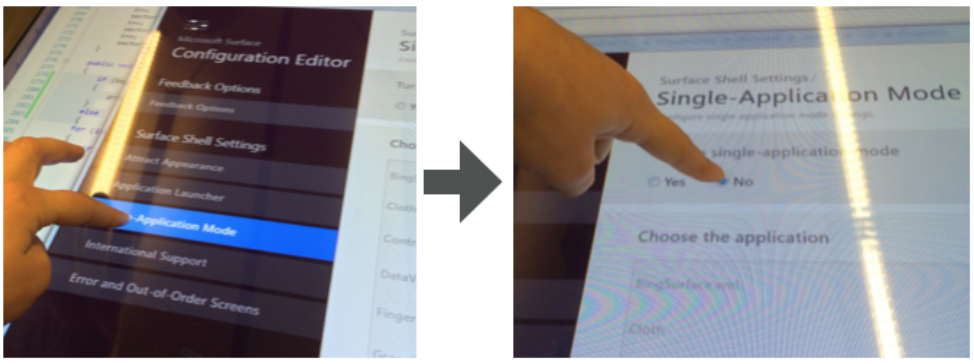
To run the application on in Microsoft Surface, go to the Surface Launcher. If the table is already running an application, see *Leaving the Application* then skip the next instruction. If the table is not currently running an application, this is done by tapping anywhere on the table and then tapping the circular logo in the centre of the table.



From the Surface Launcher scroll through the applications until OrchestraTable appears. Tap the icon for OrchestraTable, and the application will open.



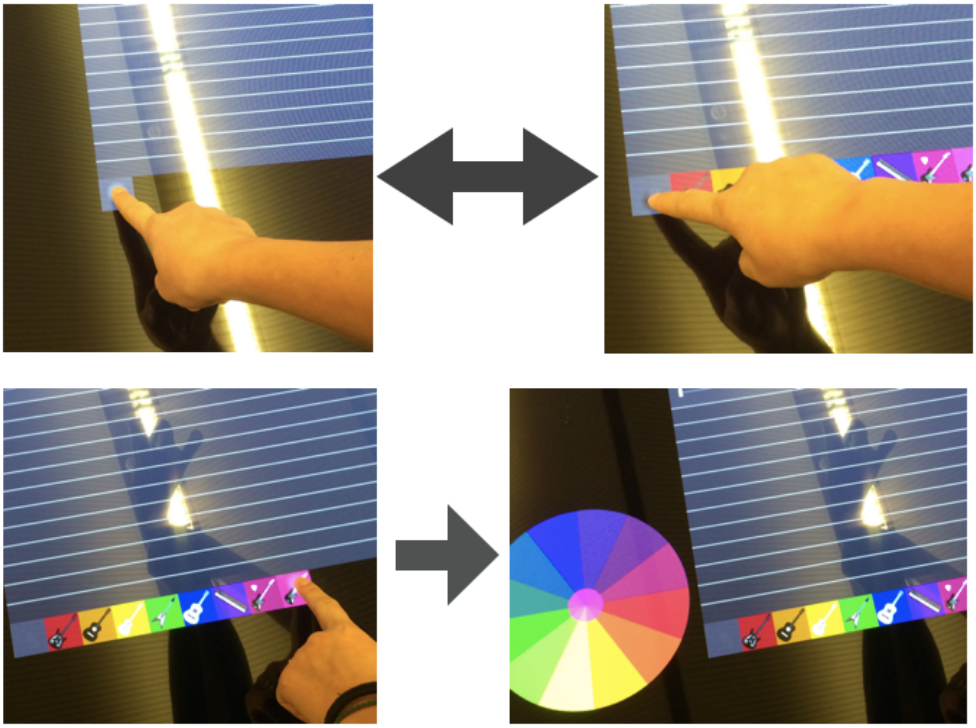
To take the table out of single application mode, open the Surface Configuration Editor and go to the Single-Application Mode page using the side navigation bar, then change the radio button from single application mode on to off.



Adding an Instrument

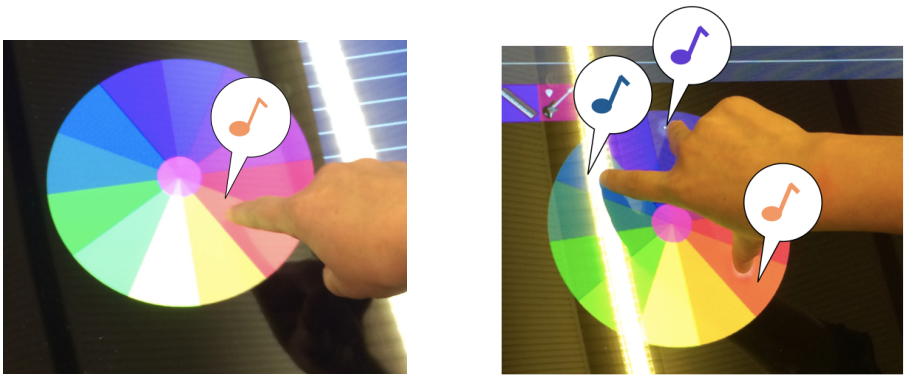
To add an instrument control ring to the table, tap the instrument menu button at the bottom of the visual staff to open the instrument list, if it is not already open, then tap the desired instrument in the list. A control ring with the instrument's colour in its centre circle will appear to the left of the screen. Please note that all control rings spawn in the same place so if multiple rings are created they will appear on top of each other and will need to be moved.

To close the instrument menu simple tap on the grey button again.

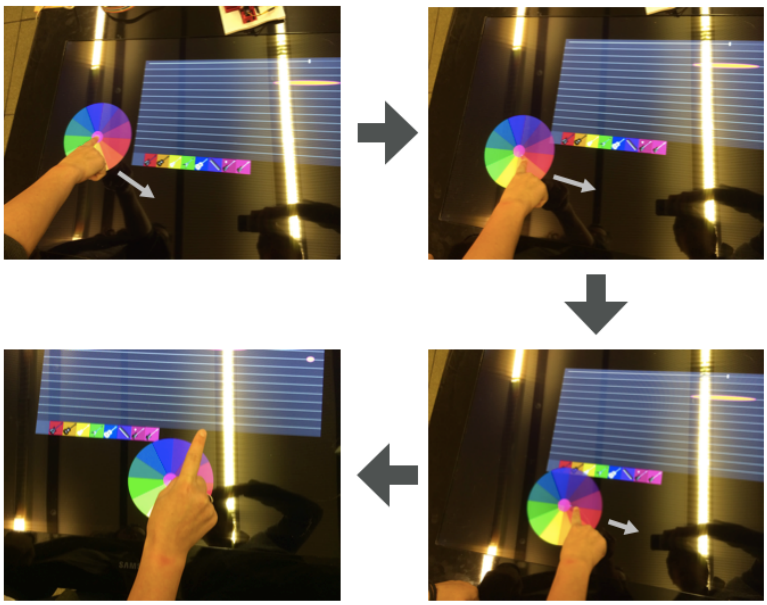


Using the Control Ring

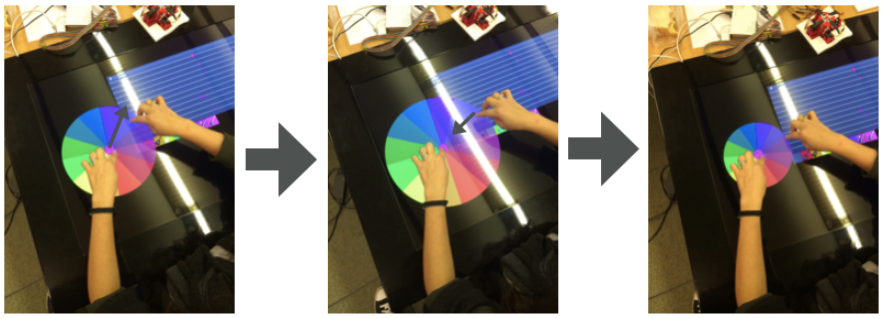
To make sound play, simply touch any one or more coloured sector of the control ring.



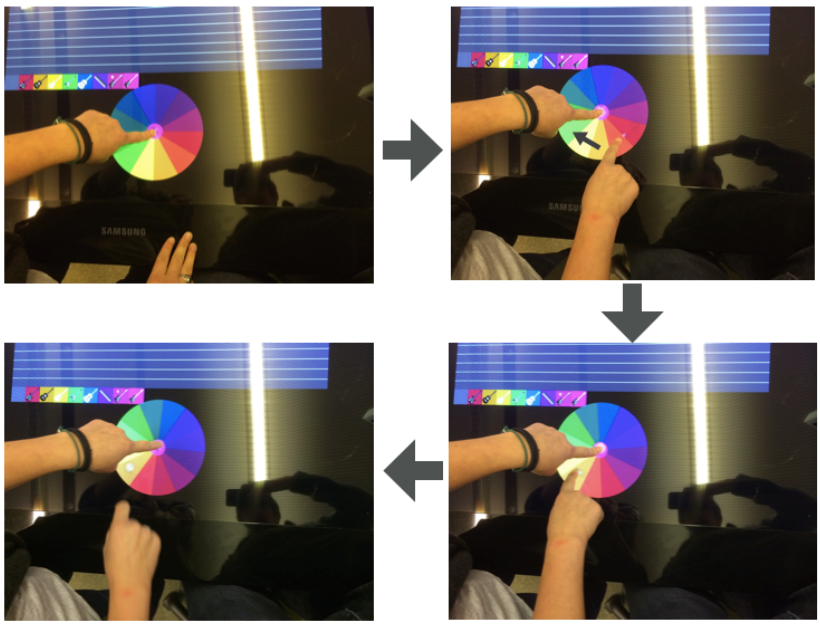
To move the control ring, touch down on the centre circle of the control ring and drag it to the new location. If you are having trouble moving the ring, try slowing down to allow the ring to follow your finger.



To scale the control ring, touch down on the centre circle of the control ring and, while maintaining the position of that finger, touch down on any other part of the ring with another finger and move the second finger towards or away from the first to shrink or expand the control ring, respectively.

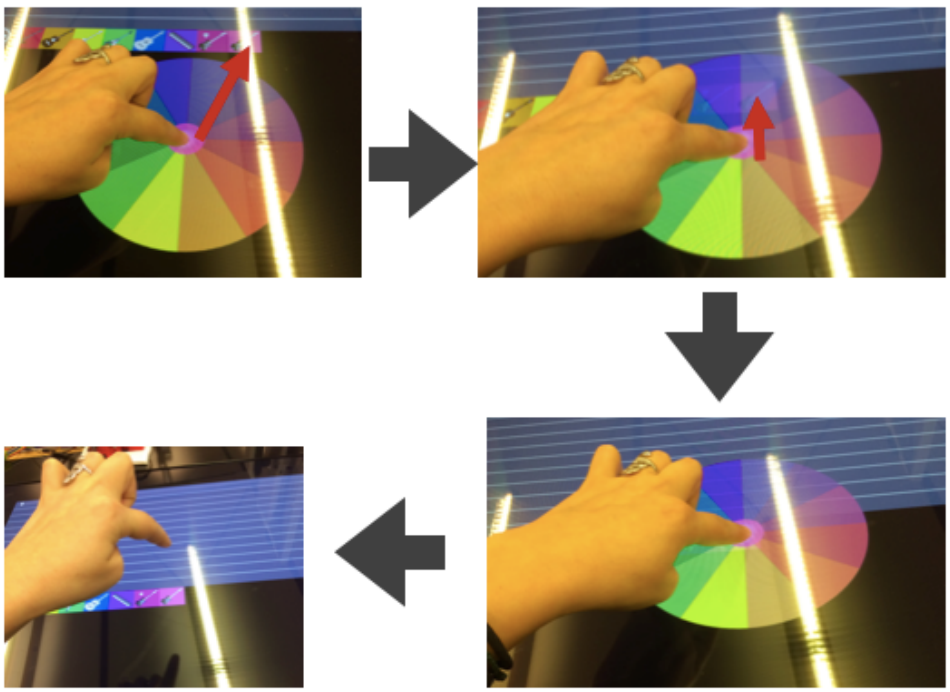


To rotate the control ring, touch down on the centre circle of the control ring and, while maintaining the position of that finger, touch down on any other part of the ring with another finger and move the second finger around the first, inside the ring, in the direction of desired rotation. Try not to move the second finger towards or away from the first too much otherwise scaling will occur as well as rotation.



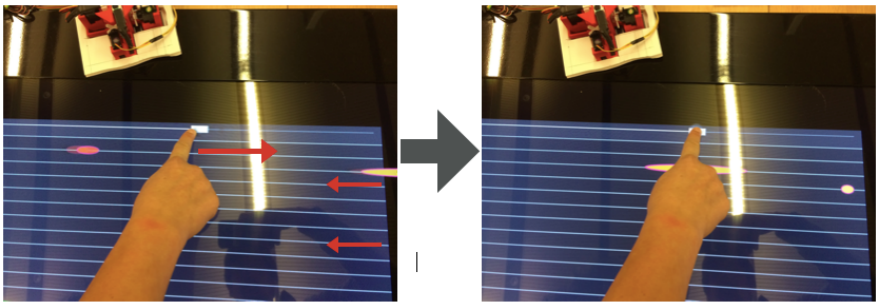
Removing an instrument

To remove an instrument control ring, make sure the instrument menu is open, if it is not tap on the grey button at the bottom of the visual staff, then touch and drag the centre circle of the control ring over the matching button and release. The control ring should disappear.



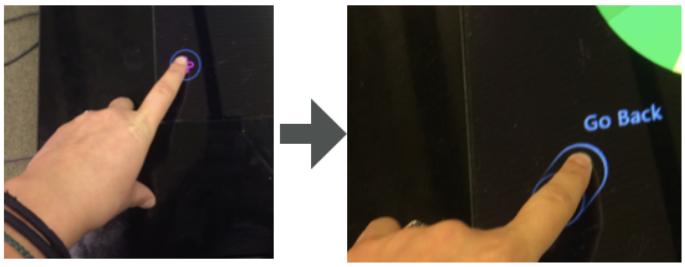
Using the Visual Staff

To scroll the visual staff, touch down on the scrollbar thumb, the small white box that shows the current position, and drag it to the desired position. As the thumb is moved, the notes will scroll by. If a new note is played it will disrupt the scrolling and jump to the current location so the user can see the current play.



Leaving the Application

To quit the application, touch in one of the corners of the table until the pink circle saying "Go Back" appears then tap the circle to exit back to the launcher. To leave the Surface side of the table, log out of the Surface account.



Maintenance Document

This project will run on the Microsoft Surface 2.0 platform on the Samsung Sur40 touch table. It may work on other devices running Microsoft Surface 2.0 but it has not been tested.

To install the application copy the Visual Studio file over to the new machine. Create a shortcut for the XML file, *SurfaceApplication2/SurfaceApplication2/bin/Release/OrchestraTable.xml*. Move the shortcut to the *C:/ProgramData/Microsoft/Surface/v2.0/Programs* file. Then open the *Surface Configuration Editor* and go into the *Application Launcher* page and move OrchestraTable into the applications to be displayed menu or tick the radio button which says display these programs and then the rest.

Now the program should appear when you open the Surface Launcher. If it does not check that you have chosen the correct XML file or check the configuration of the project.

If the project has not been run on the machine previously, the file paths in the *Colours* class must be changed to the correct full path.

RESOURCES

1. Jordan Hochenbaum, Owen Vallis, Dimitri Diakopoulos, Jim Murphy, Ajay Kapur, Designing Expressive Musical Interfaces for Tabletop Surfaces, Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 315–318, 2010.
2. Pon, Aura, Junko Ichino, Ehud Sharlin, and David Eagle. "Vuzik: Music Creation and Comprehension through Painting." ResearchGate. N.p., n.d. Web. 04 June 2013.
3. Jordà, Sergi. The Reactable: Tabletop Tangible Interfaces for Multithreaded Musical Performance. CiteSeerX — The Reactable: Tabletop Tangible Interfaces for Multithreaded Musical Performance. Music Technology Group, Pompeu Fabra University, n.d. Web. 04 June 2013.
4. Tony Bergstrom, Karrie Karahalios, and John C. Hart. 2007. Isochords: visualizing structure in music. In Proceedings of Graphics Interface 2007 (GI '07). ACM, New York, NY, USA, 297-304. DOI=10.1145/1268517.1268565 http://doi.acm.org/10.1145/1268517.1268565
5. Sean Lynch, Miguel A. Nacenta, and Sheelagh Carpendale. 2011. ToCoPlay: graphical multi-touch interaction for composing and playing music. In Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part III (INTERACT'11), Pedro Campos, Nuno Nunes, Nicholas Graham, Joaquim Jorge, and Philippe Palanque (Eds.), Vol. Part III. Springer-Verlag, Berlin, Heidelberg, 306-322. Eric Rosenbaum, MelodyMorph: A Reconfigurable Musical Instrument, Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 445–447, 2011.
6. Grant Partridge, Pourang Irani and Gordon Fitzell. 2009. Let loose with WallBalls, a collaborative tabletop instrument for tomorrow. In Proceedings of the 9th International Conference on New Interfaces for Musical Expression (NIME '09), 78-81.
7. Markus Bischof, Bettina Conradi, Peter Lachenmaier, Kai Linde, Max Meier, Philipp Pötzl, and Elisabeth André. 2008. Xenakis: combining tangible interaction with probability-based musical composition. In Proceedings of the 2nd international conference on Tangible and embedded interaction (TEI '08). ACM, New York, NY, USA, 121-124. DOI=10.1145/1347390.1347416 http://doi.acm.org/10.1145/1347390.1347416
8. Taintor, Callie. "Chronology: Technology and the Music Industry." PBS. PBS, 06 Feb. 2004. Web. 03 Apr. 2014.
9. Florent Berthaut, Haruhiro Katayose, Hironori Wakama, Naoyuki Totani, Yuichi Sato, First Person Shooters as Collaborative Multiprocess Instruments, Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 44–47, 2011.
10. "How To Play MIDI Instruments." ILab Cookbook -. Interactions Laboratory, University of Calgary, 15 Sept. 2009. Web. 29 Jan. 2014.
11. "Instrument." Def. 1. Merriam-Webster. Merriam-Webster, n.d. Web. 01 Apr. 2014.
12. Juice It or Lose It - a Talk by Martin Jonasson & Petri Purho. Perf. Martin Jonasson and Petri Purho. YouTube. TED Talks, 24 May 2012. Web. 12 Nov. 2013.
13. "Adobe Kuler." Adobe Kuler. Adobe Systems Inc., 2014. Web. 19 Feb. 2014.
14. Brewer, Cynthia, and Mark Harrower. "COLORBREWER 2.0." ColorBrewer: Color Advice for Maps. The Pennsylvania State University, n.d. Web. 19 Feb. 2014.
15. Hagen, Edward H., and Peter Hammerstein. "Did Neanderthals and Other Early Humans Sing? Seeking the Biological Roots of Music in the Territorial Advertisements of Primates, Lions, Hyenas, and Wolves." Musicae Scientiae 13.2 (2009): 291-320. SAGE Journals. SAGE Publications. Web. 03 Apr. 2014.
16. Ghosh, Pallab. "'Oldest Musical Instrument' Found." BBC News. BBC, 25 June 2009. Web. 03 Apr. 2014.

17. HURON, D. (2001), Is Music an Evolutionary Adaptation?. Annals of the New York Academy of Sciences, 930: 43–61. doi: 10.1111/j.1749-6632.2001.tb05724.x

18. Virgil. The Aeneid. N.p.: n.p., n.d. The Project Gutenberg. Project GutenBerg, 3 Apr. 2008. Web. 3 Apr. 2014.